Geometry: Combinatorics & Algorithms
Lecture Notes HS 2018

Luis Barba <luis.barba@inf.ethz.ch>
Bernd Gärtner <gaertner@inf.ethz.ch>
Michael Hoffmann <hoffmann@inf.ethz.ch>
Emo Welzl <emo@inf.ethz.ch>

August 2019

Preface

These lecture notes are designed to accompany a course on "Geometry: Combinatorics & Algorithms" that we teach at the Department of Computer Science, ETH Zürich, since 2005. The topics that are covered have changed over the years, as has the name of the course. The current version is a synthesis of topics from computational geometry, combinatorial geometry, and graph drawing that are centered around triangulations, that is, geometric representations of maximal plane graphs. The selection of topics is guided by the following criteria.

Importance. What are the most essential concepts and techniques that we want our students to know? (for instance, if they plan to write a thesis in the area)

Overlap. What is covered and to which extent in other courses of our curriculum?

Coherence. How closely related is something to the focal topic of triangulations and how well does it fit with the other topics selected?

Our main focus is on low-dimensional Euclidean space (mostly 2D), although we sometimes discuss possible extensions and/or remarkable differences when going to higher dimensions. At the end of each chapter there is a list of questions that we expect our students to be able to answer in the oral exam.

In the current setting, the course runs over 14 weeks, with two hours of lecture and two hours of exercises each week. In addition, there are three sets of graded homeworks which students have to hand in spread over the course. The target audience are third-year Bachelor or Master students of Mathematics or Computer Science.

This year's course covers the material presented in Chapters 1–10. The appendix A–H contains material that appeared in previous editions of the course.

Most parts of these notes have been gone through several iterations of proofreading over the years. But experience tells that there are always a few mistakes that escape detection. So in case you notice some problem, please let us know, regardless of whether it is a minor typo or punctuation error, a glitch in formulation, or a hole in an argument. This way the issue can be fixed for the next edition and future readers profit from your findings.

We thank Luis Barba, Kateřina Böhmová, Sergio Cabello, Tobias Christ, Cyril Frei, Anna Gundert, Tillmann Miltzow, Gabriel Nivasch, Johannes Obenaus, Júlia Pap, Kalina Petrova, Alexander Pilz, Patrick Schnider, Marek Sulovský, May Szedlák, Hemant Tyagi and Alexandra Wesolek for their helpful comments.

Bernd Gärtner, Michael Hoffmann, and Emo Welzl
Department of Computer Science, ETH Zürich
Universitätstrasse 6, CH-8092 Zürich, Switzerland
E-mail address: `{gaertner,hoffmann,emo}@inf.ethz.ch`

Contents

Chapter 1

Fundamentals

1.1 Models of Computation

When designing algorithms, one has to agree on a model of computation according to which these algorithms can be executed. There are various such models, but when it comes to geometry some are more convenient to work with than others. Even using very elementary geometric operations—such as taking the center of a circle defined by three points or computing the length of a given circular arc—the realms of rational and even algebraic numbers are quickly left behind. Representing the resulting real numbers/coordinates would be a rather painful task in, for instance, a Turing machine type model of computation.

Therefore, other models of computation are more prominent in the area of geometric algorithms and data structures. In this course we will be mostly concerned with two models: the *Real RAM* and the *algebraic computation/decision tree* model. The former is rather convenient when designing algorithms, because it sort of abstracts from the aforementioned representation issues by simply *assuming* that it can be done. The latter model typically appears in the context of lower bounds, that is, proofs that certain problems cannot be solved more efficiently than some function depending on the problem size (and possibly some other parameters).

So let us see what these models are in more detail.

Real RAM Model. A memory cell stores a real number (that is what the "Real" stands for)[1]. Any single arithmetic operation (addition, subtraction, multiplication, division, and k-th root, for small constant k) or comparison can be computed in constant time.[2] This is a quite powerful (and somewhat unrealistic) model of computation, as a single real number in principle can encode an arbitrary amount of information. Therefore we

[1]RAM stands for random access machine, meaning that every memory cell can be accessed in constant time. Not like, say, a list where one always has to start from the first element.

[2]In addition, sometimes also logarithms, other analytic functions, indirect addressing (integral), or floor and ceiling are used. As adding some of these operations makes the model more powerful, it is usually specified and emphasized explicitly when an algorithm uses them.

have to ensure that we do not abuse the power of this model. For instance, we may want to restrict the numbers that are manipulated by any single arithmetic operation to be bounded by some fixed polynomial in the numbers that appear in the input.

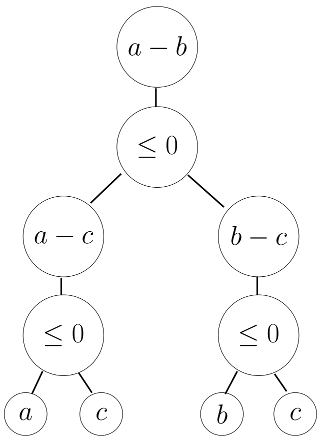
On the positive side, the real RAM model allows to abstract from the lowlands of numeric and algebraic computation and to concentrate on the algorithmic core from a combinatorial point of view.

But there are also downsides to using such a powerful model. In particular, it may be a challenge to efficiently implement a geometric algorithm designed for the real RAM on an actual computer. With bounded memory there is no way to represent general real numbers explicitly, and operations using a symbolic representation can hardly be considered constant time.

When interested in lower bounds, it is convenient to use a model of computation that encompasses and represents explicitly all possible execution paths of an algorithm. This is what the following model is about.

Algebraic Computation Trees (Ben-Or [1]). A computation is regarded as a binary tree.

- The leaves contain the (possible) results of the computation.
- Every node v with one child has an operation of the form $+, -, *, /, \sqrt{}, \ldots$ associated to it. The operands of this operation are constant input values, or among the ancestors of v in the tree.
- Every node v with two children has associated to it a branching of the form > 0, $\geqslant 0$, or $= 0$. The branch is with respect to the result of v's parent node. If the expression yields `true`, the computation continues with the left child of v; otherwise, it continues with the right child of v.



The term *decision tree* is used if all of the final results (leaves) are either `true` or `false`. If every branch is based on a linear function in the input values, we face a *linear decision tree*. Analogously one can define, say, quadratic decision trees.

The complexity of a computation or decision tree is the maximum number of vertices along any root-to-leaf path. It is well known that $\Omega(n \log n)$ comparisons are required to sort n numbers. But also for some problems that appear easier than sorting at first glance, the same lower bound holds. Consider, for instance, the following problem.

Element Uniqueness

Input: $\{x_1, \ldots, x_n\} \subset \mathbb{R}$, $n \in \mathbb{N}$.

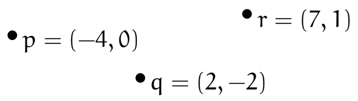
Output: Is $x_i = x_j$, for some $i, j \in \{1, \ldots, n\}$ with $i \neq j$?

Ben-Or [1] has shown that any algebraic decision tree to solve Element Uniqueness for n elements has complexity $\Omega(n \log n)$.

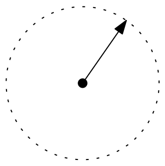
1.2 Basic Geometric Objects

We will mostly be concerned with the d-dimensional Euclidean space $\mathbb{R}^d$, for small $d \in \mathbb{N}$; typically, $d = 2$ or $d = 3$. The basic objects of interest in $\mathbb{R}^d$ are the following.

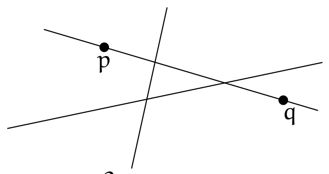
Points. A point p, typically described by its d Cartesian coordinates $p = (x_1, \ldots, x_d)$.



Directions. A vector $v \in \mathbb{S}^{d-1}$ (the $(d-1)$-dimensional unit sphere), typically described by its d Cartesian coordinates $v = (x_1, \ldots, x_d)$, with $\|v\| = \sqrt{\sum_{i=1}^{d} {x_i}^2} = 1$.

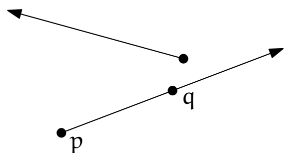


Lines. A line is a one-dimensional affine subspace. It can be described by two distinct points p and q as the set of all points r that satisfy $r = p + \lambda(q - p)$, for some $\lambda \in \mathbb{R}$.

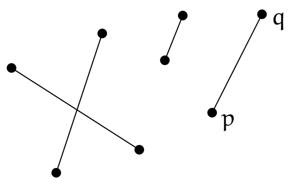


While any pair of distinct points defines a unique line, a line in $\mathbb{R}^2$ contains infinitely many points and so it may happen that a collection of three or more points lie on a line. Such a collection of points is termed *collinear*[3].

Rays. If we remove a single point from a line and take the closure of one of the connected components, then we obtain a ray. It can be described by two distinct points p and q as the set of all points r that satisfy $r = p + \lambda(q - p)$, for some $\lambda \geqslant 0$. The *orientation* of a ray is the direction $(q - p)/\|q - p\|$.



Line segment. A line segment is a compact connected subset of a line. It can be described by two points p and q as the set of all points r that satisfy $r = p + \lambda(q - p)$, for some $\lambda \in [0, 1]$. We will denote the line segment through p and q by $\overline{pq}$. Depending on the context we may allow or disallow *degenerate* line segments consisting of a single point only ($p = q$ in the above equation).



Hyperplanes. A hyperplane $\mathcal{H}$ is a $(d-1)$-dimensional affine subspace. It can be described algebraically by $d + 1$ coefficients $\lambda_1, \ldots, \lambda_{d+1} \in \mathbb{R}$, where $\|(\lambda_1, \ldots, \lambda_{d+1})\| = 1$, as the set of all points $(x_1, \ldots, x_d)$ that satisfy the linear equation $\mathcal{H} : \sum_{i=1}^{d} \lambda_i x_i = \lambda_{d+1}$.

[3]Not *colinear*, which refers to a notion in the theory of coalgebras.

If the above equation is converted into an inequality, we obtain the algebraic description of a *halfspace* (in $\mathbb{R}^2$: halfplane).

Spheres and balls. A sphere is the set of all points that are equidistant to a fixed point. It can be described by a point c (center) and a number $\rho \in \mathbb{R}$ (radius) as the set of all points p that satisfy $||p - c|| = \rho$. The *ball* of radius ρ around p consists of all points p that satisfy $||p - c|| \leqslant \rho$.

1.3 Graphs

In this section we review some basic definitions and properties of graphs. For more details and proofs, refer to any standard textbook on graph theory [2, 3, 5].

An (undirected) graph $G = (V, E)$ is defined on a set V of *vertices*. Unless explicitly stated otherwise, V is always finite. Vertices are associated to each other through *edges* which are collected in the set $E \subseteq \binom{V}{2}$. The two vertices defining an edge are *adjacent* to each other and *incident* to the edge.

For a vertex $v \in V$, denote by $N_G(v)$ the *neighborhood* of v in G, that is, the set of vertices from G that are adjacent to v. Similarly, for a set $W \subset V$ of vertices define $N_G(W) := \bigcup_{w \in W} N_G(w)$. The *degree* $\deg_G(v)$ of a vertex $v \in V$ is the size of its neighborhood, that is, the number of edges from E incident to v. The subscript is often omitted when it is clear which graph it refers to.

Lemma 1.1 (Handshaking Lemma). *In any graph* $G = (V, E)$ *we have* $\sum_{v \in V} \deg(v) = 2|E|$.

Two graphs $G = (V, E)$ and $H = (U, W)$ are *isomorphic* if there is a bijection $\phi : V \to U$ such that $\{u, v\} \in E \iff \{\phi(u), \phi(v)\} \in W$. Such a bijection ϕ is called an *isomorphism* between G and H. The structure of isomorphic graphs is identical and often we do not distinguish between them when looking at them as graphs.

For a graph G denote by $V(G)$ the set of vertices and by $E(G)$ the set of edges. A graph $H = (U, F)$ is a *subgraph* of G if $U \subseteq V$ and $F \subseteq E$. In case that $U = V$ the graph H is a *spanning* subgraph of G. For a set $W \subseteq V$ of vertices denote by $G[W]$ the *induced subgraph* of W in G, that is, the graph $(W, E \cap \binom{W}{2})$. For $F \subseteq E$ let $G \setminus F := (V, E \setminus F)$. Similarly, for $W \subseteq V$ let $G \setminus W := G[V \setminus W]$. In particular, for a vertex or edge $x \in V \cup E$ we write $G \setminus x$ for $G \setminus \{x\}$. The *union* of two graphs $G = (V, E)$ and $H = (W, F)$ is the graph $G \cup H := (V \cup W, E \cup F)$.

For an edge $e = \{u, v\} \in E$ the graph G/e is obtained from $G \setminus \{u, v\}$ by adding a new vertex w with $N_{G/e}(w) := (N_G(u) \cup N_G(v)) \setminus \{u, v\}$. This process is called *contraction* of e in G. Similarly, for a set $F \subseteq E$ of edges the graph G/F is obtained from G by contracting all edges from F (the order in which the edges from F are contracted does not matter).

Graph traversals. A *walk* in G is a sequence $W = (v_1, \ldots, v_k)$, $k \in \mathbb{N}$, of vertices such that v_i and v_{i+1} are adjacent in G, for all $1 \leqslant i < k$. The vertices v_1 and v_k are referred to as the walk's *endpoints*, the other vertices are called *interior*. A walk with endpoints v_1 and v_k is sometimes referred to as a *walk between* v_1 and v_k. For a walk W denote by $V(W)$ its set of vertices and by $E(W)$ its set of edges (pairs of vertices adjacent along W). We say that W *visits* the vertices and edges in $V(W) \cup E(W)$. A walk for which both endpoints coincide, that is, $v_1 = v_k$, is called *closed*. Otherwise the walk is *open*.

If a walk uses each edge of G at most once, it is a *trail*. A closed walk that visits each edge and each vertex at least once is called a *tour* of G. An *Euler tour* is both a trail and a tour of G, that is, it visits each edge of G exactly once. A graph that contains an Euler tour is termed *Eulerian*.

If the vertices $v_1, \ldots, v_k$ of a closed walk W are pairwise distinct except for $v_1 = v_k$, then W is a *cycle* of size $k - 1$. If the vertices $v_1, \ldots, v_k$ of a walk W are pairwise distinct, then W is a *path* of size k. A *Hamilton cycle (path)* is a cycle (path) that visits every vertex of G. A graph that contains a Hamilton cycle is *Hamiltonian*.

Two trails are *edge-disjoint* if they do not share any edge. Two paths are called (internally) *vertex-disjoint* if they do not share any vertices (except for possibly common endpoints). For two vertices $s, t \in V$ any path with endpoints s and t is called an (s, t)-*path* or a *path between* s and t.

Connectivity. Define an equivalence relation "$\sim$" on V by setting $a \sim b$ if and only if there is a path between a and b in G. The equivalence classes with respect to "$\sim$" are called *components* of G and their number is denoted by $\omega(G)$. A graph G is *connected* if $\omega(G) = 1$ and *disconnected*, otherwise.

A set $C \subset V$ of vertices in a connected graph $G = (V, E)$ is a *cut-set* of G if $G \setminus C$ is disconnected. A graph is *k-connected*, for a positive integer k, if $|V| \geqslant k + 1$ and there is no cut-set of size less than k. Similarly a graph $G = (V, E)$ is *k-edge-connected*, if $G \setminus F$ is connected, for any set $F \subseteq E$ of less than k edges. Connectivity and cut-sets are related via the following well-known theorem.

Theorem 1.2 (Menger [4]). *For any two nonadjacent vertices u, v of a graph $G = (V, E)$, the size of a minimum cut that disconnects u and v is the same as the maximum number of pairwise internally vertex-disjoint paths between u and v.*

Specific families of graphs. A graph with a maximum number of edges, that is, $(V, \binom{V}{2})$, is called a *clique*. Up to isomorphism there is only one clique on n vertices; it is referred to as the *complete graph* K_n, $n \in \mathbb{N}$. At the other extreme, the *empty graph* $\overline{K_n}$ consists of n isolated vertices that are not connected by any edge. A set U of vertices in a graph G is *independent* if $G[U]$ is an empty graph. A graph whose vertex set can be partitioned into at most two independent sets is *bipartite*. An equivalent characterization states that a graph is bipartite if and only if it does not contain any odd cycle. The bipartite graphs with a maximum number of edges (unique up to isomorphism) are the *complete* *bipartite graphs* $K_{m,n}$, for $m, n \in \mathbb{N}$. They consist of two disjoint independent sets of size m and n, respectively, and all mn edges in between.

A *forest* is a graph that is *acyclic*, that is, it does not contain any cycle. A connected forest is called *tree* and its *leaves* are the vertices of degree one. Every connected graph contains a spanning subgraph which is a tree, a so called *spanning tree*. Beyond the definition given above, there are several equivalent characterizations of trees.

Theorem 1.3. *The following statements for a graph* G *are equivalent.*

(1) G *is a tree (i.e., it is connected and acyclic).*

(2) G *is a connected graph with* n *vertices and* $n-1$ *edges.*

(3) G *is an acyclic graph with* n *vertices and* $n-1$ *edges.*

(4) Any two vertices in G *are connected by a unique path.*

(5) G *is minimally (edge-)connected, that is,* G *is connected but removal of any single edge yields a disconnected graph.*

(6) G *is maximally acyclic, that is,* G *is acyclic but adding any single edge creates a cycle.*

Directed graphs. In a directed graph or, short, *digraph* $D = (V, E)$ the set E consists of ordered pairs of vertices, that is, $E \subseteq V^2$. The elements of E are referred to as *arcs*. An arc $(u, v) \in E$ is said to be directed from its *source* u to its *target* v. For $(u, v) \in E$ we also say "there is an arc from u to v in D". Usually, we consider *loop-free* graphs, that is, arcs of the type (v, v), for some $v \in V$, are not allowed.

The *in-degree* $\deg_D^-(v) := |\{(u,v) \mid (u,v) \in E\}|$ of a vertex $v \in V$ is the number of *incoming* arcs at v. Similarly, the *out-degree* $\deg_D^+(v) := |\{(v,u) \mid (v,u) \in E\}|$ of a vertex $v \in V$ is the number of *outgoing* arcs at v. Again the subscript is often omitted when the graph under consideration is clear from the context.

From any undirected graph G one can obtain a digraph on the same vertex set by specifying a direction for each edge of G. Each of these $2^{|E(G)|}$ different digraphs is called an *orientation* of G. Similarly every digraph $D = (V, E)$ has an *underlying* undirected graph $G = (V, \{\{u,v\} \mid (u,v) \in E \text{ or } (v,u) \in E\})$. Hence most of the terminology for undirected graphs carries over to digraphs.

A *directed walk* in a digraph D is a sequence $W = (v_1, \ldots, v_k)$, for some $k \in \mathbb{N}$, of vertices such that there is an arc from v_i to v_{i+1} in D, for all $1 \leqslant i < k$. In the same way we define *directed trails*, *directed paths*, *directed cycles*, and *directed tours*.

References

[1] Michael Ben-Or, Lower bounds for algebraic computation trees. In *Proc. 15th Annu. ACM Sympos. Theory Comput.*, pp. 80–86, 1983.

[2] John Adrian Bondy and U. S. R. Murty, *Graph theory*, vol. 244 of *Graduate texts in Mathematics*, Springer-Verlag, London, 2008.

[3] Reinhard Diestel, *Graph theory*, vol. 173 of *Graduate texts in Mathematics*, Springer-Verlag, Heidelberg, 5th edn., 2016.

[4] Karl Menger, Zur allgemeinen Kurventheorie. *Fund. Math.*, **10**, 1, (1927), 96–115.

[5] Douglas B. West, *An introduction to graph theory*, Prentice Hall, Upper Saddle River, NJ, 2nd edn., 2001.

Chapter 2

Plane Embeddings

Graphs can be represented in a variety of ways, for instance, as an adjacency matrix or using adjacency lists. In this chapter we explore another type of representations that are quite different in nature, namely *geometric* representations of graphs. Geometric representations are appealing because they allow to visualize a graph along with a variety of its properties in a succinct manner. There are many degrees of freedom in selecting the type of geometric objects and the details of their geometry. This freedom allows to tailor the representation to meet specific goals, such as emphasizing certain structural aspects of the graph at hand or reducing the complexity of the obtained representation.

The most common type of geometric graph representation is a *drawing*, where vertices are mapped to points and edges to curves. Making such a map injective by avoiding edge crossings is desirable, both from a mathematically aesthetic point of view and for the sake of the practical readability of the drawing. Those graphs that allow such an *embedding* into the Euclidean plane are known as *planar*. Our goal in the following is to study the interplay between abstract planar graphs and their plane embeddings. Specifically, we want to answer the following questions:

- What is the combinatorial complexity of planar graphs (number of edges and faces)?
- Under which conditions are plane embeddings unique (in a certain sense)?
- How can we represent plane embeddings (in a data structure)?
- What is the geometric complexity of plane embeddings, that is, can we bound the size of the coordinates used and the complexity of the geometric objects used to represent edges?

Most definitions we use directly extend to multigraphs. But for simplicity, we use the term "graph" throughout.

2.1 Drawings, Embeddings and Planarity

A *curve* is a set $C \subset \mathbb{R}^2$ that is of the form $\{\gamma(t) : 0 \leqslant t \leqslant 1\}$, where $\gamma : [0, 1] \to \mathbb{R}^2$ is a continuous function. The function γ is called a *parameterization* of C. The points $\gamma(0)$

and $\gamma(1)$ are the *endpoints* of the curve. For a *closed* curve, we have $\gamma(0) = \gamma(1)$. A curve is *simple*, if it admits a parameterization γ that is injective on $[0, 1]$. For a closed simple curve we allow as an exception that $\gamma(0) = \gamma(1)$. The following famous theorem describes an important property of the plane. A proof can, for instance, be found in the book of Mohar and Thomassen [22].

Theorem 2.1 (Jordan). *Any simple closed curve C partitions the plane into exactly two regions (connected open sets), each bounded by C.*

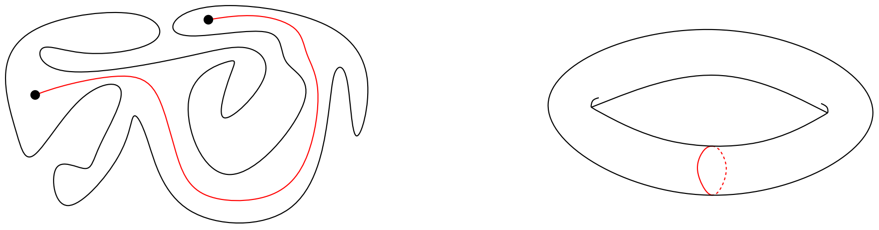


Figure 2.1: *A Jordan curve and two points in one of its faces (left); a simple closed curve that does not disconnect the torus (right).*

Observe that, for instance, on the torus there are closed curves that do not disconnect the surface (and so the theorem does not hold there).

Drawings. As a first criterion for a reasonable geometric representation of a graph, we would like to have a clear separation between different vertices and also between a vertex and nonincident edges. Formally, a *drawing* of a graph $G = (V, E)$ in the plane is a function f that assigns

- a point $f(v) \in \mathbb{R}^2$ to every vertex $v \in V$ and
- a simple curve $f(\{u, v\}) : [0, 1] \to \mathbb{R}^2$ with endpoints $f(u)$ and $f(v)$ to every edge $\{u, v\} \in E$,

such that

(1) f is injective on V and

(2) $f(\{u, v\}) \cap f(V) = \{f(u), f(v)\}$, for every edge $\{u, v\} \in E$.

A common point $f(e) \cap f(e')$ between two curves that represent distinct edges $e, e' \in E$ is called a *crossing* if it is not a common endpoint of e and e'.

For simplicity, when discussing a drawing of a graph $G = (V, E)$ it is common to treat vertices and edges as geometric objects. That is, a vertex $v \in V$ is treated as the point $f(v)$ and an edge $e \in E$ is treated as the curve $f(e)$. For instance, the last sentence of the previous paragraph may be phrased as "A common point of two edges that is not a common endpoint is called a crossing."

Often it is convenient to make additional assumptions about the interaction of edges in a drawing. For example, in a nondegenerate drawing one may demand that no three edges share a single crossing or that every pair of distinct edges intersects in at most finitely many points.

Planar vs. plane. A graph is *planar* if it admits a drawing in the plane without crossings. Such a drawing is also called a *crossing-free* drawing or a (plane) *embedding* of the graph. A planar graph together with a particular plane embedding is called a *plane* graph. Note the distinction between "planar" and "plane": the former indicates the possibility of an embedding, whereas the latter refers to a concrete embedding (Figure 2.2).

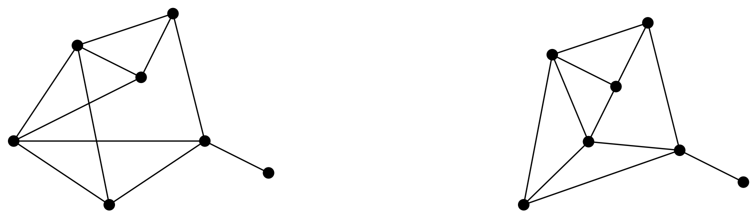


Figure 2.2: *A planar graph (left) and a plane embedding of it (right).*

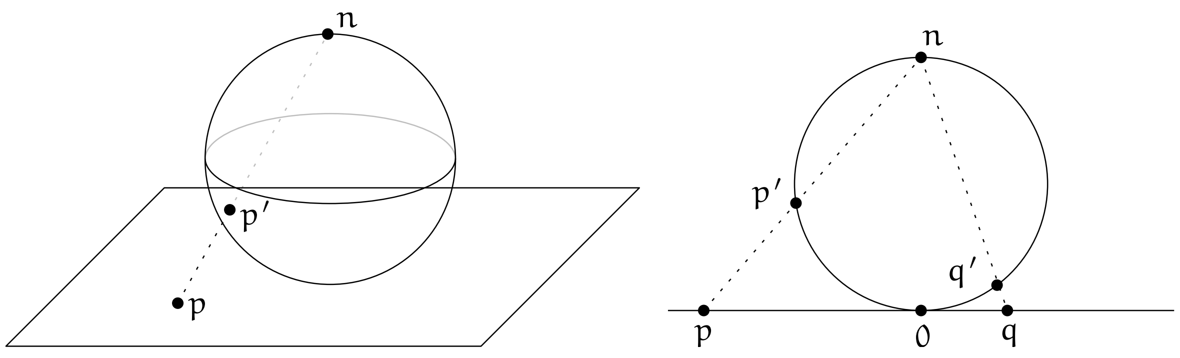
A *geometric graph* is a graph together with a drawing, in which all edges are realized as straight-line segments. Note that such a drawing is completely defined by the mapping for the vertices. A plane geometric graph is also called a *plane straight-line graph* (PSLG). In contrast, a plane graph in which the edges may form arbitrary simple curves is called a *topological plane graph*.

The *faces* of a plane graph are the maximally connected regions of the plane that do not contain any point used by the embedding (as the image of a vertex or an edge). Each embedding of a finite graph has exactly one *unbounded face*, also called *outer* or *infinite* face. Using stereographic projection, it is not hard to show that the role of the unbounded face is not as special as it may seem at first glance.

Theorem 2.2. *If a graph* G *has a plane embedding in which some face is bounded by the cycle* $(v_1, \ldots, v_k)$, *then* G *also has a plane embedding in which the unbounded face is bounded by the cycle* $(v_1, \ldots, v_k)$.

Proof. **(Sketch)** Take a plane embedding Γ of G and map it to the sphere using *stereographic projection*: Imagine $\mathbb{R}^2$ being the x/y-plane in $\mathbb{R}^3$ and place a unit sphere S such that its south pole touches the origin. We obtain a bijective continuous map between $\mathbb{R}^2$ and $S \setminus \{n\}$, where n is the north pole of S, as follows: A point $p \in \mathbb{R}^2$ is mapped to the point p' that is the intersection of the line through p and n with S, see Figure 2.3.

Consider the resulting embedding Γ' of G on S: The infinite face of Γ corresponds to the face of Γ' that contains the north pole n of S. Now rotate the embedding Γ' on

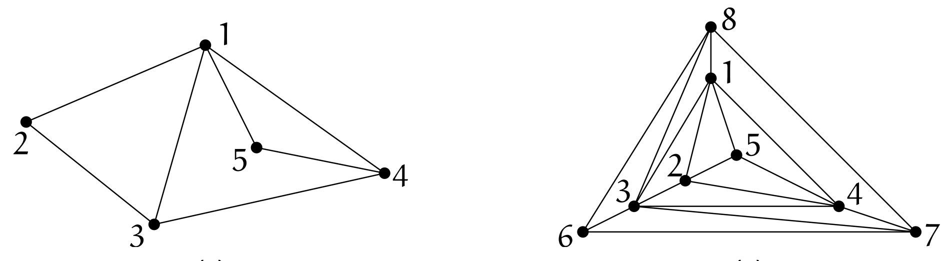


(a) Three-dimensional view. (b) Cross-section view.

Figure 2.3: *Stereographic projection.*

S such that the desired face contains n. Mapping back to the plane using stereographic projection results in an embedding in which the desired face is the outer face. □

Exercise 2.3. *Consider the plane graphs depicted in Figure 2.4. For both graphs give a plane embedding in which the cycle* $1, 2, 3$ *bounds the outer face.*



(a) (b)

Figure 2.4: *Make* $1, 2, 3$ *bound the outer face.*

Duality. Every plane graph G has a *dual* G^*, whose vertices are the faces of G and two are connected by an edge in G^*, if and only if they have a common edge in G. In general, G^* is a multigraph (may contain loops and multiple edges) and it depends on the embedding. That is, an abstract planar graph G may have several nonisomorphic duals. If G is a connected plane graph, then $(G^*)^* = G$. We will see later in Section 2.3 that the dual of a 3-connected planar graph is unique (up to isomorphism).

The Euler Formula and its ramifications. One of the most important tools for planar graphs (and more generally, graphs embedded on a surface) is the Euler–Poincaré Formula.

Theorem 2.4 (Euler's Formula). *For every connected plane graph with* n *vertices,* e *edges, and* f *faces, we have* $n - e + f = 2$.

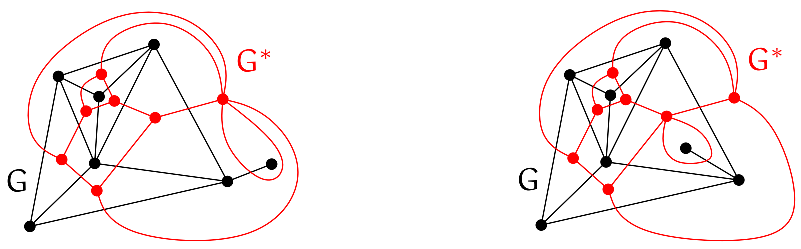


Figure 2.5: *Two plane drawings and their duals for the same planar graph.*

In particular, this shows that for any planar graph the number of faces is the same in every plane embedding. In other words the number of faces is an invariant of an abstract planar graph. It also follows (stated below as a corollary) that planar graphs are *sparse*, that is, they have a linear number of edges (and faces) only. So the asymptotic complexity of a planar graph is already determined by its number of vertices.

Corollary 2.5. *A simple planar graph on $n \geqslant 3$ vertices has at most $3n-6$ edges and at most $2n-4$ faces.*

Proof. Without loss of generality we may assume that G is connected. (If not, add edges between components of G until the graph is connected. The number of faces remains unchanged and the number of edges only increases.) The statement is easily checked for $n = 3$, where G is either a triangle or a path and, therefore, has no more than $3 \cdot 3 - 6 = 3$ edges and no more than $2 \cdot 3 - 4 = 2$ faces. So consider a simple planar graph G on $n \geqslant 4$ vertices. Consider a plane drawing of G and denote by E the set of edges and by F the set of faces of G. Let

$$X = \{(e, f) \in E \times F : e \text{ bounds } f\}$$

denote the set of incident edge-face pairs. We count X in two different ways.

First note that each edge bounds at most two faces and so $|X| \leqslant 2 \cdot |E|$.

Second note that in a simple connected planar graph on four or more vertices every face is bounded by at least three vertices: Every bounded face needs at least three edges to be enclosed and if there is no cycle on the boundary of the unbounded face, then—given that G is connected—G must be a tree on four or more vertices and so it has at least three edges, all of which bound the unbounded face. Therefore $|X| \geqslant 3 \cdot |F|$.

Using Euler's Formula we conclude that

$$\begin{aligned} 4 &= 2n - 2|E| + 2|F| \leqslant 2n - 3|F| + 2|F| = 2n - |F| \text{ and} \\ 6 &= 3n - 3|E| + 3|F| \leqslant 3n - 3|E| + 2|E| = 3n - |E| , \end{aligned}$$

which yields the claimed bounds. □

It also follows that the degree of a "typical" vertex in a planar graph is a small constant. There exist several variations of this statement, a few more of which we will encounter during this course.

Corollary 2.6. *The average vertex degree in a simple planar graph is less than six.*

Exercise 2.7. *Prove Corollary 2.6.*

Exercise 2.8. *Show that neither* K_5 *(the complete graph on five vertices) nor* $K_{3,3}$ *(the complete bipartite graph where both classes have three vertices) is planar.*

Characterizing planarity. The classical theorems of Kuratowski and Wagner provide a characterization of planar graphs in terms of forbidden substructures. A *subdivision* of a graph $G = (V, E)$ is a graph that is obtained from G by replacing each edge with a path.

Theorem 2.9 (Kuratowski [20, 27]). *A graph is planar if and only if it does not contain a subdivision of* $K_{3,3}$ *or* K_5*.*

A *minor* of a graph $G = (V, E)$ is a graph that is obtained from G using zero or more edge contractions, edge deletions, and/or vertex deletions.

Theorem 2.10 (Wagner [30]). *A graph is planar if and only if it does not contain* $K_{3,3}$ *or* K_5 *as a minor.*

In some sense, Wagner's Theorem is a special instance[1] of a much more general theorem.

Theorem 2.11 (Graph Minor Theorem, Robertson/Seymour [25]). *Every minor-closed family of graphs can be described in terms of a finite set of forbidden minors.*

Being *minor-closed* means that for every graph from the family also all of its minors belong to the family. For instance, the family of planar graphs is minor-closed because planarity is preserved under removal of edges and vertices and under edge contractions. The Graph Minor Theorem is a celebrated result that was established by Robertson and Seymour in a series of twenty papers, see also the survey by Lovász [21]. They also describe an $O(n^3)$ algorithm (with horrendous constants, though) to decide whether a graph on n vertices contains a fixed (constant-size) minor. Later, Kawarabayashi et al. [18] showed that this problem can be solved in $O(n^2)$ time. As a consequence, every minor-closed property can be decided in polynomial time.

Unfortunately, the result is nonconstructive in the sense that in general we do not know how to obtain the set of forbidden minors for a given family/property. For instance, for the family of toroidal graphs (graphs that can be embedded without crossings on the torus) more than $16'000$ forbidden minors are known, and we do not know how many there are in total. So while we know that there exists a quadratic time algorithm to test membership for minor-closed families, we have no idea what such an algorithm looks like in general.

Graph families other than planar graphs for which the forbidden minors are known include forests (K_3) and outerplanar graphs ($K_{2,3}$ and K_4). A graph is *outerplanar* if it admits a plane embedding such that all vertices appear on the outer face (Figure 2.6).

[1]It is more than just a special instance because it also specifies the forbidden minors explicitly.

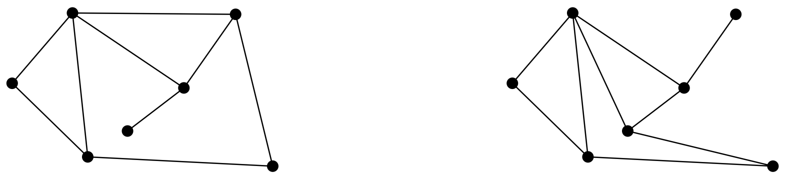


Figure 2.6: *An outerplanar graph (left) and a plane embedding of it in which all vertices are incident to the outer face (right).*

Exercise 2.12. *(a) Give an example of a 6-connected planar graph or argue that no such graph exists.*

(b) Give an example of a 5-connected planar graph or argue that no such graph exists.

(c) Give an example of a 3-connected outerplanar graph or argue that no such graph exists.

Planarity testing. For planar graphs we do not have to contend ourselves with a cubic-time algorithm, as there are several approaches to solve the problem in linear time. In fact, there is quite a number of papers that describe different linear time algorithms, all of which—from a very high-level point of view—can be regarded as an annotated depth-first-search. The first such algorithm was described by Hopcroft and Tarjan [17], while the current state-of-the-art is probably among the "path searching" method by Boyer and Myrwold [5] and the "LR-partition" method by de Fraysseix et al [13]. Although the overall idea in all these approaches is easy to convey, there are many technical details, which make an in-depth discussion rather painful to go through.

2.2 Graph Representations

There are two standard representations for an abstract graph $G = (V, E)$ on $n = |V|$ vertices. For the *adjacency matrix* representation we consider the vertices to be ordered as $V = \{v_1, \ldots, v_n\}$. The adjacency matrix of an undirected graph is a symmetric $n \times n$-matrix $A = (a_{ij})_{1 \leqslant i,j \leqslant n}$ where $a_{ij} = a_{ji} = 1$, if $\{i, j\} \in E$, and $a_{ij} = a_{ji} = 0$, otherwise. Storing such a matrix explicitly requires $\Omega(n^2)$ space, and allows to test in constant time whether or not two given vertices are adjacent.

In an *adjacency list* representation, we store for each vertex a list of its neighbors in G. This requires only $O(n+|E|)$ storage, which is better than for the adjacency matrix in case that $|E| = o(n^2)$. On the other hand, the adjacency test for two given vertices is not a constant-time operation, because it requires a search in one of the lists. Depending on the representation of these lists, such a search takes $O(d)$ time (unsorted list) or $O(\log d)$ time (sorted random-access representation, such as a balanced search tree), where d is the minimum degree of the two vertices.

Both representations have their merits. The choice of which one to use (if any) typically depends on what one wants to do with the graph. When dealing with embedded graphs, however, additional information concerning the embedding is needed beyond the pure incidence structure of the graph. The next section discusses a standard data structure to represent embedded graphs.

2.2.1 The Doubly-Connected Edge List

The *doubly-connected edge list* (DCEL) is a data structure to represent a plane graph in such a way that it is easy to traverse and to manipulate. In order to avoid unnecessary complications, let us discuss only connected graphs here that contain at least two vertices. It is not hard to extend the data structure to cover all plane graphs. For simplicity we also assume that we deal with a straight-line embedding and so the geometry of edges is defined by the mapping of their endpoints already. For more general embeddings, the geometric description of edges has to be stored in addition.

The main building block of a DCEL is a list of *halfedges*. Every actual edge is represented by two halfedges going in opposite direction, and these are called *twins*, see Figure 2.7. Along the boundary of each face, halfedges are oriented counterclockwise.

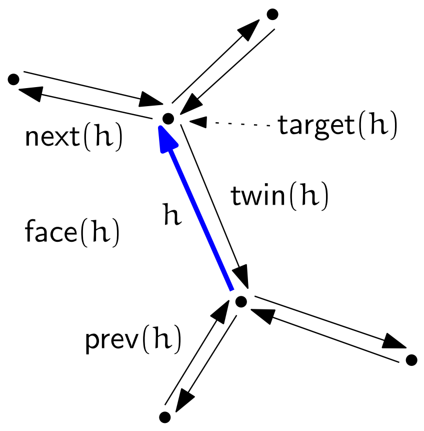


Figure 2.7: *A halfedge in a DCEL.*

A DCEL stores a list of halfedges, a list of vertices, and a list of faces. These lists are unordered but interconnected by various pointers. A vertex v stores a pointer halfedge(v) to an arbitrary halfedge originating from v. Every vertex also knows its coordinates, that is, the point point(v) it is mapped to in the represented embedding. A face f stores a pointer halfedge(f) to an arbitrary halfedge within the face. A halfedge h stores *five* pointers:

- a pointer target(h) to its target vertex,
- a pointer face(h) to the incident face,
- a pointer twin(h) to its twin halfedge,

- a pointer next(h) to the halfedge following h along the boundary of face(h), and
- a pointer prev(h) to the halfedge preceding h along the boundary of face(h).

A constant amount of information is stored for every vertex, (half-)edge, and face of the graph. Therefore the whole DCEL needs storage proportional to $|V| + |E| + |F|$, which is $O(n)$ for a plane graph with n vertices by Corollary 2.5.

This information is sufficient for most tasks. For example, traversing all edges around a face f can be done as follows:

```
$s \leftarrow \mathsf{halfedge}(f)$
$h \leftarrow s$
do
    something with $h$
    $h \leftarrow \mathsf{next}(h)$
while $h \neq s$
```

Exercise 2.13. *Give pseudocode to traverse all edges incident to a given vertex v of a DCEL.*

Exercise 2.14. *Why is the previous halfedge* prev(·) *stored explicitly and the source vertex of a halfedge is not?*

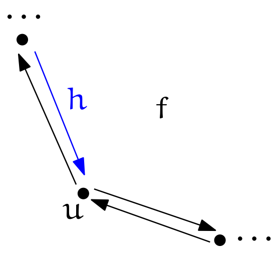
2.2.2 Manipulating a DCEL

In many applications, plane graphs appear not just as static objects but rather they evolve over the course of an algorithm. Therefore the data structure used to represent the graph must allow for efficient update operations to change it.

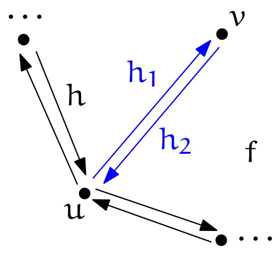
First of all, we need to be able to generate new vertices, edges, and faces, to be added to the corresponding list within the DCEL and—symmetrically—the ability to delete an existing entity. Then it should be easy to add a new vertex v to the graph within some face f. As we maintain a connected graph, we better link the new vertex to somewhere, say, to an existing vertex u. For such a connection to be possible, we require that the open line segment uv lies completely in f.

Of course, two halfedges are to be added connecting u and v. But where exactly? Given that from a vertex and from a face only some arbitrary halfedge is directly accessible, it turns out convenient to use a halfedge in the interface. Let h denote the halfedge incident to f for which $\mathsf{target}(h) = u$. Our operation then becomes (see also Figure 2.8)

add-vertex-at(v, h)
Precondition: the open line segment $\overline{\mathsf{point}(v)\mathsf{point}(u)}$, where $u := \mathsf{target}(h)$, lies completely in $f := \mathsf{face}(h)$.
Postcondition: a new vertex v has been inserted into f, connected by an edge to u.



(a) before



(b) after

Figure 2.8: *Add a new vertex connected to an existing vertex* u*.*

and it can be realized by manipulating a constant number of pointers as follows.

```
add-vertex-at($v, h$) {
    $h_1 \leftarrow$ a new halfedge
    $h_2 \leftarrow$ a new halfedge
    halfedge($v$) $\leftarrow h_2$
    twin($h_1$) $\leftarrow h_2$
    twin($h_2$) $\leftarrow h_1$
    target($h_1$) $\leftarrow v$
    target($h_2$) $\leftarrow u$
    face($h_1$) $\leftarrow f$
    face($h_2$) $\leftarrow f$
    next($h_1$) $\leftarrow h_2$
    next($h_2$) $\leftarrow$ next($h$)
    prev($h_1$) $\leftarrow h$
    prev($h_2$) $\leftarrow h_1$
    next($h$) $\leftarrow h_1$
    prev(next($h_2$)) $\leftarrow h_2$
}
```

Similarly, it should be possible to add an edge between two existing vertices u and v, provided the open line segment uv lies completely within a face f of the graph, see Figure 2.9. Since such an edge insertion splits f into two faces, the operation is called *split-face*. Again we use the halfedge h that is incident to f and for which target$(h) = u$. Our operation becomes then

split-face(h, v)
Precondition: v is incident to $f :=$ face(h) but not adjacent to $u :=$ target(h).
The open line segment $\overline{\text{point}(v)\text{point}(u)}$ lies completely in f.
Postcondition: f has been split by a new edge uv.

The implementation is slightly more complicated compared to add-vertex-at above, because the face f is destroyed and so we have to update the face information of all incident

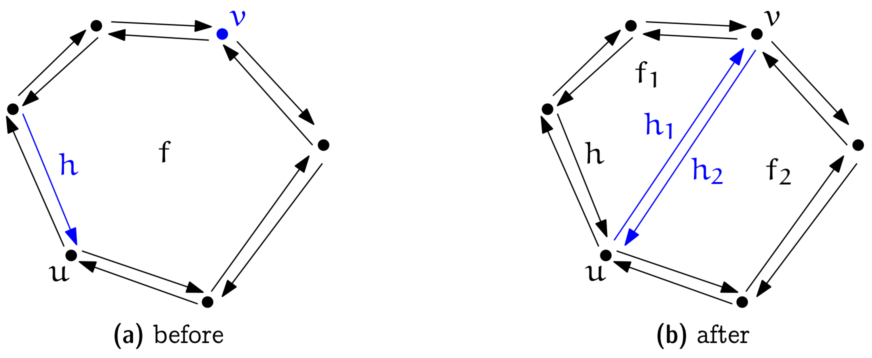


Figure 2.9: *Split a face by an edge* uv.

halfedges. In particular, this is not a constant time operation, but its time complexity is proportional to the size of f.

```
split-face($h, v$) {
    $f_1 \leftarrow$ a new face
    $f_2 \leftarrow$ a new face
    $h_1 \leftarrow$ a new halfedge
    $h_2 \leftarrow$ a new halfedge
    halfedge($f_1$) $\leftarrow h_1$
    halfedge($f_2$) $\leftarrow h_2$
    twin($h_1$) $\leftarrow h_2$
    twin($h_2$) $\leftarrow h_1$
    target($h_1$) $\leftarrow v$
    target($h_2$) $\leftarrow u$
    next($h_2$) $\leftarrow$ next($h$)
    prev(next($h_2$)) $\leftarrow h_2$
    prev($h_1$) $\leftarrow h$
    next($h$) $\leftarrow h_1$
    $i \leftarrow h_2$
    loop
        face($i$) $\leftarrow f_2$
        if target($i$) $= v$ break the loop
        $i \leftarrow$ next($i$)
    endloop
    next($h_1$) $\leftarrow$ next($i$)
    prev(next($h_1$)) $\leftarrow h_1$
    next($i$) $\leftarrow h_2$
    prev($h_2$) $\leftarrow i$
    $i \leftarrow h_1$
    do
        face($i$) $\leftarrow f_1$
        $i \leftarrow$ next$(i)$
    until target$(i) = u$
    delete the face $f$
}
```

In a similar fashion one can realize the inverse operation join-face(h) that removes the edge (represented by the halfedge) h, thereby joining the faces face(h) and face(twin(h)).

It is easy to see that every connected plane graph on at least two vertices can be constructed using the operations add-vertex-at and split-face, starting from an embedding of K_2 (two vertices connected by an edge).

Exercise 2.15. *Give pseudocode for the operation join-face(h). Also specify preconditions, if needed.*

Exercise 2.16. *Give pseudocode for the operation split-edge(h), that splits the edge (represented by the halfedge) h into two by a new vertex w, see Figure 2.10.*

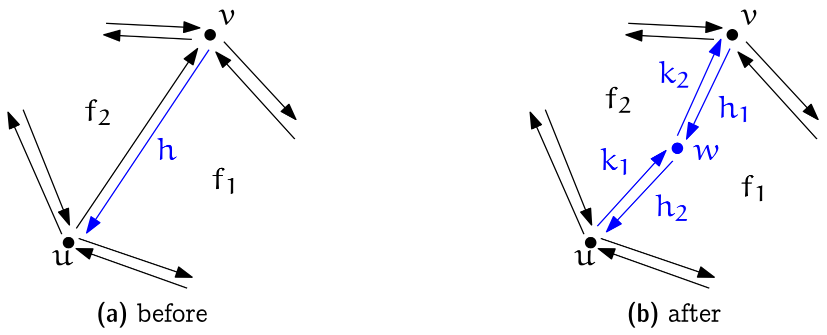


Figure 2.10: *Split an edge by a new vertex.*

2.2.3 Graphs with Unbounded Edges

In some cases it is convenient to consider plane graphs, in which some edges are not mapped to a line segment but to an unbounded curve, such as a ray. This setting is not really much different from the one we studied before, except that one vertex is placed "at infinity". One way to think of it is in terms of *stereographic projection* (see the proof of Theorem 2.2). The further away a point in $\mathbb{R}^2$ is from the origin, the closer its image on the sphere S gets to the north pole n of S. But there is no way to reach n except in the limit. Therefore, we can imagine drawing the graph on S instead of in $\mathbb{R}^2$ and putting the "infinite vertex" at n.

All this is just for the sake of a proper geometric interpretation. As far as a DCEL representation of such a graph is concerned, there is no need to consider spheres or, in fact, anything beyond what we have discussed before. The only difference to the case with all finite edges is that there is this special infinite vertex, which does not have any

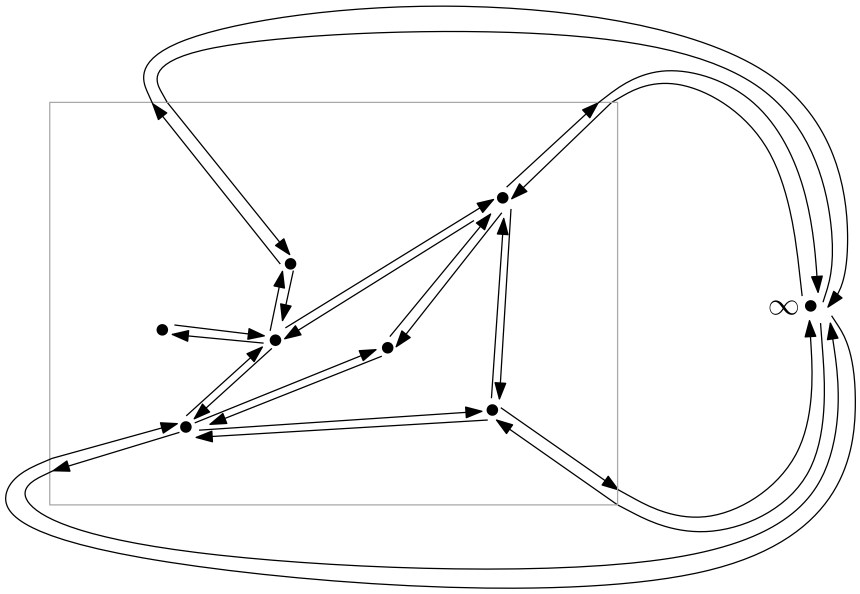


Figure 2.11: *A DCEL with unbounded edges. Usually, we will not show the infinite vertex and draw all edges as straight-line segments. This yields a geometric drawing, like the one within the gray box.*

point/coordinates associated to it. But other than that, the infinite vertex is treated in exactly the same way as the finite vertices: it has in– and outgoing halfedges along which the unbounded faces can be traversed (Figure 2.11).

Remarks. It is actually not so easy to point exactly to where the DCEL data structure originates from. Often Muller and Preparata [23] are credited, but while they use the term DCEL, the data structure they describe is different from what we discussed above and from what people usually consider a DCEL nowadays. Overall, there are a large number of variants of this data structure, which appear under the names *winged edge* data structure [3], *halfedge* data structure [31], or *quad-edge* data structure [15]. Kettner [19] provides a comparison of all these and some additional references.

2.2.4 Combinatorial Embeddings

The DCEL data structure discussed in the previous section provides a fully fleshed-out representation of what is called a *combinatorial embedding*. From a mathematical point of view this can be regarded an equivalence relation on embeddings: Two embeddings are equivalent if their face boundaries—regarded as circular sequences of edges (or vertices) in counterclockwise order—are the same (as sets) up to a global change of orientation (reversing the order of all sequences simultaneously). For instance, the faces of the plane graphs shown in Figure 2.12a are (each face is described as a circular sequence of vertices)

(a) : $\{(1,2,3),(1,3,6,4,5,4),(1,4,6,3,2)\}$,
(b) : $\{(1,2,3,6,4,5,4),(1,3,2),(1,4,6,3)\}$, and
(c) : $\{(1,4,5,4,6,3),(1,3,2),(1,2,3,6,4)\}$.

Note that a vertex can appear several times along the boundary of a face (if it is a cut-vertex). Clearly (b) is not equivalent to (a) nor (c), because it is the only graph that contains a face bounded by seven vertices. However, (a) and (c) turn out to be equivalent: after reverting orientations f_1 takes the role of h_2, f_2 takes the role of h_1, and f_3 takes the role of h_3.

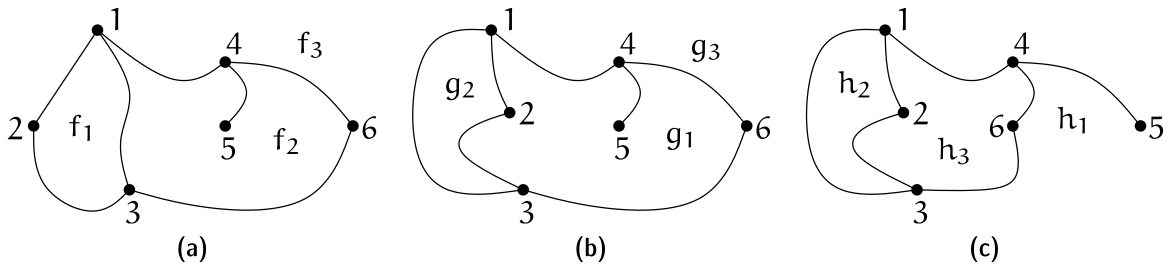


Figure 2.12: *Equivalent embeddings?*

Exercise 2.17. *Let* G *be a planar graph with vertex set* $\{1,\ldots,9\}$. *Try to find an embedding corresponding to the following list of circular sequences of faces:*

(a) $\{(1,4,5,6,3),(1,3,6,2),(1,2,6,7,8,9,7,6,5),(7,9,8),(1,5,4)\}$

(b) $\{(1,4,5,6,3),(1,3,6,2),(1,2,6,7,8,9,7,6,5),(7,9,8),(1,4,5)\}$

In a dual interpretation one can just as well define equivalence in terms of the cyclic order of neighbors around all vertices. In this form, a compact way to describe a combinatorial embedding is as a so-called *rotation system* that consists of a permutation π and an involution ρ, both of which are defined on the set of halfedges (in this context often called *darts* or *flags*) of the embedding. The orbits of π correspond to the vertices, as they iterate over the incident halfedges. The involution ρ maps each halfedge to its twin.

Many people prefer this dual view, because one does not have to discuss the issue of vertices or edges that appear several times on the boundary of a face. The following lemma shows that such an issue does not arise when dealing with biconnected graphs.

Lemma 2.18. *In a biconnected plane graph every face is bounded by a cycle.*

We leave the proof as an exercise. Intuitively the statement is probably clear. But we believe it is instructive to think about how to make a formal argument. An easy consequence is the following corollary, whose proof we also leave as an exercise.

Corollary 2.19. *In a 3-connected plane graph the neighbors of a vertex lie on a cycle.*

Note that the statement does not read "form a cycle" but rather "lie on a cycle".

Exercise 2.20. *Prove Lemma 2.18 and Corollary 2.19.*

2.3 Unique Embeddings

We have seen in Lemma 2.18 that all faces in biconnected plane graphs are bounded by cycles. Conversely one might wonder which cycles of a planar graph G bound a face in *some* plane embedding of G. Such a cycle is called a *facial cycle* (Figure 2.13).

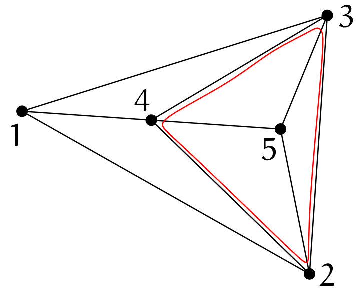


Figure 2.13: *The cycle* $(1,2,3)$ *is facial and we can show that* $(2,3,4)$ *is not.*

In fact, we will look at a slightly different class of cycles, namely those that bound a face in *every* plane embedding of G. The lemma below provides a complete characterization of those cycles. In order to state it, let us introduce a bit more terminology. A *chord* of a cycle C in a graph G is an edge that connects two vertices of C but is not an edge of C. A cycle C in a graph G is an *induced cycle*, if $C = G[V(C)]$, that is, C does not have any chord in G.

Lemma 2.21. *Let* C *be a cycle in a planar graph* G *such that* $G \neq C$ *and* G *is not* C *plus a single chord of* C*. Then* C *bounds a face in every plane embedding of* G *if and only if* C *is an induced cycle and it is not separating (i.e.,* $G \setminus C$ *is connected).*

Proof. "$\Leftarrow$": Consider any plane embedding Γ of G. As $G \setminus C$ is connected, by the Jordan Curve Theorem it is contained either in the interior of C or in the exterior of C in Γ. In either case, the other component of the plane is bounded by C, because there are no edges among the vertices of C.

"$\Rightarrow$": Using contraposition, suppose that C is not induced or $G \setminus C$ is disconnected. We have to show that there exists a plane embedding of G in which C does not bound a face.

If C is not induced, then there is a chord c of C in G. As $G \neq C \cup c$, either G has a vertex v that is not in C or G contains another chord $d \neq c$ of C. In either case, consider any plane embedding Γ of G in which C bounds a face. (If such an embedding does not exist, there is nothing to show.) We can modify Γ by drawing the chord c in the face bounded by C to obtain an embedding Γ' of G in which C does not bound a face: one of the two regions bounded by C according to the Jordan Curve Theorem contains c and the other contains either the vertex v or the other chord d.

If $G \setminus C$ contains two components A and B, then consider a plane embedding Γ of G. If C is not a face in Γ, there is nothing to show. Hence suppose that C is a face of Γ (Figure 2.14a). From Γ we obtain induced plane embeddings Γ_A of $G \setminus B = A \cup C$ and Γ_B of $G \setminus A = B \cup C$. Using Theorem 2.2 we may suppose that C bounds the outer face in Γ_A and it does not bound the outer face in Γ_B. Then we can glue both embeddings at C, that is, extend Γ_B to an embedding of G by adding Γ_A within the face bounded by C (Figure 2.14b). The resulting embedding is a plane drawing of G in which C does not bound a face.

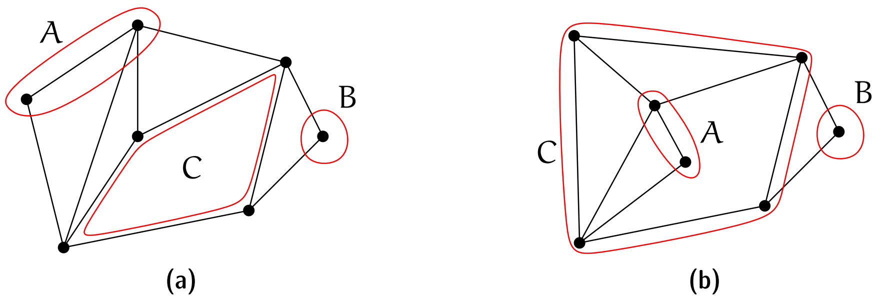


Figure 2.14: *Construct a plane embedding of G in which C does not bound a face.*

Finally, consider the case that $G \setminus C = \emptyset$ (which is not a connected graph according to our definition). As we considered above the case that C is not an induced cycle, the only remaining case is $G = C$, which is excluded explicitly. □

For both special cases for G that are excluded in Lemma 2.21 it is easy to see that all cycles in G bound a face in every plane embedding. This completes the characterization. Also observe that in these special cases G is not 3-connected.

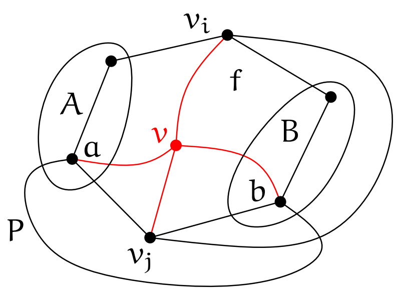
Corollary 2.22. *A cycle C of a 3-connected planar graph G bounds a face in every plane embedding of G if and only if C is an induced cycle and it is not separating.* □

The following theorem tells us that for a wide range of graphs we have little choice as far as a plane embedding is concerned, at least from a combinatorial point of view. Geometrically, there is still a lot of freedom, though.

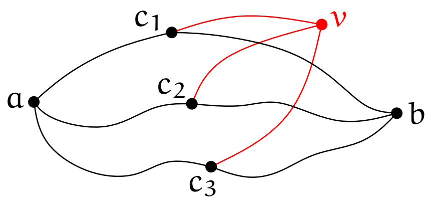
Theorem 2.23 (Whitney [32]). *A 3-connected planar graph has a unique combinatorial plane embedding (up to equivalence).*

Proof. Let G be a 3-connected planar graph and suppose there exist two embeddings Φ_1 and Φ_2 of G that are not equivalent. That is, there is a cycle $C = (v_1, \ldots, v_k)$, $k \geqslant 3$, in G that bounds a face in, say, Φ_1 but C does not bound a face in Φ_2. By Corollary 2.22 such a cycle has a chord or it is separating. We consider both options.

Case 1: C has a chord $\{v_i, v_j\}$, with $j \geqslant i+2$. Denote $A = \{v_x : i < x < j\}$ and $B = \{v_x : x < i \vee j < x\}$ and observe that both A and B are nonempty (because $\{v_i, v_j\}$ is a chord and so v_i and v_j are not adjacent in C). Given that G is 3-connected, there is at least one path P from A to B that does not use either of v_i or v_j. Let a denote the last vertex of P that is in A, and let b denote the first vertex of P that is in B. As C bounds a face f in Φ_1, we can add a new vertex v inside the face bounded by C and connect v by four pairwise internally disjoint curves to each of v_i, v_j, a, and b. The result is a plane graph $G' \supset G$ that contains a subdivision of K_5 with branch vertices v, v_i, v_j, a, and b. By Kuratowski's Theorem (Theorem 2.9) this contradicts the planarity of G'.



(a) Case 1.



(b) Case 2.

Figure 2.15: *Illustration of the two cases in Theorem 2.23.*

Case 2: C is induced and separating. Then $G \setminus C$ contains two distinct components A and B. (We have $V(G) \neq V(C)$ and, in particular, $G \setminus C \neq \emptyset$ because C is induced and G is 3-connected.) Consider now the embedding Φ_1 in which C bounds a face, without loss of generality (Theorem 2.2) a bounded face f. Hence both A and B are embedded in the exterior of f.

Choose vertices $a \in A$ and $b \in B$ arbitrarily. As G is 3-connected, by Menger's Theorem (Theorem 1.2), there are at least three pairwise internally vertex-disjoint paths from a to b. Fix three such paths $\alpha_1, \alpha_2, \alpha_3$ and denote by c_i the first point of α_i that is on C, for $1 \leqslant i \leqslant 3$. Note that c_1, c_2, c_3 are well defined, because C separates A and B, and they are pairwise distinct. Therefore, $\{a, b\}$ and $\{c_1, c_2, c_3\}$ are branch vertices of a $K_{2,3}$ subdivision in G. We can add a new vertex v inside the face bounded by C and connect v by three pairwise internally disjoint curves to each of c_1, c_2, and c_3. The result is a plane graph $G' \supset G$ that contains a $K_{3,3}$ subdivision. By Kuratowski's Theorem (Theorem 2.9) this contradicts the planarity of G'.

In both cases we arrived at a contradiction and so there does not exist such a cycle C. Thus Φ_1 and Φ_2 are equivalent. □

Whitney's Theorem does not provide a characterization of unique embeddability, because there are both biconnected graphs that have a unique plane embedding (such as cycles) and biconnected graphs that admit several nonequivalent plane embeddings (for instance, a triangulated pentagon).

2.4 Triangulating a Plane Graph

We like to study worst case scenarios not so much to dwell on "how bad things could get" but rather—phrased positively—because worst case examples provide universal bounds of the form "things are always at least that good". Most questions related to embeddings get harder the more edges the graph has because every additional edge needs to avoid potential crossings with other edges. Therefore, let us study the class of maximal planar graphs. A graph is *maximal planar* if no edge can be added so that the resulting graph is still planar. Corollary 2.5 tells us that a (maximal) planar graph on n vertices cannot have more than $3n-6$ edges. Yet we would like to learn a bit more about how these graphs look like.

Lemma 2.24. *A maximal planar graph on $n \geqslant 3$ vertices is biconnected.*

Proof. Consider a maximal planar graph $G = (V, E)$. We may suppose that G is connected because adding an edge between two distinct components of a planar graph maintains planarity. Therefore, if G is not biconnected, then it has a cut-vertex v. Take a plane drawing Γ of G. As $G \setminus v$ is disconnected, removal of v also splits $N_G(v)$ into at least two components. Therefore, there are two vertices $a, b \in N_G(v)$ that are adjacent in the circular order of vertices around v in Γ and are in different components of $G \setminus v$. In particular, $\{a, b\} \notin E$ and we can add this edge to G (routing it very close to the path (a, v, b) in Γ) without violating planarity. This is in contradiction to G being maximal planar and so G is biconnected. □

Lemma 2.25. *In a maximal planar graph on $n \geqslant 3$ vertices, all faces are topological triangles, that is, every face is bounded by exactly three edges.*

Proof. Consider a maximal planar graph $G = (V, E)$ and a plane drawing Γ of G. By Lemma 2.24 we know that G is biconnected and so by Lemma 2.18 every face of Γ is bounded by a cycle. Suppose that there is a face f in Γ that is bounded by a cycle $v_0, \ldots, v_{k-1}$ of $k \geqslant 4$ vertices. We claim that at least one of the edges $\{v_0, v_2\}$ or $\{v_1, v_3\}$ is not present in G.

Suppose to the contrary that $\{\{v_0, v_2\}, \{v_1, v_3\}\} \subseteq E$. Then we can add a new vertex v' in the interior of f and connect v' inside f to all of v_0, v_1, v_2, v_3 by an edge (curve) without introducing a crossing. In other words, given that G is planar, also the graph $G' = (V \cup \{v'\}, E \cup \{\{v_i, v'\} : i \in \{0, 1, 2, 3\}\})$ is planar. However, v_0, v_1, v_2, v_3, v' are branch vertices of a K_5 subdivision in G': v' is connected to all other vertices within f, along the boundary ∂f of f each vertex v_i is connected to both $v_{(i-1) \bmod 4}$ and $v_{(i+1) \bmod 4}$ and the missing two connections are provided by the edges $\{v_0, v_2\}$ and $\{v_1, v_3\}$ (Figure 2.16a). By Kuratowski's Theorem this is in contradiction to G' being planar. Therefore, one of the edges $\{v_0, v_2\}$ or $\{v_1, v_3\}$ is not present in G, as claimed.

So suppose without loss of generality that $\{v_1, v_3\} \notin E$. But then we can add this edge (curve) within f to Γ without introducing a crossing (Figure 2.16b). It follows that the edge $\{v_1, v_3\}$ can be added to G without sacrificing planarity, which is in contradiction

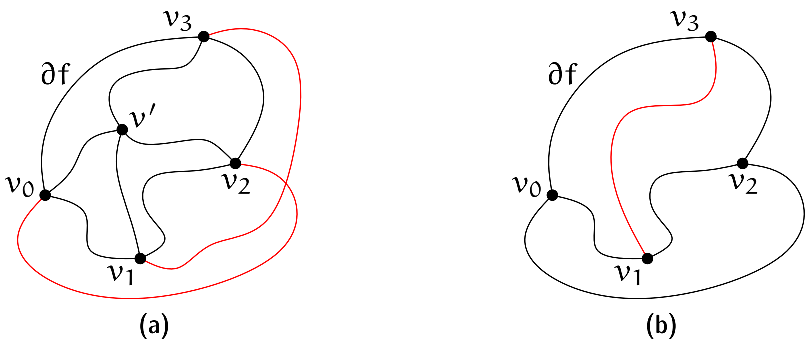


Figure 2.16: *Every face of a maximal planar graph is a topological triangle.*

to G being maximal planar. Therefore, there is no such face f bounded by four or more vertices. □

Exercise 2.26. *Does every minimal nonplanar graph* G *(that is, every nonplanar graph* G *whose proper subgraphs are all planar) contain an edge* e *such that* $G \setminus e$ *is maximal planar?*

Many questions for graphs are formulated for connected graphs only because it is easy to add edges to a disconnected graph to make it connected. For similar reasons many questions about planar embeddings are formulated for maximal planar graphs only because it is easy to add edges to a planar graph so as to make it maximal planar. Well, this last statement is not entirely obvious. Let us look at it in more detail.

An *augmentation* of a given planar graph $G = (V, E)$ to a maximal planar graph $G' = (V, E)$ with $E' \supseteq E$ is also called a *topological triangulation*. The proof of Lemma 2.25 already contains the basic idea for an algorithm to topologically triangulate a plane graph.

Theorem 2.27. *For a given connected plane graph* $G = (V, E)$ *on* n *vertices one can compute in* $O(n)$ *time and space a maximal plane graph* $G' = (V, E')$ *with* $E \subseteq E'$*.*

Proof. Suppose, for instance, that G is represented as a DCEL[2], from which one can easily extract the face boundaries. If some vertex v appears several times along the boundary of a single face, it is a cut-vertex. We fix this by adding an edge between the two neighbors of all but the first occurrence of v. This can easily be done in linear time by maintaining a counter for each vertex on the face boundary. The total number of edges and vertices along the boundary of all faces is proportional to the number of edges in G, which by Corollary 2.5 is linear. Hence we may suppose that all faces of G are bounded by a cycle.

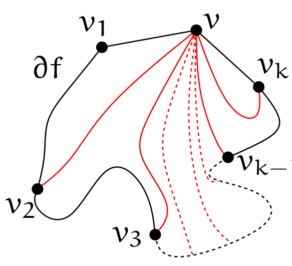
For every face f that is bounded by more than three vertices, select a vertex v_f on its boundary and store with every vertex all faces that select it. Then process every vertex

[2]If you wonder how the—possibly complicated—curves that correspond to edges are represented: they do not need to be, because here we need a representation of the combinatorial embedding only.

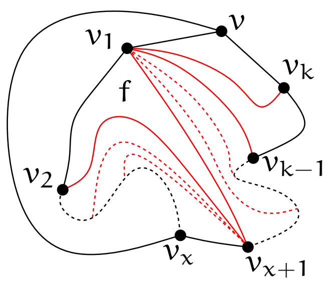
v as follows: First mark all neighbors of v in G. Then process all faces that selected v. For each such face f with $v_f = v$ iterate over the boundary $\partial f = (v, v_1, \ldots, v_k)$, where $k \geqslant 3$, of f to test whether there is any marked vertex other than the two neighbors v_1 and v_k of v along ∂f.

If there is no such vertex, we can safely triangulate f using a star from v, that is, by adding the edges $\{v, v_i\}$, for $i \in \{2, \ldots, k-1\}$ (Figure 2.17a).

Otherwise, let v_x be the first marked vertex in the sequence $v_2, \ldots, v_{k-1}$. The edge $\{v, v_x\}$ that is embedded as a curve in the exterior of f prevents any vertex from $v_1, \ldots, v_{x-1}$ from being connected by an edge in G to any vertex from $v_{x+1}, \ldots, v_k$. (This is exactly the argument that we made in the proof of Lemma 2.25 above for the edges $\{v_0, v_2\}$ and $\{v_1, v_3\}$, see Figure 2.16a.) In particular, we can safely triangulate f using a bi-star from v_1 and v_{x+1}, that is, by adding the edges $\{v_1, v_i\}$, for $i \in \{x+1, \ldots, k\}$, and $\{v_j, v_{x+1}\}$, for $j \in \{2, \ldots, x-1\}$ (Figure 2.17b).



(a) Case 1: v does not have any neighbor on ∂f other than v_1 and v_k.



(b) Case 2: v has a neighbor v_x on ∂f other than v_1 and v_k.

Figure 2.17: *Topologically triangulating a plane graph.*

Finally, conclude the processing of v by removing all marks on its neighbors.

Regarding the runtime bound, note that every face is traversed a constant number of times. In this way, each edge is touched a constant number of times, which by Corollary 2.5 uses linear time overall. Similarly, marking the neighbors of a chosen vertex is done at most twice (mark und unmark) per vertex. Therefore, the overall time needed can be bounded by $\sum_{v \in V} \deg_G(v) = 2|E| = O(n)$ by the Handshaking Lemma and Corollary 2.5. □

Theorem 2.28. *A maximal planar graph on $n \geqslant 4$ vertices is 3-connected.*

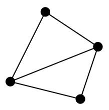
Exercise 2.29. *Prove Theorem 2.28.*

Using any of the standard planarity testing algorithms we can obtain a combinatorial embedding of a planar graph in linear time. Together with Theorem 2.27 this yields the following

Corollary 2.30. *For a given planar graph $G = (V, E)$ on n vertices one can compute in $O(n)$ time and space a maximal planar graph $G' = (V, E')$ with $E \subseteq E'$.* □

The results discussed in this section can serve as a tool to fix the combinatorial embedding for a given graph G: augment G using Theorem 2.27 to a maximal planar graph G', whose combinatorial embedding is unique by Theorem 2.23.

Being maximal planar is a property of an abstract graph. In contrast, a geometric graph to which no straight-line edge can be added without introducing a crossing is called a *triangulation*. Not every triangulation is maximal planar, as the example depicted to the right shows.



It is also possible to triangulate a geometric graph in linear time. But this problem is much more involved. Triangulating a single face of a geometric graph amounts to what is called "triangulating a simple polygon". This can be done in near-linear[3] time using standard techniques, and in linear time using Chazelle's famous algorithm, whose description spans a fourty pages paper [8].

Exercise 2.31. *We discussed the DCEL structure to represent plane graphs in Section 2.2.1. An alternative way to represent an embedding of a maximal planar graph is the following: For each triangle, store references to its three vertices and to its three neighboring triangles. Compare both approaches. Discuss different scenarios where you would prefer one over the other. In particular, analyze the space requirements of both.*

Connectivity serves as an important indicator for properties of planar graphs. Another example is the following famous theorem of Tutte that provides a sufficient condition for Hamiltonicity. Its proof is beyond the scope of our lecture.

Theorem 2.32 (Tutte [28]). *Every 4-connected planar graph is Hamiltonian.*

Moreover, for a given 4-connected planar graph a Hamiltonian cycle can also be computed in linear time [9].

2.5 Compact Straight-Line Drawings

As a next step we consider plane embeddings in the geometric setting, where every edge is drawn as a straight-line segment. A classical theorem of Wagner and Fáry states that this is not a restriction in terms of plane embeddability.

Theorem 2.33 (Fáry [12], Wagner [29]). *Every planar graph has a plane straight-line embedding.*

This statement is quite surprising, considering how much more freedom arbitrarily complex Jordan arcs allow compared to line segments, which are completely determined by their endpoints. In order to further increase the level of appreciation, let us note that a similar "straightening" is not possible, when fixing the point set on which the vertices are to be embedded: For a given planar graph $G = (V, E)$ on n vertices and a given

[3] $O(n \log n)$ or—using more elaborate tools—$O(n \log^* n)$ time

set $P \subset \mathbb{R}^2$ of n points, one can always find a plane embedding of G that maps V to P [24]. However, this is not possible in general with a plane *straight-line* embedding. For instance, K_4 does not admit a plane straight-line embedding on a set of points that form a convex quadrilateral, such as a rectangle. In fact, it is NP-hard to decide whether a given planar graph admits a plane straight-line embedding on a given point set [6].

Exercise 2.34. **a)** *Show that for every natural number $n \geqslant 4$ there exist a planar graph G on n vertices and a set $P \subset \mathbb{R}^2$ of n points in general position (no three points are collinear) so that G does not admit a plane straight-line embedding on P.*

b) *Show that for every natural number $n \geqslant 6$ there exist a planar graph G on n vertices and a set $P \subset \mathbb{R}^2$ of n points so that (1) P is in general position (no three points are collinear); (2) P has a triangular convex hull (that is, there are three points in P that form a triangle that contains all other points from P); and (3) G does not admit a plane straight-line embedding on P.*

Exercise 2.35. *Show that for every set $P \subset \mathbb{R}^2$ of $n \geqslant 3$ in general position (no three points are collinear) the cycle C_n on n vertices admits a plane straight-line embedding on P.*

Although Fáry-Wagner's theorem has a nice inductive proof, we will not discuss it here. Instead we will prove a stronger statement that implies Theorem 2.33.

A very nice property of straight-line embeddings is that they are easy to represent: We need to store points/coordinates for the vertices only. From an algorithmic and complexity point of view the space needed by such a representation is important, because it appears in the input and output size of algorithms that work on embedded graphs. While the Fáry-Wagner Theorem guarantees the existence of a plane straight-line embedding for every planar graph, it does not provide bounds on the size of the coordinates used in the representation. But the following strengthening provides such bounds, by describing an algorithm that embeds (without crossings) a given planar graph on a linear size integer grid.

Theorem 2.36 (de Fraysseix, Pach, Pollack [14]). *Every planar graph on $n \geqslant 3$ vertices has a plane straight-line drawing on the $(2n-3) \times (n-1)$ integer grid.*

2.5.1 Canonical Orderings

The key concept behind the algorithm is the notion of a canonical ordering, which is a vertex order that allows to construct a plane drawing in a natural (hence canonical) way. Reading it backwards one may think of a shelling or peeling order that destructs the graph vertex by vertex from the outside. A canonical ordering also provides a succinct representation for the combinatorial embedding.

Definition 2.37. *A plane graph is* **internally triangulated** *if it is biconnected and every bounded face is a (topological) triangle. Let G be an internally triangulated plane graph and $C_\circ(G)$ its outer cycle. A permutation $\pi = (v_1, v_2, \ldots, v_n)$ of $V(G)$ is a* **canonical ordering** *for G, if*

(CO1) G_k *is internally triangulated, for all* $k \in \{3, \dots, n\}$*;*

(CO2) v_1v_2 *is on the outer cycle* $C_\circ(G_k)$ *of* G_k*, for all* $k \in \{3, \dots, n\}$*;*

(CO3) v_{k+1} *is located in the outer face of* G_k *and its neighbors (in* G_k*) appear consecutively along* $C_\circ(G_k)$*, for all* $k \in \{3, \dots, n-1\}$*;*

where G_k *is the subgraph of* G *induced by* $v_1, \dots, v_k$*.*

Figure 2.18 shows an example. Note that there are permutations that do not correspond to a canonical order: for instance, when choosing the vertex 4 as the next vertex to be removed in Figure 2.18b, the resulting graph $G'_7 = G[\{1, 2, 3, 5, 6, 7, 8\}]$ is not biconnected (because 1 is a cut-vertex).

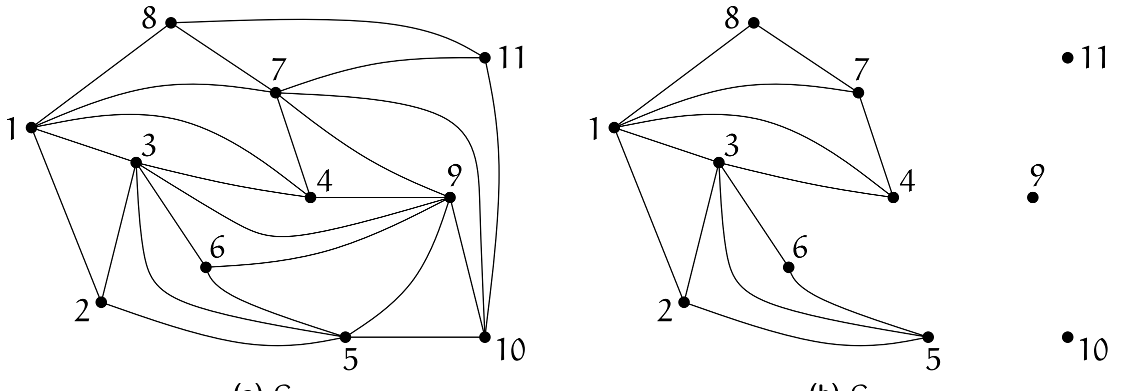


Figure 2.18: *An internally triangulated plane graph with a canonical ordering.*

Theorem 2.38. *For every internally triangulated plane graph* G *and every edge* $\{v_1, v_2\}$ *on its outer face, there exists a canonical ordering for* G *that starts with* v_1, v_2*. Moreover, such an ordering can be computed in linear time.*

Proof. Induction on n, the number of vertices. For a triangle, any order suffices and so the statement holds. Hence consider an internally triangulated plane graph $G = (V, E)$ on $n \geqslant 4$ vertices. We claim that it is enough to select a vertex $v_n \notin \{v_1, v_2\}$ on $C_\circ(G)$ that is not incident to a chord of $C_\circ(G)$ and then apply induction on $G \setminus \{v_n\}$.

We will show later that such a vertex v_n always exists. First let us prove the claim. We need to argue that if v_n is selected as described

(i) the plane graph $G_{n-1} := G \setminus \{v_n\}$ is internally triangulated,

(ii) the given edge $\{v_1, v_2\}$ is on the outer face $C_\circ(G_{n-1})$ of G_{n-1}, and

(iii) we can extend the inductively obtained canonical ordering for G_{n-1} with v_n to obtain a canonical ordering for G.

Property (ii) is an immediate consequence of $v_n \notin \{v_1, v_2\}$.

Regarding (iii) note that (CO1) and (CO2) for $k = n$ hold by assumption. For (CO3) recall that G is plane and $v_n \in C_\circ(G)$. Hence all neighbors of v_n in G must appear on $C_\circ(G_{n-1})$. Consider the circular sequence of neighbors around v_n in G and break it into a linear sequence $u_1, \ldots, u_m$, for some $m \geqslant 2$, that starts and ends with the neighbors of v_n in $C_\circ(G)$. As G is internally triangulated, each of the bounded faces spanned by v_n, u_i, u_{i+1}, for $i \in \{1, \ldots, m-1\}$, is a triangle and hence $\{u_i, u_{i+1}\} \in E$. This implies (CO3).

It remains to show (i). The way G_{n-1} is obtained from G, every bounded face f of G_{n-1} also appears as a bounded face of G. As G is internally triangulated, f is a triangle. It remains to show that G_{n-1} is biconnected. The outer cycle $C_\circ(G_{n-1})$ of G_{n-1} is obtained from $C_\circ(G)$ by removing v_n and replacing it with the (possibly empty) sequence $u_2, \ldots, u_{m-1}$. As v_n is not incident to a chord of $C_\circ(G)$ (and so neither of $u_2, \ldots, u_{m-1}$ appeared along $C_\circ(G)$ already), the resulting sequence forms a cycle, indeed. Add a new vertex v in the outer face of G_{n-1} and connect v to every vertex of $C_\circ(G_{n-1})$ to obtain a maximal planar graph $H \supset G_{n-1}$. By Theorem 2.28 H is 3-connected and so G_{n-1} is biconnected, as desired. This also completes the proof of the claim.

Next let us show that we can always find a vertex $v_n \notin \{v_1, v_2\}$ on $C_\circ(G)$ that is not incident to a chord of $C_\circ(G)$. If $C_\circ(G)$ does not have any chord, this is obvious, because every cycle has at least three vertices, one of which is neither v_1 nor v_2. So suppose that $C_\circ(G)$ has a chord c. The endpoints of c split $C_\circ(G)$ into two paths, one of which does not have v_1 nor v_2 as an internal vertex. Among all chords of $C_\circ(G)$ select c such that this path has minimal length. (It has always at least two edges, because there is always at least one vertex "behind" a chord.) Then by definition of c this path is an induced path in G and none of its (at least one) interior vertices is incident to a chord of $C_\circ(G)$, because such a chord would cross c. So we can select v_n from these vertices. By the way the path is selected with respect to c, this procedure does not select v_1 nor v_2.

Regarding the runtime bound, we maintain the following information for each vertex v: whether it has been chosen already, whether it is on the outer face of the current graph, and the number of incident chords with respect to the current outer cycle. Given a combinatorial embedding of G, it is straighforward to initialize this information in linear time. (Every edge is considered at most twice, once for each endpoint on the outer face.) We also maintain an unsorted list of the *eligible* vertices, that is, those vertices that are on the outer face and not incident to any chord. This list is straightforward to maintain: Whenever a vertex information is updated, check before and after the update whether it is eligible and correspondingly add it to or remove it from the list of eligible vertices.

When removing a vertex, there are two cases: Either v_n has two neighbors u_1 and u_2 only (Figure 2.19a), in which case the edge u_1u_2 ceases to be a chord. Thus, the chord count for u_1 and u_2 has to be decremented by one. Otherwise, there are $m \geqslant 3$ neighbors $u_1, \ldots, n_m$ (Figure 2.19b) and (1) all vertices $u_2, \ldots, u_{m-1}$ are new on the outer cycle, and (2) every edge incident to u_i, for $i \in \{2, \ldots, m-1\}$, and some other vertex on the outer cycle other than u_{i-1} or u_{i+1} is a new chord (and the corresponding counters at the endpoints have to by incremented by one).

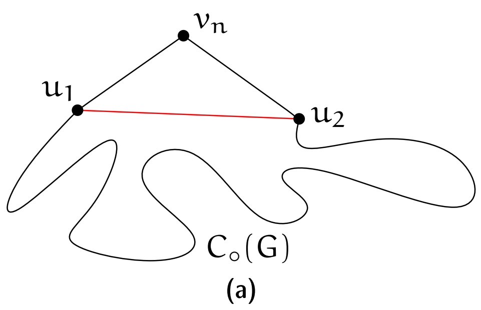


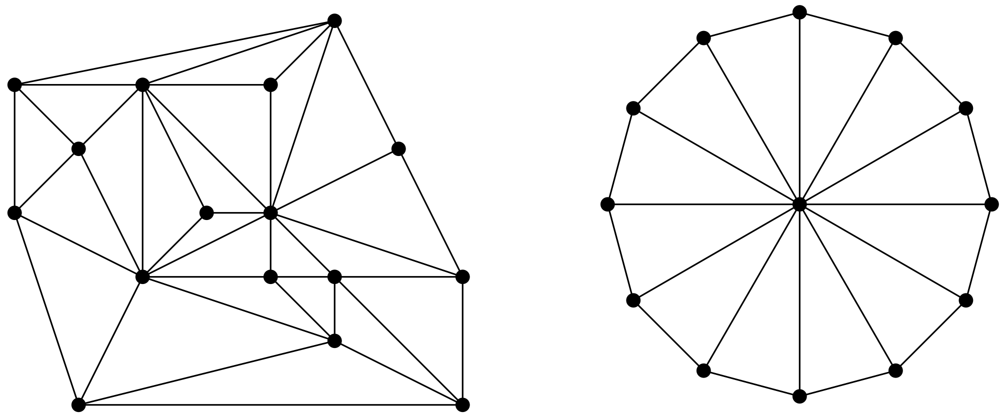
Figure 2.19: *Processing a vertex when computing a canonical ordering.*

During the course of the algorithm every vertex appears once as a new vertex on the outer face. At this point all incident edges are examined. Overall, every edge is inspected at most twice—once for each endpoint—which takes linear time by Corollary 2.5. □

Using one of the linear time planarity testing algorithms, we can obtain a combinatorial embedding for a given maximal planar graph G. As every maximal plane graph is internally triangulated, we can then use Theorem 2.38 to provide us with a canonical ordering for G, in overall linear time.

Corollary 2.39. *Every maximal planar graph admits a canonical ordering. Moreover, such an ordering can be computed in linear time.* □

Exercise 2.40. *(a) Compute a canonical ordering for the following internally triangulated plane graphs:*



(b) Give a family of internally triangulated plane graphs G_n on $n = 2k$ vertices with at least $k!$ canonical orderings.

Exercise 2.41. *(a) Describe a plane graph* G *with* n *vertices that can be embedded (while preserving the outer face) on a grid of size* $(2n/3-1) \times (2n/3-1)$ *but not on a smaller grid.*

(b) Can you draw G *on a smaller grid if you are allowed to change the embedding?*

As simple as they may appear, canonical orderings are a powerful and versatile tool to work with plane graphs. As an example, consider the following partitioning theorem.

Theorem 2.42 (Schnyder [26]). *For every maximal planar graph* G *on at least three vertices and every fixed face* f *of* G*, the multigraph obtained from* G *by doubling the (three) edges of* f *can be partitioned into three spanning trees.*

Exercise 2.43. *Prove Theorem 2.42.* Hint: *Take a canonical ordering and build one tree by taking for every vertex* v_k *the edge to its first neighbor on the outer cycle* $C_\circ(G_{k-1})$.

Of a similar flavor is the following open problem, for which only partial answers for specific types of point sets are known [1, 4].

Problem 2.44 (In memoriam Ferran Hurtado (1951–2014)).
Can every complete geometric graph on $n = 2k$ vertices (the complete straight line graph on a set of n points in general position) be partitioned into k plane spanning trees?

2.5.2 The Shift-Algorithm

Let $(v_1, \ldots, v_n)$ be a canonical ordering. The general plan is to construct an embedding by inserting vertices in this order, starting from the triangle $P(v_1) = (0,0)$, $P(v_3) = (1,1)$, $P(v_2) = (2,0)$; see Figure 2.20.

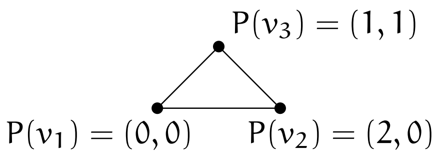


Figure 2.20: *Initialization of the shift algorithm.*

At each step, some vertices will be shifted to the right, making room for the edges to the freshly inserted vertex. For each vertex v_i already embedded, maintain a set $L(v_i)$ of vertices that move rigidly together with v_i. Initially $L(v_i) = \{v_i\}$, for $1 \leqslant i \leqslant 3$.

Ensure that the following invariants hold after Step k (that is, after v_k has been inserted):

(i) $P(v_1) = (0,0)$, $P(v_2) = (2k-4, 0)$;

(ii) The x-coordinates of the points on $C_\circ(G_k) = (w_1, \dots, w_t)$, where $w_1 = v_1$ and $w_t = v_2$, are strictly increasing (in this order)[4];

(iii) each edge of $C_\circ(G_k)$ is drawn as a straight-line segment with slope ± 1.

Clearly these invariants hold for G_3, embedded as described above. Invariant (i) implies that after Step n we have $P(v_2) = (2n-4, 0)$, while (iii) implies that the Manhattan distance[5] between any two points on $C_\circ(G_k)$ is even.

Idea: put v_{k+1} at $\mu(w_p, w_q)$, where $w_p, \dots, w_q$ are its neighbors on $C_\circ(G_k)$ (recall that they appear consecutively along $C_\circ(G_k)$ by definition of a canonical ordering), where

$$\mu((x_p, y_p), (x_q, y_q)) = \frac{1}{2}(x_p - y_p + x_q + y_q, -x_p + y_p + x_q + y_q)$$

is the point of intersection between the line $\ell_1 : y = x - x_p + y_p$ of slope 1 through $w_p = (x_p, y_p)$ and the line $\ell_2 : y = x_q - x + y_q$ of slope -1 through $w_q = (x_q, y_q)$.

Proposition 2.45. *If the Manhattan distance between w_p and w_q is even, then $\mu(w_p, w_q)$ is on the integer grid.*

Proof. By Invariant (ii) we know that $x_p < x_q$. Suppose without loss of generality that $y_p \leqslant y_q$. The Manhattan distance d of w_p and w_q is $x_q - x_p + y_q - y_p$, which by assumption is an even number. Adding the even number $2x_p$ to d yields the even number $x_q + x_p + y_q - y_p$, half of which is the x-coordinate of $\mu((x_p, y_p), (x_q, y_q))$. Adding the even number $2y_p$ to d yields the even number $x_q - x_p + y_q + y_p$, half of which is the y-coordinate of $\mu((x_p, y_p), (x_q, y_q))$. □

After Step n we have $P(v_n) = (n-2, n-2)$, because v_n is a neighbor of both v_1 and v_2. However, $P(v_{k+1})$ may not "see" all of $w_p, \dots, w_q$, in case that the slope of $w_p w_{p+1}$ is 1 and/or the slope of $w_{q-1} w_q$ is -1 (Figure 2.21).

In order to resolve these problems we shift some points around so that after the shift w_{p+1} does not lie on the line of slope 1 through w_p and w_{q-1} does not lie on the line of slope -1 through w_q. The process of inserting v_{k+1} then looks as follows.

1. Shift $\bigcup_{i=p+1}^{q-1} L(w_i)$ to the right by one unit.
2. Shift $\bigcup_{i=q}^{t} L(w_i)$ to the right by two units.
3. $P(v_{k+1}) := \mu(w_p, w_q)$.
4. $L(v_{k+1}) := \{v_{k+1}\} \cup \bigcup_{i=p+1}^{q-1} L(w_i)$.

[4]The notation is a bit sloppy here because both t and the w_i in general depend on k. So in principle we should write w_i^k instead of w_i. But as the k would just make a constant appearance throughout, we omit it to avoid index clutter.

[5]The *Manhattan distance* of two points (x_1, y_1) and (x_2, y_2) is $|x_2 - x_1| + |y_2 - y_1|$.

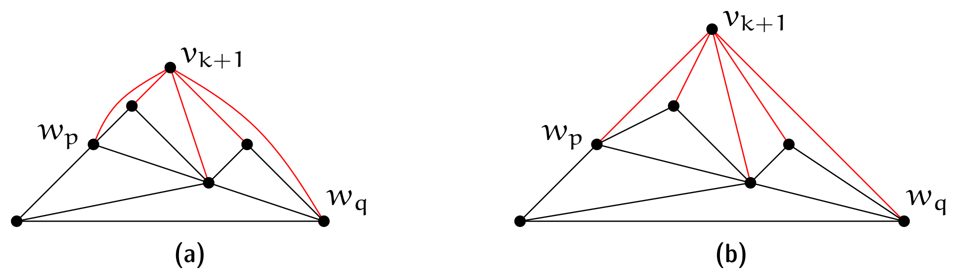


Figure 2.21: *(a) The new vertex v_{k+1} is adjacent to all of $w_p, \ldots, w_q$. If we place v_{k+1} at $\mu(w_p, w_q)$, then some edges may overlap, in case that w_{p+1} lies on the line of slope 1 through w_p or w_{q-1} lies on the line of slope -1 through w_q; (b) shifting $w_{p+1}, \ldots, w_{q-1}$ by one and $w_q, \ldots, w_t$ by two units to the right solves the problem.*

Observe that the Manhattan distance between w_p and w_q remains even, because the shift increases their x-difference by two and leaves the y-coordinates unchanged. Therefore by Proposition 2.45 the vertex v_{k+1} is embedded on the integer grid.

The slopes of the edges $w_p w_{p+1}$ and $w_{q-1} w_q$ (might be just a single edge, in case that $p+1 = q$) become < 1 in absolute value, whereas the slopes of all other edges along the outer cycle remain ± 1. As all edges from v_{k+1} to $w_{p+1}, \ldots, w_{q-1}$ have slope > 1 in absolute value, and the edges $v_{k+1} w_p$ and $v_{k+1} w_q$ have slope ± 1, each edge $v_{k+1} w_i$, for $i \in \{p, \ldots, q\}$ intersects the outer cycle in exactly one point, which is w_i. In other words, adding all edges from v_{k+1} to its neighbors in G_k as straight-line segments results in a plane drawing.

Next we argue that the invariants (i)–(iii) are maintained. For (i) note that we start shifting with w_{p+1} only so that even in case that v_1 is a neighbor of v_{k+1}, it is never shifted. On the other hand, v_2 is always shifted by two, because we shift every vertex starting from (and including) w_q. Clearly both the shifts and the insertion of v_{k+1} maintain the strict order along the outer cycle, and so (ii) continues to hold. Finally, regarding (iii) note that the edges $w_p w_{p+1}$ and $w_{q-1} w_q$ (possibly this is just a single edge) are the only edges on the outer cycle whose slope is changed by the shift. But these edges do not appear on $C_\circ(G_{k+1})$ anymore. The two edges $v_{k+1} w_p$ and $v_{k+1} w_q$ incident to the new vertex v_{k+1} that appear on $C_\circ(G_{k+1})$ have slope 1 and -1, respectively. So all of (i)–(iii) are invariants of the algorithm, indeed.

So far we have argued about the shift with respect to vertices on the outer cycle of G_k only. To complete the proof of Theorem 2.36 it remains to show that the drawing remains plane under shifts also in its interior part.

Lemma 2.46. *Let G_k, $k \geqslant 3$, be straight-line grid embedded as described, $C_\circ(G_k) = (w_1, \ldots, w_t)$, and let $\delta_1 \leqslant \ldots \leqslant \delta_t$ be nonnegative integers. If for each i, we shift $L(w_i)$ by δ_i to the right, then the resulting straight-line drawing is plane.*

Proof. Induction on k. For G_3 this is obvious. Let $v_k = w_\ell$, for some $1 < \ell < t$.

Construct a delta sequence Δ for G_{k-1} as follows. If v_k has only two neighbors in G_k, then $C_\circ(G_{k-1}) = (w_1, \dots, w_{\ell-1}, w_{\ell+1}, \dots, w_t)$ and we set $\Delta = \delta_1, \dots, \delta_{\ell-1}, \delta_{\ell+1}, \dots, \delta_t$. Otherwise, $C_\circ(G_{k-1}) = (w_1, \dots, w_{\ell-1} = u_1, \dots, u_m = w_{\ell+1}, \dots, w_t)$, where $u_1, \dots, u_m$ are the $m \geqslant 3$ neighbors of v_k in G_k. In this case we set

$$\Delta = \delta_1, \dots, \delta_{\ell-1}, \underbrace{\delta_\ell, \dots, \delta_\ell}_{m-2 \text{ times}}, \delta_{\ell+1}, \dots, \delta_t \,.$$

Clearly, Δ is monotonely increasing and by the inductive assumption a correspondingly shifted drawing of G_{k-1} is plane. When adding v_k and its incident edges back, the drawing remains plane: All vertices $u_2, \dots, u_{m-1}$ (possibly none) move rigidly with (by exactly the same amount as) v_k by construction. Stretching the edges of the chain $w_{\ell-1}, w_\ell, w_{\ell+1}$ by moving $w_{\ell-1}$ to the left and/or $w_{\ell+1}$ to the right cannot create any crossings. □

Linear time. The challenge in implementing the shift algorithm efficiently lies in the eponymous shift operations, which modify the x-coordinates of potentially many vertices. In fact, it is not hard to see that a naive implementation—which keeps track of all coordinates explicitly—may use quadratic time. De Fraysseix et al. described an implementation of the shift algorithm that uses $O(n \log n)$ time. Then Chrobak and Payne [10] observed how to improve the runtime to linear, using the following ideas.

Recall that $P(v_{k+1}) = (x_{k+1}, y_{k+1})$, where

$$\begin{aligned} x_{k+1} &= \frac{1}{2}(x_p - y_p + x_q + y_q) \text{ and} \\ y_{k+1} &= \frac{1}{2}(-x_p + y_p + x_q + y_q) = \frac{1}{2}((x_q - x_p) + y_p + y_q)\,. \end{aligned} \tag{2.47}$$

Thus,

$$x_{k+1} - x_p = \frac{1}{2}((x_q - x_p) + y_q - y_p)\,. \tag{2.48}$$

In other words, we need the y-coordinates of w_p and w_q together with the relative x-position (offset) of w_p and w_q only to determine the y-coordinate of v_{k+1} and its offset to w_p.

Maintain the outer cycle as a rooted binary tree T, with root v_1. For each node v of T, the *left child* is the first vertex covered by insertion of v (if any), that is, w_{p+1} in the terminology from above (if $p+1 \neq q$), whereas the *right child* of v is the next node along the outer cycle (if any; either along the current outer cycle or along the one at the point where both points were covered together). See Figure 2.22 for an example.

At each node v of T we also store its x-offset $\mathrm{dx}(v)$ with respect to the parent node. For the root v_1 of the tree set $\mathrm{dx}(v_1) = 0$. In this way, a whole subtree (and, thus, a whole set $L(\cdot)$) can be shifted by changing a single offset entry at its root.

Initially, $\mathrm{dx}(v_1) = 0$, $\mathrm{dx}(v_2) = \mathrm{dx}(v_3) = 1$, $y(v_1) = y(v_2) = 0$, $y(v_3) = 1$, $\mathrm{left}(v_1) = \mathrm{left}(v_2) = \mathrm{left}(v_3) = 0$, $\mathrm{right}(v_1) = v_3$, $\mathrm{right}(v_2) = 0$, and $\mathrm{right}(v_3) = v_2$.

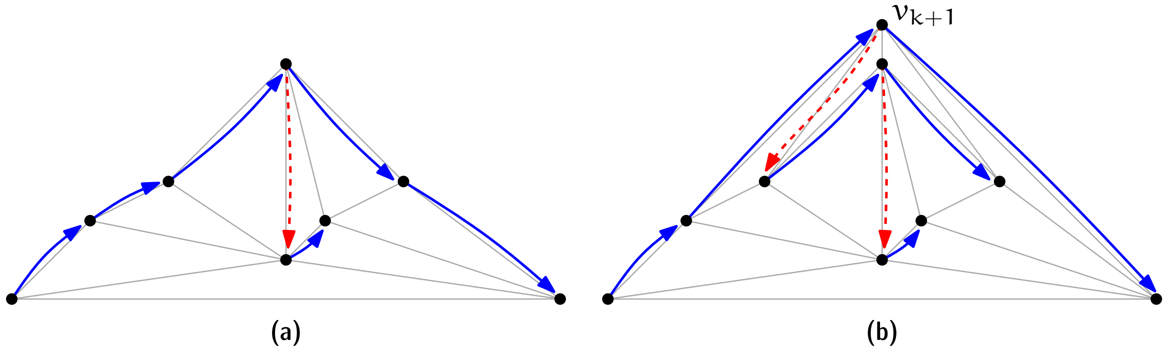


Figure 2.22: *Maintaining the binary tree representation when inserting a new vertex v_{k+1}. Red (dashed) arrows point to left children, blue (solid) arrows point to right children.*

Inserting a vertex v_{k+1} works as follows. As before, let $w_1, \ldots, w_t$ denote the vertices on the outer cycle $C_\circ(G_k)$ and $w_p, \ldots, w_q$ be the neighbors of v_{k+1}.

1. Increment $\mathrm{dx}(w_{p+1})$ and $\mathrm{dx}(w_q)$ by one. *(This implements the shift.)*
2. Compute $\Delta_{pq} = \sum_{i=p+1}^{q} \mathrm{dx}(w_i)$. *(This is the total offset between w_p and w_q.)*
3. Set $\mathrm{dx}(v_k) \leftarrow \frac{1}{2}(\Delta_{pq} + y(w_q) - y(w_p))$ and $y(v_k) \leftarrow \frac{1}{2}(\Delta_{pq} + y(w_q) + y(w_p))$. *(This is exactly what we derived in (2.47) and (2.48).)*
4. Set $\mathrm{right}(w_p) \leftarrow v_k$ and $\mathrm{right}(v_k) \leftarrow w_q$. *(Update the current outer cycle.)*
5. If $p + 1 = q$, then set $\mathrm{left}(v_k) \leftarrow 0$; else set $\mathrm{left}(v_k) \leftarrow w_{p+1}$ and $\mathrm{right}(w_{q-1}) \leftarrow 0$. *(Update $L(v_{k+1})$, the part that is covered by insertion of v_{k+1}.)*
6. Set $\mathrm{dx}(w_q) \leftarrow \Delta_{pq} - \mathrm{dx}(v_k)$ and—unless $p + 1 = q$—set $\mathrm{dx}(w_{p+1}) \leftarrow \mathrm{dx}(w_{p+1}) - \mathrm{dx}(v_k)$. *(Update the offsets according to the changes in the previous two steps.)*

Observe that the only step that possibly cannot be executed in constant time is Step 2. But all vertices but the last vertex w_q for which we sum the offsets are covered by the insertion of v_{k+1}. As every vertex can be covered at most once, the overall complexity of this step during the algorithm is linear. Therefore, this first phase of the algorithm can be completed in linear time.

In a second phase, the final x-coordinates can be computed from the offsets by a single recursive pre-order traversal of the tree. The (pseudo–)code given below is to be called with the root vertex v_1 and an offset of zero. Clearly this yields a linear time algorithm overall.

```
compute_coordinate(Vertex v, Offset d) {
  if (v == 0) return;
  x(v) = dx(v) + d;
  compute_coordinate(left(v), x(v));
  compute_coordinate(right(v), x(v));
}
```

2.5.3 Remarks and Open Problems

From a geometric complexity point of view, Theorem 2.36 provides very good news for planar graphs in a similar way that the Euler Formula does from a combinatorial complexity point of view. Euler's Formula tells us that we can obtain a combinatorial representation (for instance, as a DCEL) of any plane graph using $O(n)$ space, where n is the number of vertices.

Now the shift algorithm tells us that for any planar graph we can even find a geometric plane (straight-line) representation using $O(n)$ space. In addition to the combinatorial information, we only have to store $2n$ numbers from the range $\{0, 1, \ldots, 2n-4\}$.

When we make such claims regarding space complexity we implicitly assume the so-called *word RAM model*. In this model each address in memory contains a *word* of b bits, which means that it can be used to represent any integer from $\{0, \ldots, 2^b - 1\}$. One also assumes that b is sufficiently large, for instance, in our case $b \geqslant \log n$.

There are also different models such as the *bit complexity model*, where one is charged for every bit used to store information. In our case that would already incur an additional factor of $\log n$ for the combinatorial representation: for instance, for each halfedge we store its endpoint, which is an index from $\{1, \ldots, n\}$.

Edge lengths. Theorem 2.36 shows that planar graphs admit a plane straight-line drawing where all vertices have integer coordinates. It is an open problem whether a similar statement can be made for edge lengths.

Problem 2.49 (Harborth's Conjecture [16]). Every planar graph admits a plane straight-line drawing where all Euclidean edge lengths are integral.

Without the planarity restriction such a drawing is possible because for every $n \in \mathbb{N}$ one can find a set of n points in the plane, not all collinear, such that their distances are all integral. In fact, such a set of points can be constructed to lie on a circle of integral radius [2]. When mapping the vertices of K_n onto such a point set, all edge lengths are integral. In the same paper it is also shown that there exists no infinite set of points in the plane so that all distances are integral, unless all of these points are collinear. Unfortunately, collinear point sets are not very useful for drawing graphs. The existence of a dense subset of the plane where all distances are rational would resolve Harborth's Conjecture. However, it is not known whether such a set exists, and in fact the suspected answer is "no".

Problem 2.50 (Erdős–Ulam Conjecture [11]). There is no dense set of points in the plane whose Euclidean distances are all rational.

Generalizing the Fáry-Wagner Theorem. As discussed above, not every planar graph on n vertices admits a plane straight-line embedding on every set of n points. But Theorem 2.33 states that for every planar graph G on n vertices there *exists* a set P of n points in the plane so that G admits a plane straight-line embedding on P (that is, so that the vertices of G are mapped bijectively to the points in P). It is an open problem whether this statement can be generalized to hold for several graphs, in the following sense.

Problem 2.51. What is the largest number $k \in \mathbb{N}$ for which the following statement holds? For every collection of k planar graphs $G_1, \ldots, G_k$ on n vertices each, there exists a set P of n points so that G_i admits a plane straight-line embedding on P, for every $i \in \{1, \ldots, k\}$.

By Theorem 2.33 we know that the statement holds for $k = 1$. Already for $k = 2$ it is not known whether the statement holds. However, it is known that k is finite. Specifically, there exists a collection of $7,393$ planar graphs on 35 vertices each so that for every set P of 35 points in the plane at least one of these graphs does not admit a plane straight-line embedding on P [7]. Therefore we have $k \leqslant 7392$.

Questions

1. *What is an embedding? What is a planar/plane graph?* Give the definitions and explain the difference between planar and plane.
2. *How many edges can a planar graph have? What is the average vertex degree in a planar graph?* Explain Euler's formula and derive your answers from it.
3. *How can plane graphs be represented on a computer?* Explain the DCEL data structure and how to work with it.
4. *How can a given plane graph be (topologically) triangulated efficiently?* Explain what it is, including the difference between topological and geometric triangulation. Give a linear time algorithm, for instance, as in Theorem 2.27.
5. *What is a combinatorial embedding? When are two combinatorial embeddings equivalent? Which graphs have a unique combinatorial plane embedding?* Give the definitions, explain and prove Whitney's Theorem.
6. *What is a canonical ordering and which graphs admit such an ordering? For a given graph, how can one find a canonical ordering efficiently?* Give the definition. State and prove Theorem 2.38.
7. *Which graphs admit a plane embedding using straight line edges? Can one bound the size of the coordinates in such a representation?* State and prove Theorem 2.36.

References

[1] Oswin Aichholzer, Thomas Hackl, Matias Korman, Marc van Kreveld, Maarten Löffler, Alexander Pilz, Bettina Speckmann, and Emo Welzl, Packing plane spanning trees and paths in complete geometric graphs. *Inform. Process. Lett.*, **124**, (2017), 35–41.

[2] Norman H. Anning and Paul Erdős, Integral distances. *Bull. Amer. Math. Soc.*, **51**, 8, (1945), 598–600.

[3] Bruce G. Baumgart, A polyhedron representation for computer vision. In *Proc. AFIPS Natl. Comput. Conf.*, vol. 44, pp. 589–596, AFIPS Press, Alrington, Va., 1975.

[4] Prosenjit Bose, Ferran Hurtado, Eduardo Rivera-Campo, and David R. Wood, Partitions of complete geometric graphs into plane trees. *Comput. Geom. Theory Appl.*, **34**, 2, (2006), 116–125.

[5] John M. Boyer and Wendy J. Myrvold, On the cutting edge: simplified O(n) planarity by edge addition. *J. Graph Algorithms Appl.*, **8**, 3, (2004), 241–273.

[6] Sergio Cabello, Planar embeddability of the vertices of a graph using a fixed point set is NP-hard. *J. Graph Algorithms Appl.*, **10**, 2, (2006), 353–363.

[7] Jean Cardinal, Michael Hoffmann, and Vincent Kusters, On universal point sets for planar graphs. *J. Graph Algorithms Appl.*, **19**, 1, (2015), 529–547.

[8] Bernard Chazelle, Triangulating a simple polygon in linear time. *Discrete Comput. Geom.*, **6**, 5, (1991), 485–524.

[9] Norishige Chiba and Takao Nishizeki, The Hamiltonian cycle problem is linear-time solvable for 4-connected planar graphs. *J. Algorithms*, **10**, 2, (1989), 187–211.

[10] Marek Chrobak and Thomas H. Payne, A linear-time algorithm for drawing a planar graph on a grid. *Inform. Process. Lett.*, **54**, (1995), 241–246.

[11] Paul Erdős, Ulam, the man and the mathematician. *J. Graph Theory*, **9**, 4, (1985), 445–449.

[12] István Fáry, On straight lines representation of planar graphs. *Acta Sci. Math. Szeged*, **11**, 4, (1948), 229–233.

[13] Hubert de Fraysseix, Patrice Ossona de Mendez, and Pierre Rosenstiehl, Trémaux trees and planarity. *Internat. J. Found. Comput. Sci.*, **17**, 5, (2006), 1017—-1030.

[14] Hubert de Fraysseix, János Pach, and Richard Pollack, How to draw a planar graph on a grid. *Combinatorica*, **10**, 1, (1990), 41–51.

[15] Leonidas J. Guibas and Jorge Stolfi, Primitives for the manipulation of general subdivisions and the computation of Voronoi diagrams. *ACM Trans. Graph.*, **4**, 2, (1985), 74–123.

[16] Heiko Harborth and Arnfried Kemnitz, Plane integral drawings of planar graphs. *Discrete Math.*, **236**, 1–3, (2001), 191–195.

[17] John Hopcroft and Robert E. Tarjan, Efficient planarity testing. *J. ACM*, **21**, 4, (1974), 549–568.

[18] Ken ichi Kawarabayashi, Yusuke Kobayashi, and Bruce Reed, The disjoint paths problem in quadratic time. *J. Combin. Theory Ser. B*, **102**, 2, (2012), 424–435.

[19] Lutz Kettner, *Software design in computational geometry and contour-edge based polyhedron visualization*. Ph.D. thesis, ETH Zürich, Zürich, Switzerland, 1999.

[20] Kazimierz Kuratowski, Sur le problème des courbes gauches en topologie. *Fund. Math.*, **15**, 1, (1930), 271–283.

[21] László Lovász, Graph minor theory. *Bull. Amer. Math. Soc.*, **43**, 1, (2006), 75–86.

[22] Bojan Mohar and Carsten Thomassen, *Graphs on surfaces*, Johns Hopkins University Press, Baltimore, 2001.

[23] David E. Muller and Franco P. Preparata, Finding the intersection of two convex polyhedra. *Theoret. Comput. Sci.*, **7**, (1978), 217–236.

[24] János Pach and Rephael Wenger, Embedding planar graphs at fixed vertex locations. *Graphs Combin.*, **17**, (2001), 717–728.

[25] Neil Robertson and Paul Seymour, Graph Minors. XX. Wagner's Conjecture. *J. Combin. Theory Ser. B*, **92**, 2, (2004), 325–357.

[26] Walter Schnyder, Planar graphs and poset dimension. *Order*, **5**, (1989), 323–343.

[27] Carsten Thomassen, Kuratowski's Theorem. *J. Graph Theory*, **5**, 3, (1981), 225–241.

[28] William T. Tutte, A theorem on planar graphs. *Trans. Amer. Math. Soc.*, **82**, 1, (1956), 99–116.

[29] Klaus Wagner, Bemerkungen zum Vierfarbenproblem. *Jahresbericht der Deutschen Mathematiker-Vereinigung*, **46**, (1936), 26–32.

[30] Klaus Wagner, Über eine Eigenschaft der ebenen Komplexe. *Math. Ann.*, **114**, 1, (1937), 570–590.

[31] Kevin Weiler, Edge-based data structures for solid modeling in a curved surface environment. *IEEE Comput. Graph. Appl.*, **5**, 1, (1985), 21–40.

[32] Hassler Whitney, Congruent graphs and the connectivity of graphs. *Amer. J. Math.*, **54**, 1, (1932), 150–168.

Chapter 3

Polygons

Although we can think of a line $\ell \subset \mathbb{R}^2$ as an infinite point set that consists of all points in $\mathbb{R}^2$ that are on ℓ, there still exists a finite description for ℓ. Such a description is, for instance, provided by the three coefficients $a, b, c \in \mathbb{R}$ of an equation of the form $ax + by = c$, with $(a, b) \neq (0, 0)$. Actually this holds true for all of the fundamental geometric objects that were mentioned in Chapter 1: Each of them has constant *description complexity* (or, informally, just *size*), that is, it can be described by a constant[1] number of parameters.

In this course we will typically deal with objects that are not of constant size. Often these are formed by merely aggregating constant-size objects, for instance, points to form a finite set of points. But sometimes we also demand additional structure that goes beyond aggregation only. Probably the most fundamental geometric objects of this type are what we call *polygons*. You probably learned this term in school, but what *is* a polygon precisely? Consider the examples shown in Figure 3.1. Are these all polygons? If not, where would you draw the line?

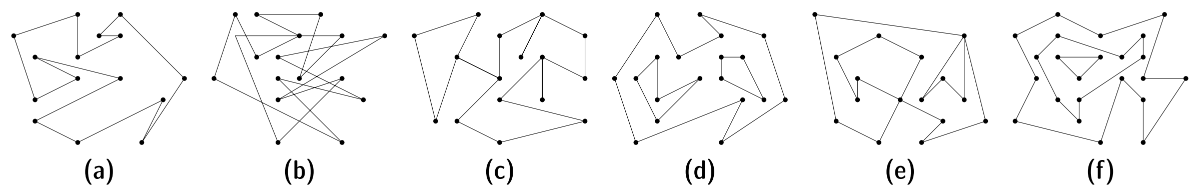


Figure 3.1: *What is a polygon?*

3.1 Classes of Polygons

Obviously, there is not *the* right answer to such a question and certainly there are different types of polygons. Often the term polygon is used somewhat sloppily in place

[1]Unless specified differently, we will always assume that the dimension is (a small) constant. In a high-dimensional space $\mathbb{R}^d$, one has to account for a description complexity of $\Theta(d)$.

of what we call a *simple polygon*, defined below.

Definition 3.1. *A* **simple polygon** *is a compact region* $P \subset \mathbb{R}^2$ *that is bounded by a simple closed curve* $\gamma : [0,1] \to \mathbb{R}^2$ *that consists of a finite number of line segments. A* **curve** *is a continuous map* $\gamma : [0,1] \to \mathbb{R}^2$*. A curve* γ *is* **closed***, if* $\gamma(0) = \gamma(1)$ *and it is* **simple** *if it is injective on* $[0,1)$*, that is, the curve does not intersect itself.*

Out of the examples shown above only Polygon 3.1a is simple. For each of the remaining polygons it is impossible to combine the bounding segments into a simple closed curve.

The term *compact* for subsets of $\mathbb{R}^d$ means bounded and closed. A subset of $P \subset \mathbb{R}^d$ is *bounded*, if it is contained in the ball of radius r around the origin, for some finite $r > 0$. Being closed means that the boundary is considered to be part of the polygon. In order to formally define these terms, let us briefly review a few basic notions from topology.

The standard topology of $\mathbb{R}^d$ is defined in terms of the Euclidean metric. A point $p \in \mathbb{R}^d$ is *interior* to a set $P \subseteq \mathbb{R}^d$, if there exists an ε-ball $B_\varepsilon(p) = \{x \in \mathbb{R}^d : \|x-p\| < \varepsilon\}$ around p, for some $\varepsilon > 0$, that is completely contained in P. A set is *open*, if all of its points are interior; and it is *closed*, if its complement is open.

Exercise 3.2. *Determine for each of the following sets whether they are open or closed in* $\mathbb{R}^2$*.* *a)* $B_1(0)$ *b)* $\{(1,0)\}$ *c)* $\mathbb{R}^2$ *d)* $\mathbb{R}^2\backslash\mathbb{Z}^2$ *e)* $\mathbb{R}^2\backslash\mathbb{Q}^2$ *f)* $\{(x,y) : x \in \mathbb{R}, y \geqslant 0\}$

Exercise 3.3. *Show that the union of countably many open sets in* $\mathbb{R}^d$ *is open. Show that the union of a finite number of closed sets in* $\mathbb{R}^d$ *is closed. (These are two of the axioms that define a topology. So the statements are needed to assert that the metric topology is a topology, indeed.) What follows for intersections of open and closed sets? Finally, show that the union of countably many closed sets in* $\mathbb{R}^d$ *is not necessarily closed.*

The *boundary* ∂P of a set $P \subset \mathbb{R}^d$ consists of all points that are neither interior to P nor to its complement $\mathbb{R}^d \setminus P$. By definition, for every $p \in \partial P$ every ball $B_\varepsilon(p)$ contains both points from P and from $\mathbb{R}^d \setminus P$. Sometimes one wants to consider a set $P \subset \mathbb{R}^d$ open although it is not. In that case one can resort to the *interior* P° of P that is formed by the subset of points interior to P. Similarly, the *closure* $\overline{P}$ of P is defined by $\overline{P} = P \cup \partial P$.

Lower-dimensional objects, such as line segments in $\mathbb{R}^2$ or triangles in $\mathbb{R}^3$, do not possess any interior point (because the ε-balls needed around any such point are full-dimensional). Whenever we want to talk about the interior of a lower-dimensional set S, we use the qualifier *relative* and write $\mathrm{relint}(S)$ to denote the interior of S relative to the smallest affine subspace that contains S.

For instance, the smallest affine subspace that contains a line segment is a line and so the relative interior of a line segment in $\mathbb{R}^2$ consists of all points except the endpoints, just like for an interval in $\mathbb{R}^1$. Similarly, for a triangle in $\mathbb{R}^3$ the smallest affine subspace that contains it is a plane. Hence its relative interior is just the interior of the triangle, considered as a two-dimensional object.

Exercise 3.4. *Show that for any* $P \subset \mathbb{R}^d$ *the interior* P° *is open. (Why is there something to show to begin with?) Show that for any* $P \subset \mathbb{R}^d$ *the closure* $\overline{P}$ *is closed.*

When describing a simple polygon P it is sufficient to describe only its boundary ∂P. As ∂P by definition is a simple closed curve γ that consists of finitely many line segments, we can efficiently describe it as a sequence $p_1, \ldots, p_n$ of points, such that γ is formed by the line segments $\overline{p_1 p_2}, \overline{p_2 p_3}, \ldots, \overline{p_{n-1} p_n}, \overline{p_n p_1}$. These points are referred to as the *vertices* of the polygon, and the segments connecting them are referred as the *edges* of the polygon. The *set of vertices* of a polygon P is denoted by $V(P)$, and the *set of edges* of P is denoted by $E(P)$.

Knowing the boundary, it is easy to tell apart the (bounded) interior from the (unbounded) exterior. This is asserted even for much more general curves by Theorem 2.1 (Jordan curve theorem). To prove this theorem in its full generality is surprisingly difficult. For simple polygons the situation is easier, though. The essential idea can be worked out algorithmically, which we leave as an exercise.

Exercise 3.5. *Describe an algorithm to decide whether a point lies inside or outside of a simple polygon. More precisely, given a simple polygon* $P \subset \mathbb{R}^2$ *as a list of its vertices* $(v_1, v_2, \ldots, v_n)$ *in counterclockwise order and a query point* $q \in \mathbb{R}^2$*, decide whether* q *is inside* P*, on the boundary of* P*, or outside. The runtime of your algorithm should be* $O(n)$*.*

There are good reasons to ask for the boundary of a polygon to form a simple curve: For instance, in the example depicted in Figure 3.1b there are several regions for which it is completely unclear whether they should belong to the interior or to the exterior of the polygon. A similar problem arises for the interior regions in Figure 3.1f. But there are more general classes of polygons that some of the remaining examples fall into. We will discuss only one such class here. It comprises polygons like the one from Figure 3.1d.

Definition 3.6. *A region* $P \subset \mathbb{R}^2$ *is a* **simple polygon with holes** *if it can be described as* $P = F \setminus \bigcup_{H \in \mathcal{H}} H^\circ$*, where* $\mathcal{H}$ *is a finite collection of pairwise disjoint simple polygons (called holes) and* F *is a simple polygon for which* $F^\circ \supset \bigcup_{H \in \mathcal{H}} H$*.*

The way this definition heavily depends on the notion of simple polygons makes it straightforward to derive a similar trichotomy as the Jordan Curve Theorem provides for simple polygons, that is, every point in the plane is either inside, or on the boundary, or outside of P (exactly one of these three).

3.2 Polygon Triangulation

From a topological point of view, a simple polygon is nothing but a disk and so it is a very elementary object. But geometrically a simple polygon can be—as if mocking the label we attached to it—a pretty complicated shape, see Figure 3.2 for an example. While there is an easy and compact one-dimensional representation in terms of the boundary, as a sequence of vertices/points, it is often desirable to work with a more structured representation of the whole two-dimensional shape.

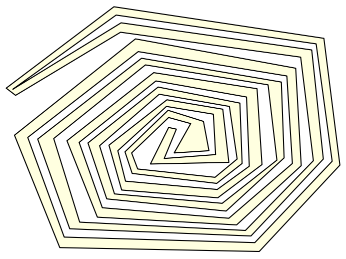


Figure 3.2: *A simple (?) polygon.*

For instance, it is not straightforward to compute the area of a general simple polygon. In order to do so, one usually describes the polygon in terms of simpler geometric objects, for which computing the area is easy. Good candidates for such shapes are triangles, rectangles, and trapezoids. Indeed, it is not hard to show that every simple polygon admits a "nice" partition into triangles, which we call a triangulation.

Definition 3.7. *A* **triangulation** *of a simple polygon* P *with vertex set* $V(P)$ *is a collection* $\mathcal{T}$ *of triangles, such that*

(1) $P = \bigcup_{T \in \mathcal{T}} T$;

(2) $V(P) = \bigcup_{T \in \mathcal{T}} V(T)$; *and*

(3) for every distinct pair $T, U \in \mathcal{T}$, *the intersection* $T \cap U$ *is either a common vertex, or a common edge, or empty.*

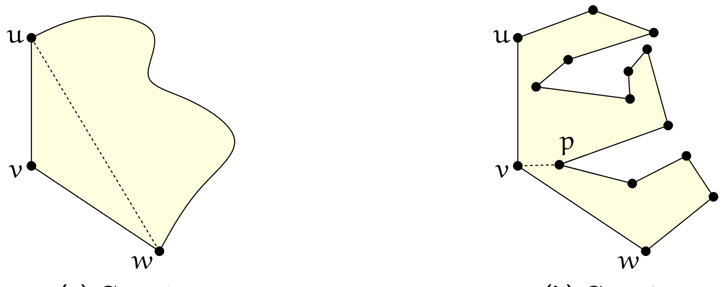
Exercise 3.8. *Show that each condition in Definition 3.7 is necessary in the following sense: Give an example of a non-triangulation that would form a triangulation if the condition was omitted. Is the definition equivalent if (3) is replaced by* $T^\circ \cap U^\circ = \emptyset$, *for every distinct pair* $T, U \in \mathcal{T}$?

If we are given a triangulation of a simple polygon P it is easy to compute the area of P by simply summing up the area of all triangles from $\mathcal{T}$. Triangulations are an incredibly useful tool in planar geometry, and one reason for their importance is that every simple polygon admits one.

Theorem 3.9. *Every simple polygon has a triangulation.*

Proof. Let P be a simple polygon on n vertices. We prove the statement by induction on n. For $n = 3$ we face a triangle P that is a triangulation by itself. For $n > 3$ consider the lexicographically smallest vertex v of P, that is, among all vertices of P with a smallest x-coordinate the one with smallest y-coordinate. Denote the neighbors of v (next vertices) along ∂P by u and w. Consider the line segment $\overline{uw}$. We distinguish two cases.

Case 1: except for its endpoints u and w, the segment $\overline{uw}$ lies completely in P°. Then $\overline{uw}$ splits P into two smaller polygons, the triangle uvw and a simple polygon P' on $n-1$ vertices (Figure 3.3a). By the inductive hypothesis, P' has a triangulation that together with T yields a triangulation of P.



(a) Case 1. **(b)** Case 2.

Figure 3.3: *Cases in the proof of Theorem 3.9.*

Case 2: $\text{relint}(\overline{uw}) \not\subset P^\circ$ (Figure 3.3b). By choice of v, the polygon P is contained in the closed halfplane to the right of the vertical line through v. Therefore, as the segments $\overline{uv}$ and $\overline{vw}$ are part of a simple closed curve defining ∂P, every point sufficiently close to v and between the rays vu and vw must be in P°.

On the other hand, since $\text{relint}(\overline{uw}) \not\subset P^\circ$, there is some point from ∂P in the interior of the triangle $T = uvw$ (by the choice of v the points u, v, w are not collinear and so T is a triangle, indeed) or on the line segment $\overline{uw}$. In particular, as ∂P is composed of line segments, there is a vertex of P in T° or on $\overline{uw}$ (otherwise, a line segment would have to intersect the line segment $\overline{uw}$ twice, which is impossible). Among all such vertices select p to be one that is furthest from the line uw. Then the open line segment $\text{relint}(\overline{vp})$ is contained in T° and, thus, it splits P into two polygons P_1 and P_2 on less than n vertices each (in one of them, u does not appear as a vertex, whereas w does not appear as a vertex in the other). By the inductive hypothesis, both P_1 and P_2 have triangulations and their union yields a triangulation of P. □

Exercise 3.10. *In the proof of Theorem 3.9, would the argument in Case 2 also work if the point* p *was chosen to be a vertex of* P *in* T° *that is closest to* v *(in Euclidean distance)?*

The configuration from Case 1 above is called an *ear*: three consecutive vertices u, v, w of a simple polygon P such that the relative interior of $\overline{uw}$ lies in P°. In fact, we could have skipped the analysis for Case 2 by referring to the following theorem.

Theorem 3.11 (Meisters [13, 14]). *Every simple polygon that is not a triangle has two non-overlapping ears, that is, two ears* A *and* B *such that* $A^\circ \cap B^\circ = \emptyset$.

But knowing Theorem 3.9 we can obtain Theorem 3.11 as a direct consequence of the following

Theorem 3.12. *Every triangulation of a simple polygon on $n \geqslant 4$ vertices contains at least two (triangles that are) ears.*

Exercise 3.13. *Prove Theorem 3.12.*

Exercise 3.14. *Let P be a simple polygon with vertices $v_1, v_2, \dots, v_n$ (in counterclockwise order), where v_i has coordinates (x_i, y_i). Show that the area of P is*

$$\frac{1}{2}\sum_{i=1}^{n} x_i y_{i+1} - x_{i+1} y_i,$$

where $(x_{n+1}, y_{n+1}) = (x_1, y_1)$.

The number of edges and triangles in a triangulation of a simple polygon are completely determined by the number of vertices, as the following simple lemma shows.

Lemma 3.15. *Every triangulation of a simple polygon on $n \geqslant 3$ vertices consists of $n-2$ triangles and $2n-3$ edges.*

Proof. Proof by induction on n. The statement is true for $n = 3$. For $n > 3$ consider a simple polygon P on n vertices and an arbitrary triangulation T of P. Any edge uv in T that is not an edge of P (and there must be such an edge because P is not a triangle) partitions P into two polygons P_1 and P_2 with n_1 and n_2 vertices, respectively. Since $n_1, n_2 < n$ we conclude by the inductive hypothesis that T partitions P_1 into $n_1 - 2$ triangles and P_2 into $n_2 - 2$ triangles, using $2n_1 - 3$ and $2n_2 - 3$ edges, respectively.

All vertices of P appear in exactly one of P_1 or P_2, except for u and v, which appear in both. Therefore $n_1 + n_2 = n+2$ and so the number of triangles in T is $(n_1-2)+(n_2-2) = (n_1 + n_2) - 4 = n + 2 - 4 = n - 2$. Similarly, all edges of T appear in exactly one of P_1 or P_2, except for the edge uv, which appears in both. Therefore the number of edges in T is $(2n_1 - 3) + (2n_2 - 3) - 1 = 2(n_1 + n_2) - 7 = 2(n + 2) - 7 = 2n - 3$. □

The universal presence of triangulations is something particular about the plane: The natural generalization of Theorem 3.9 to dimension three and higher does not hold. What is this generalization, anyway?

Tetrahedralizations in $\mathbb{R}^3$. A simple polygon is a planar object that is a topological disk that is locally bounded by patches of lines. The corresponding term in $\mathbb{R}^3$ is a *polyhedron*, and although we will not formally define it here yet, a literal translation of the previous sentence yields an object that topologically is a ball and is locally bounded by patches of planes. A triangle in $\mathbb{R}^2$ corresponds to a tetrahedron in $\mathbb{R}^3$ and a *tetrahedralization* is a nice partition into tetrahedra, where "nice" means that the union of the tetrahedra covers the object, the vertices of the tetrahedra are vertices of the polyhedron, and any two distinct tetrahedra intersect in either a common triangular face, or a common edge, or a common vertex, or not at all.[2]

[2]These "nice" subdivisions can be defined in an abstract combinatorial setting, where they are called *simplicial complices.*

Unfortunately, there are polyhedra in $\mathbb{R}^3$ that do not admit a tetrahedralization. The following construction is due to Schönhardt [17]. It is based on a triangular prism, that is, two congruent triangles placed in parallel planes where the corresponding sides of both triangles are connected by a rectangle (Figure 3.4a). Then one triangle is twisted/rotated slightly within its plane. As a consequence, the rectangular faces are not plane anymore, but they obtain an inward dent along their diagonal in direction of the rotation (Figure 3.4b). The other (former) diagonals of the rectangular faces—labeled ab', bc', and

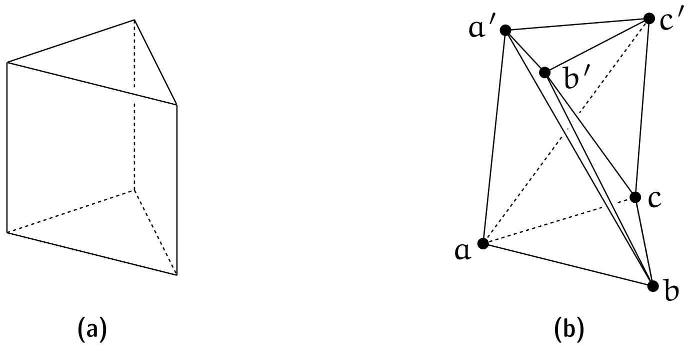


Figure 3.4: *The Schönhardt polyhedron cannot be subdivided into tetrahedra without adding new vertices.*

ca' in Figure 3.4b—are now epigonals, that is, they lie in the exterior of the polyhedron. Since these epigonals are the only edges between vertices that are not part of the polyhedron, there is no way to add edges to form a tetrahedron for a subdivision. Clearly the polyhedron is not a tetrahedron by itself, and so we conclude that it does not admit a subdivision into tetrahedra without adding new vertices. Actually, it is NP-complete to decide whether a non-convex polyhedron has a tetrahedralization [15]. If adding new vertices—so-called Steiner vertices—is allowed, then there is no problem to construct a tetrahedralization, and this holds true in general. Even if a tetrahedralization of a polyhedron exists, there is another significant difference to polygons in $\mathbb{R}^2$. While the number of triangles in a triangulation of a polygon depends only on the number of vertices, the number of tetrahedra in two different tetrahedralization of the same polyhedron may be different. See Figure 3.5 for a simple example of a polyhedron that has tetrahedralization with two or three tetrahedra. Deciding whether a convex polyhedron has a tetrahedralization with at most a given number of tetrahedra is NP-complete [6].

Exercise 3.16. *Characterize all possible tetrahedralizations of the three-dimensional cube.*

Algorithms. Knowing that a triangulation exists is nice, but it is much better to know that it can also be constructed efficiently.

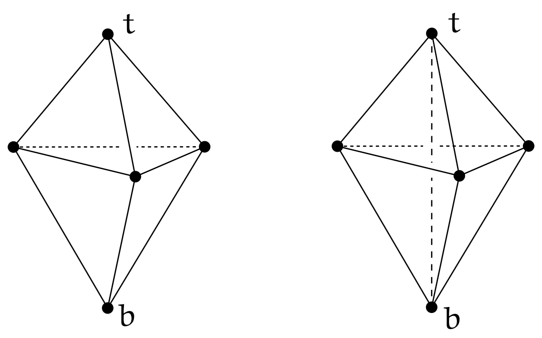


Figure 3.5: *Two tetrahedralizations of the same polyhedron, a triangular bipyramid. The left partition uses two polyhedra; both the top vertex t and the bottom vertex b belong to only one tetrahedron. The right partition uses three polyhedra that all share the dashed diagonal bt.*

Exercise 3.17. *Convert Theorem 3.9 into an $O(n^2)$ time algorithm to construct a triangulation for a given simple polygon on n vertices.*

The runtime achieved by the straightforward application of Theorem 3.9 is not optimal. We will revisit this question at several times during this course[3] and discuss improved algorithms for the problem of triangulating a simple polygon.

The best (in terms of worst-case runtime) algorithm known due to Chazelle [7] computes a triangulation in linear time. But this algorithm is very complicated and we will not discuss it here. There is also a somewhat simpler randomized algorithm to compute a triangulation in expected linear time [4], which we will not discuss in detail, either. The question of whether there exists a simple (which is not really a well-defined term, of course, except that Chazelle's Algorithm does not qualify) deterministic linear time algorithm to triangulate a simple polygon remains open [10].

Polygons with holes. It is interesting to note that the complexity of the triangulation problem changes to $\Theta(n \log n)$, if the polygon may contain holes [5]. This means that there is an algorithm to construct a triangulation for a given simple polygon with holes on a total of n vertices (counting both the vertices on the outer boundary and those of holes) in $O(n \log n)$ time. But there is also a lower bound of $\Omega(n \log n)$ operations that holds in all models of computation in which there exists a corresponding lower bound for comparison-based sorting. This difference in complexity is a very common pattern: There are many problems that are (sometimes much) harder for simple polygons with holes than for simple polygons. So maybe the term "simple" has some justification, after all...

[3]This is actually not true in this iteration of the course. But in the full version of the lecture notes you can find the corresponding material in the appendix, in chapters A and C.

General triangle covers. What if we drop the "niceness" conditions required for triangulations and just want to describe a given simple polygon as a union of triangles? It turns out this is a rather drastic change and, for instance, it is unlikely that we can efficiently find an optimal/minimal description of this type: Christ has shown [8] that it is NP-hard to decide whether for a simple polygon P on n vertices and a positive integer k, there exists a set of at most k triangles whose union is P. In fact, the problem is not even known to be in NP, because it is not clear whether the coordinates of solutions can always be encoded compactly.

3.3 The Art Gallery Problem

In 1973 Victor Klee posed the following question: "How many guards are necessary, and how many are sufficient to patrol the paintings and works of art in an art gallery with n walls?" From a geometric point of view, we may think of an "art gallery with n walls" as a simple polygon bounded by n edges, that is, a simple polygon P with n vertices. And a guard can be modeled as a point where we imagine the guard to stand and observe everything that is in sight. In sight, finally, refers to the walls of the gallery (edges of the polygon) that are opaque and, thus, prevent a guard to see what is behind. In other words, a guard (point) g can watch over every point $p \in P$, for which the line segment $\overline{gp}$ lies completely in P°, see Figure 3.6.

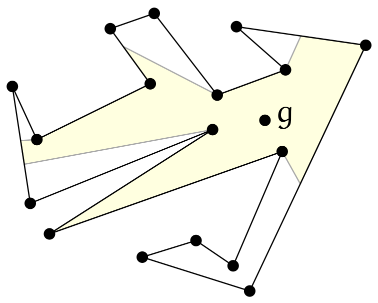


Figure 3.6: *The region that a guard g can observe.*

It is not hard to see that $\lfloor n/3 \rfloor$ guards are necessary in general.

Exercise 3.18. *Describe a family $(P_n)_{n \geqslant 3}$ of simple polygons such that P_n has n vertices and at least $\lfloor n/3 \rfloor$ guards are needed to guard it.*

What is more surprising: $\lfloor n/3 \rfloor$ guards are always sufficient as well. Chvátal [9] was the first to prove that, but then Fisk [11] gave a much simpler proof using—you may have guessed it—triangulations. Fisk's proof was considered so beautiful that it was included into "Proofs from THE BOOK" [3], a collection inspired by Paul Erdős' belief in "a place where God keeps aesthetically perfect proofs". The proof is based on the following lemma.

Lemma 3.19. *Every triangulation of a simple polygon is 3-colorable. That is, each vertex can be assigned one of three colors in such a way that adjacent vertices receive different colors.*

Proof. Induction on n. For $n = 3$ the statement is obvious. For $n > 3$, by Theorem 3.12 the triangulation contains an ear uvw. Cutting off the ear creates a triangulation of a polygon on $n - 1$ vertices, which by the inductive hypothesis admits a 3-coloring. Now whichever two colors the vertices u and w receive in this coloring, there remains a third color to be used for v. □

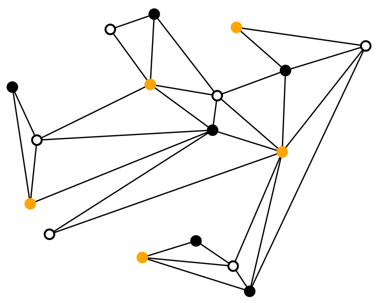


Figure 3.7: *A triangulation of a simple polygon on* 17 *vertices and a 3-coloring of it. The vertices shown solid orange form the smallest color class and guard the polygon using* $\lfloor 17/3 \rfloor = 5$ *guards.*

Theorem 3.20 (Fisk [11]). *Every simple polygon on* n *vertices can be guarded using at most* $\lfloor n/3 \rfloor$ *guards.*

Proof. Consider a triangulation of the polygon and a 3-coloring of the vertices as ensured by Lemma 3.19. Take the smallest color class, which clearly consists of at most $\lfloor n/3 \rfloor$ vertices, and put a guard at each vertex. As every point of the polygon is contained in at least one triangle and every triangle has exactly one vertex in the guarding set, the whole polygon is guarded. □

3.4 Optimal Guarding

While Exercise 3.18 shows that the bound in Theorem 3.20 is tight in general, it is easy to see that Fisk's method does not necessarily minimize the number of guards. Also, it is natural to lift the restriction that guards can be placed at vertices only, and allow guards to be placed anywhere on the boundary or even anywhere in the interior of the polygon. In all these variants, we can ask for the minimum number of guards required to guard a given polygon P. These problems have been shown to be NP-hard by Lee and Lin [12] already in the 1980s. However, if the guards are not constrained to lie on vertices, it is not clear whether the corresponding decision problem actually is in NP. In

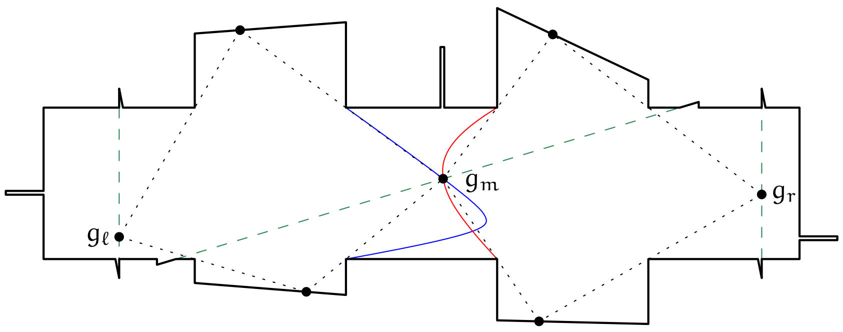


Figure 3.8: *To guard this polygon with three guards, there must be one guard on each of the green dashed segments. The middle guard g_m must be to the left of the blue curve, to the right of the red curve, and on the dashed green line. The intersection point of these three curves has irrational coordinates.*

fact, recent results by Abrahamsen et al. suggest that this is unlikely to be the case. In the remainder of this section we will briefly discuss some of these results.

Recall that, to show that a problem is in NP, one usually describes a *certificate* that allows to verify a solution for any problem instance in polynomial time. If we restrict the guards to be on vertices, a natural certificate for a solution is the set of vertices on which we place guards. In the general problem, a natural candidate for a certificate are the coordinates of the guards. Since no more than $\lfloor n/3 \rfloor$ guards are required, this seems a reasonable certificate. But what if the number of bits needed to explicitly represent these coordinates are exponential in n? One might be tempted to think that any reasonable guard can be placed at an intersection point of some lines that are defined by polygon vertices. Alas, in general this is not correct: some guards with irrational coordinates may be required, even if all vertices of P have integral coordinates. This surprising result has been presented in 2017 and we will sketch its main ideas, referring to the paper by Abrahamsen, Adamaszek, and Miltzow [1] for more details and the exact construction.

Consider the polygon shown in Figure 3.8, which consists of a main rectangular region with triangular, rectangular, and trapezoidal regions attached. We will argue that, if this polygon is guarded with less than four guards, at least one of the guards has an irrational coordinate. The polygon contains three pairs of triangular regions with the following structure. Each pair is connected by a green dashed segment in the figure. This segment contains one edge of each of the two triangles and separates their interiors. Hence, a single guard that sees both of these triangles has to be placed on this separating segment. Further, there is no other point that can guard two of these six triangles. Therefore, if we have only three guards, each of them must be placed on one of these three disjoint segments. The small rectangular regions to the left, top, and bottom outside the main rectangular region further constrain the positions of the guards along these segments.

Let the guards be g_ℓ, g_m, and g_r, as in the figure. The guard g_ℓ cannot see all the points inside the left two trapezoidal regions, and thus g_m has to be placed appropriately.

For each position of g_ℓ on its segment, we get a unique rightmost position on which a second guard can be placed to guard the two trapezoids. The union of these points defines an arc that is a segment of a quadratic curve (the roots of a quadratic polynomial). We get an analogous curve for g_r and the two trapezoids attached to the right. By a careful choice of the vertex coordinates, these two curves cross at a point p that also lies on the segment for the guard g_m and has irrational coordinates. It then follows from a detailed argument (see [1]) that p is the only feasible placement of g_m. Let us point out that the choice of the vertex coordinates to achieve this is far from trivial. For example, there can only be a single line defined by two points with rational coordinates that passes through p, and this is the line on which the guard g_m is constrained to lie on.

Exercise 3.21. *Let P be a polygon with vertices on the integer grid, and let g be a point inside that polygon with at least one irrational coordinate. Show that there can be at most one diagonal of P passing through g.*

Nevertheless, the sketched construction leads to the following result.

Theorem 3.22 (Abrahamsen et al. [1]). *For any k, there is a simple polygon P with integer vertex coordinates such that P can be guarded by $3k$ guards, while a guard set having only rational coordinates requires $4k$ guards.*

Abrahamsen, Adamaszek, and Miltzow [2] showed recently that the art gallery problem is actually complete in a complexity class called $\exists\mathbb{R}$. The *existential theory of the reals* (see [16] for details) is the set of true sentences of the form $\exists x_1, \ldots, x_n \in \mathbb{R} \colon \phi(x_1, \ldots, x_n)$ for a quantifier-free Boolean formula ϕ without negation that can use the constants 0 and 1, as well as the operators $+$, $*$, and $<$. For example, $\exists x, y \colon (x < y) \wedge (x * y < 1 + 1)$ is such a formula. A problem is in the complexity class $\exists\mathbb{R}$ if it allows for a polynomial-time reduction to the problem of deciding such formulas, and it is complete if in addition every problem in $\exists\mathbb{R}$ can be reduced to it by a polynomial-time reduction.

For the art gallery problem, the result by Abrahamsen et al. [2] implies that the coordinates of an optimal guard set may be doubly-exponential in the input size. This statement does not exclude the possibility of a more concise, implicit way to express the existence of an optimal solution. However, if we found such a way, then this would imply that the art gallery problem is in NP, which, in turn, would imply NP $= \exists\mathbb{R}$.

Questions

8. *What is a simple polygon/a simple polygon with holes?* Explain the definitions and provide some examples of members and non-members of the respective classes. For a given polygon you should be able to tell which of these classes it belongs to or does not belong to and argue why this is the case.

9. *What is a closed/open/bounded set in $\mathbb{R}^d$? What is the interior/closure of a point set?* Explain the definitions and provide some illustrative examples. For a given set you should be able to argue which of the properties mentioned it possesses.
10. *What is a triangulation of a simple polygon? Does it always exist?* Explain the definition and provide some illustrative examples. Present the proof of Theorem 3.9 in detail.
11. *How about higher dimensional generalizations? Can every polyhedron in $\mathbb{R}^3$ be nicely subdivided into tetrahedra?* Explain Schönhardt's construction.
12. *How many points are needed to guard a simple polygon?* Present the proofs of Theorem 3.12, Lemma 3.19, and Theorem 3.20 in detail.
13. *Is there a compact representation for optimal guard placements?* State Theorem 3.22 and sketch the construction.

References

[1] Mikkel Abrahamsen, Anna Adamaszek, and Tillmann Miltzow, Irrational guards are sometimes needed. In *Proc. 33rd Internat. Sympos. Comput. Geom.*, pp. 3:1–3:15, 2017.

[2] Mikkel Abrahamsen, Anna Adamaszek, and Tillmann Miltzow, The art gallery problem is $\exists\mathbb{R}$-complete. In *Proc. 50th Annu. ACM Sympos. Theory Comput.*, pp. 65–73, 2018.

[3] Martin Aigner and Günter M. Ziegler, *Proofs from THE BOOK*, Springer-Verlag, Berlin, 3rd edn., 2003.

[4] Nancy M. Amato, Michael T. Goodrich, and Edgar A. Ramos, A randomized algorithm for triangulating a simple polygon in linear time. *Discrete Comput. Geom.*, **26**, 2, (2001), 245–265.

[5] Takao Asano, Tetsuo Asano, and Ron Y. Pinter, Polygon triangulation: efficiency and minimality. *J. Algorithms*, **7**, 2, (1986), 221–231.

[6] Alexander Below, Jesús A. De Loera, and Jürgen Richter-Gebert, The complexity of finding small triangulations of convex 3-polytopes. *J. Algorithms*, **50**, 2, (2004), 134–167.

[7] Bernard Chazelle, Triangulating a simple polygon in linear time. *Discrete Comput. Geom.*, **6**, 5, (1991), 485–524.

[8] Tobias Christ, Beyond triangulation: covering polygons with triangles. In *Proc. 12th Algorithms and Data Struct. Sympos.*, vol. 6844 of *Lecture Notes Comput. Sci.*, pp. 231–242, Springer-Verlag, 2011.

[9] Václav Chvátal, A combinatorial theorem in plane geometry. *J. Combin. Theory Ser. B*, **18**, 1, (1975), 39–41.

[10] Erik D. Demaine, Joseph S. B. Mitchell, and Joseph O'Rourke, The Open Problems Project, Problem #10. `http://cs.smith.edu/~orourke/TOPP/P10.html`.

[11] Steve Fisk, A short proof of Chvátal's watchman theorem. *J. Combin. Theory Ser. B*, **24**, 3, (1978), 374.

[12] Der-Tsai Lee and Arthur K. Lin, Computational complexity of art gallery problems. *IEEE Trans. Inform. Theory*, **32**, 2, (1986), 276–282.

[13] Gary H. Meisters, Polygons have ears. *Amer. Math. Monthly*, **82**, 6, (1975), 648–651.

[14] Gary H. Meisters, Principal vertices, exposed points, and ears. *Amer. Math. Monthly*, **87**, 4, (1980), 284–285.

[15] Jim Ruppert and Raimund Seidel, On the difficulty of triangulating three-dimensional nonconvex polyhedra. *Discrete Comput. Geom.*, **7**, (1992), 227–253.

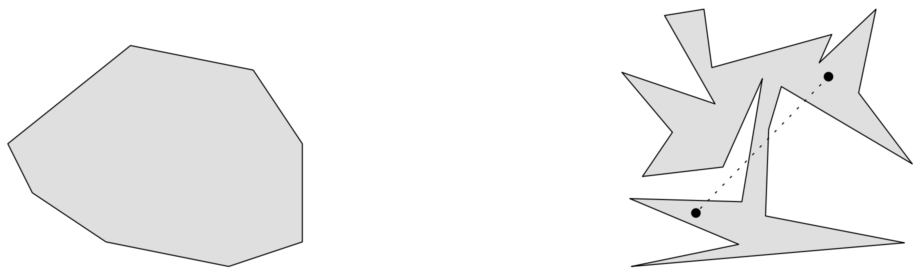
[16] Marcus Schaefer, Complexity of some geometric and topological problems. In *Proc. 17th Int. Sympos. Graph Drawing (GD 2009)*, pp. 334–344, 2009.

[17] Erich Schönhardt, Über die Zerlegung von Dreieckspolyedern in Tetraeder. *Math. Ann.*, **98**, (1928), 309–312.

Chapter 4

Convex Hull

There exists an incredible variety of point sets and polygons. Among them, some have certain properties that make them "nicer" than others in some respect. For instance, look at the two polygons shown below.



(a) A convex polygon. **(b)** A non-convex polygon.

Figure 4.1: *Examples of polygons: Which do you like better?*

As it is hard to argue about aesthetics, let us take a more algorithmic stance. When designing algorithms, the polygon shown on the left appears much easier to deal with than the visually and geometrically more complex polygon shown on the right. One particular property that makes the left polygon nice is that one can walk between any two vertices along a straight line without ever leaving the polygon. In fact, this statement holds true not only for vertices but for any two points within the polygon. A polygon or, more generally, a set with this property is called *convex*.

Definition 4.1. *A set* $P \subseteq \mathbb{R}^d$ *is* **convex** *if* $\overline{pq} \subseteq P$*, for any* $p, q \in P$.

An alternative, equivalent way to phrase convexity would be to demand that for every line $\ell \subset \mathbb{R}^d$ the intersection $\ell \cap P$ be connected. The polygon shown in Figure 4.1b is not convex because there are some pairs of points for which the connecting line segment is not completely contained within the polygon. An immediate consequence of the definition is the following

Observation 4.2. *For any family $(P_i)_{i\in I}$ of convex sets, the intersection $\bigcap_{i\in I} P_i$ is convex.*

Indeed there are many problems that are comparatively easy to solve for convex sets but very hard in general. We will encounter some particular instances of this phenomenon later in the course. However, not all polygons are convex and a discrete set of points is never convex, unless it consists of at most one point only. In such a case it is useful to make a given set P convex, that is, approximate P with or, rather, encompass P within a convex set $H \supseteq P$. Ideally, H differs from P as little as possible, that is, we want H to be a smallest convex set enclosing P.

At this point let us step back for a second and ask ourselves whether this wish makes sense at all: Does such a set H (always) exist? Fortunately, we are on the safe side because the whole space $\mathbb{R}^d$ is certainly convex. It is less obvious, but we will see below that H is actually unique. Therefore it is legitimate to refer to H as **the** smallest convex set enclosing P or—shortly—the *convex hull* of P.

4.1 Convexity

In this section we will derive an algebraic characterization of convexity. Such a characterization allows to investigate convexity using the machinery from linear algebra.

Consider $P \subset \mathbb{R}^d$. From linear algebra courses you should know that the *linear hull*

$$\operatorname{lin}(P) := \left\{ \sum_{i=1}^{n} \lambda_i p_i \;\middle|\; n \in \mathbb{N} \wedge \forall i \in \{1,\ldots,n\}\colon p_i \in P, \lambda_i \in \mathbb{R} \right\}$$

is the set of all *linear combinations* of P (smallest linear subspace containing P). For instance, if $P = \{p\} \subset \mathbb{R}^2 \setminus \{0\}$ then $\operatorname{lin}(P)$ is the line through p and the origin.

Similarly, the *affine hull*

$$\operatorname{aff}(P) := \left\{ \sum_{i=1}^{n} \lambda_i p_i \;\middle|\; n \in \mathbb{N} \wedge \forall i \in \{1,\ldots,n\}\colon p_i \in P, \lambda_i \in \mathbb{R} \wedge \sum_{i=1}^{n} \lambda_i = 1 \right\}$$

is the set of all *affine combinations* of P (smallest affine subspace containing P). For instance, if $P = \{p, q\} \subset \mathbb{R}^2$ and $p \neq q$ then $\operatorname{aff}(P)$ is the line through p and q.

It turns out that convexity can be described in a very similar way algebraically, which leads to the notion of *convex combinations*.

Proposition 4.3. *A set $P \subseteq \mathbb{R}^d$ is convex if and only if $\sum_{i=1}^{n} \lambda_i p_i \in P$, for all $n \in \mathbb{N}$, $p_1, \ldots, p_n \in P$, and $\lambda_1, \ldots, \lambda_n \geqslant 0$ with $\sum_{i=1}^{n} \lambda_i = 1$.*

Proof. "$\Leftarrow$": obvious with $n = 2$.

"$\Rightarrow$": Induction on n. For $n = 1$ the statement is trivial. For $n \geqslant 2$, let $p_i \in P$ and $\lambda_i \geqslant 0$, for $1 \leqslant i \leqslant n$, and assume $\sum_{i=1}^{n} \lambda_i = 1$. We may suppose that $\lambda_i > 0$, for all i. (Simply omit those points whose coefficient is zero.) We need to show that $\sum_{i=1}^{n} \lambda_i p_i \in P$.

Define $\lambda = \sum_{i=1}^{n-1} \lambda_i$ and for $1 \leqslant i \leqslant n-1$ set $\mu_i = \lambda_i/\lambda$. Observe that $\mu_i \geqslant 0$ and $\sum_{i=1}^{n-1} \mu_i = 1$. By the inductive hypothesis, $q := \sum_{i=1}^{n-1} \mu_i p_i \in P$, and thus by convexity of P also $\lambda q + (1-\lambda)p_n \in P$. We conclude by noting that $\lambda q + (1-\lambda)p_n = \lambda \sum_{i=1}^{n-1} \mu_i p_i + \lambda_n p_n = \sum_{i=1}^{n} \lambda_i p_i$. □

Definition 4.4. *The* **convex hull** $\text{conv}(P)$ *of a set* $P \subseteq \mathbb{R}^d$ *is the intersection of all convex supersets of* P.

At first glance this definition is a bit scary: There may be a whole lot of supersets for any given P and it is not clear that taking the intersection of all of them yields something sensible to work with. However, by Observation 4.2 we know that the resulting set is convex, at least. The missing bit is provided by the following proposition, which characterizes the convex hull in terms of exactly those convex combinations that appeared in Proposition 4.3 already.

Proposition 4.5. *For any* $P \subseteq \mathbb{R}^d$ *we have*

$$\text{conv}(P) = \left\{ \sum_{i=1}^{n} \lambda_i p_i \;\middle|\; n \in \mathbb{N} \wedge \sum_{i=1}^{n} \lambda_i = 1 \wedge \forall i \in \{1, \ldots, n\} : \lambda_i \geqslant 0 \wedge p_i \in P \right\}.$$

The elements of the set on the right hand side are referred to as *convex combinations* of P.

Proof. "$\supseteq$": Consider a convex set $C \supseteq P$. By Proposition 4.3 (only-if direction) the right hand side is contained in C. As C was arbitrary, the claim follows.

"$\subseteq$": Denote the set on the right hand side by R. Clearly $R \supseteq P$. We show that R forms a convex set. Let $p = \sum_{i=1}^{n} \lambda_i p_i$ and $q = \sum_{i=1}^{n} \mu_i p_i$ be two convex combinations. (We may suppose that both p and q are expressed over the same p_i by possibly adding some terms with a coefficient of zero.)

Then for $\lambda \in [0,1]$ we have $\lambda p + (1-\lambda)q = \sum_{i=1}^{n} (\lambda\lambda_i + (1-\lambda)\mu_i)p_i \in R$, as $\underbrace{\lambda\lambda_i}_{\geqslant 0} + \underbrace{(1-\lambda)}_{\geqslant 0}\underbrace{\mu_i}_{\geqslant 0} \geqslant 0$, for all $1 \leqslant i \leqslant n$, and $\sum_{i=1}^{n} (\lambda\lambda_i + (1-\lambda)\mu_i) = \lambda + (1-\lambda) = 1$. □

In linear algebra the notion of a basis in a vector space plays a fundamental role. In a similar way we want to describe convex sets using as few entities as possible, which leads to the notion of extremal points, as defined below.

Definition 4.6. *The convex hull of a finite point set* $P \subset \mathbb{R}^d$ *forms a* **convex polytope**. *Each* $p \in P$ *for which* $p \notin \text{conv}(P \setminus \{p\})$ *is called a* **vertex** *of* $\text{conv}(P)$. *A vertex of* $\text{conv}(P)$ *is also called an* **extremal point** *of* P. *A convex polytope in* $\mathbb{R}^2$ *is called a* **convex polygon**.

Essentially, the following proposition shows that the term vertex above is well defined.

Proposition 4.7. *A convex polytope in $\mathbb{R}^d$ is the convex hull of its vertices.*

Proof. Let $P = \{p_1, \ldots, p_n\}$, $n \in \mathbb{N}$, such that without loss of generality $p_1, \ldots, p_k$ are the vertices of $\mathcal{P} := \operatorname{conv}(P)$. We prove by induction on n that $\operatorname{conv}(p_1, \ldots, p_n) \subseteq \operatorname{conv}(p_1, \ldots, p_k)$. For $n = k$ the statement is trivial.

For $n > k$, p_n is not a vertex of $\mathcal{P}$ and hence p_n can be expressed as a convex combination $p_n = \sum_{i=1}^{n-1} \lambda_i p_i$. Thus for any $x \in \mathcal{P}$ we can write $x = \sum_{i=1}^{n} \mu_i p_i = \sum_{i=1}^{n-1} \mu_i p_i + \mu_n \sum_{i=1}^{n-1} \lambda_i p_i = \sum_{i=1}^{n-1} (\mu_i + \mu_n \lambda_i) p_i$. As $\sum_{i=1}^{n-1} (\mu_i + \mu_n \lambda_i) = 1$, we conclude inductively that $x \in \operatorname{conv}(p_1, \ldots, p_{n-1}) \subseteq \operatorname{conv}(p_1, \ldots, p_k)$. □

4.2 Classic Theorems for Convex Sets

Next we will discuss a few fundamental theorems about convex sets in $\mathbb{R}^d$. The proofs typically use the algebraic characterization of convexity and then employ some techniques from linear algebra.

Theorem 4.8 (Radon [9]). *Any set $P \subset \mathbb{R}^d$ of $d+2$ points can be partitioned into two disjoint subsets P_1 and P_2 such that* $\operatorname{conv}(P_1) \cap \operatorname{conv}(P_2) \neq \emptyset$.

Proof. Let $P = \{p_1, \ldots, p_{d+2}\}$. No more than $d+1$ points can be affinely independent in $\mathbb{R}^d$. Hence suppose without loss of generality that p_{d+2} can be expressed as an affine combination of $p_1, \ldots, p_{d+1}$, that is, there exist $\lambda_1, \ldots, \lambda_{d+1} \in \mathbb{R}$ with $\sum_{i=1}^{d+1} \lambda_i = 1$ and $\sum_{i=1}^{d+1} \lambda_i p_i = p_{d+2}$. Let P_1 be the set of all points p_i for which λ_i is positive and let $P_2 = P \setminus P_1$. Then setting $\lambda_{d+2} = -1$ we can write $\sum_{p_i \in P_1} \lambda_i p_i = \sum_{p_i \in P_2} -\lambda_i p_i$, where all coefficients on both sides are non-negative. Since $\sum_{i=1}^{i=d+2} \lambda_i = 0$ we have $s := \sum_{p_i \in P_1} \lambda_i = \sum_{p_i \in P_2} -\lambda_i$. Renormalizing by $\mu_i = \lambda_i / s$ and $\nu_i = \lambda_i / s$ yields convex combinations $\sum_{p_i \in P_1} \mu_i p_i = \sum_{p_i \in P_2} \nu_i p_i$ that describe a common point of $\operatorname{conv}(P_1)$ and $\operatorname{conv}(P_2)$. □

Theorem 4.9 (Helly). *Consider a collection $\mathcal{C} = \{C_1, \ldots, C_n\}$ of $n \geqslant d+1$ convex subsets of $\mathbb{R}^d$, such that any $d+1$ pairwise distinct sets from $\mathcal{C}$ have non-empty intersection. Then also the intersection $\bigcap_{i=1}^n C_i$ of all sets from $\mathcal{C}$ is non-empty.*

Proof. Induction on n. The base case $n = d+1$ holds by assumption. Hence suppose that $n \geqslant d+2$. Consider the sets $D_i = \bigcap_{j \neq i} C_j$, for $i \in \{1, \ldots, n\}$. As D_i is an intersection of $n-1$ sets from $\mathcal{C}$, by the inductive hypothesis we know that $D_i \neq \emptyset$. Therefore we can find some point $p_i \in D_i$, for each $i \in \{1, \ldots, n\}$. Now by Theorem 4.8 the set $P = \{p_1, \ldots, p_n\}$ can be partitioned into two disjoint subsets P_1 and P_2 such that $\operatorname{conv}(P_1) \cap \operatorname{conv}(P_2) \neq \emptyset$. We claim that any point $p \in \operatorname{conv}(P_1) \cap \operatorname{conv}(P_2)$ also lies in $\bigcap_{i=1}^n C_i$, which completes the proof.

Consider some C_i, for $i \in \{1, \ldots, n\}$. By construction $D_j \subseteq C_i$, for $j \neq i$. Thus p_i is the only point from P that may not be in C_i. As p_i is part of only one of P_1 or P_2, say, of P_1, we have $P_2 \subseteq C_i$. The convexity of C_i implies $\operatorname{conv}(P_2) \subseteq C_i$ and, therefore, $p \in C_i$. □

There is a nice application of Helly's theorem showing the existence of so-called centerpoints of finite point sets. Basically, a centerpoint is one possible generalization of the median of one-dimensional sets. (See, e.g., [5].)

Definition 4.10. *Let $P \subset \mathbb{R}^d$ be a set of n points. A point p, not necessarily in P, is a* centerpoint *of P if every open halfspace that contains more than $\frac{dn}{d+1}$ points of P also contains p.*

Stated differently, every closed halfspace containing a centerpoint contains at least $\frac{n}{d+1}$ points of P (which is clearly equivalent to containing at least $\lceil \frac{n}{d+1} \rceil$ points). We have the following result, which we prove similar to [8].

Theorem 4.11. *For every set $P \subset \mathbb{R}^d$ of n points there exists a centerpoint.*

Proof. We may assume that P contains at least $d+1$ affinely independent points (otherwise, a centerpoint can be found in a lower-dimensional affine sub-space).

Let $\mathcal{A}$ be the family of subsets of P that are defined by the intersection of P with an open halfspace, and that contain more than $\frac{dn}{d+1}$ points. Note that since P is finite, also the number of sets in $\mathcal{A} = \{A_1, \dots, A_m\}$ is finite. Let $C_i := \text{conv}(A_i)$. If there exists a point c that is in the intersection $\bigcap_{i=1}^m C_i$, then c is contained in any open halfspace that contains more than $\frac{dn}{d+1}$ points of P and thus is a centerpoint of P. We show the existence of c by showing that any $d+1$ elements of $\{C_1, \dots, C_m\}$ have a common point, and then applying Theorem 4.9.

For any $d+1$ sets in $\mathcal{A}$, suppose that any point of P occurs in at most d of these subsets. Then in total we would have at most dn occurrences of points in these subsets. However, by the choice of $\mathcal{A}$, each set contains more than $\frac{dn}{d+1}$ points, so the total number of occurrences is more than $(d+1)\frac{dn}{d+1} = dn$. Hence, any $d+1$ sets in $\mathcal{A}$ have a common point, and thus $\bigcap_{i=1}^m C_i$ also contains a point c. □

Exercise 4.12. *Show that the number of points in Definition 4.10 is best possible, that is, for every n there is a set of n points in $\mathbb{R}^d$ such that for any $p \in \mathbb{R}^d$ there is an open halfspace containing $\lfloor \frac{dn}{d+1} \rfloor$ points but not p.*

Theorem 4.13 (Carathéodory [3]). *For any $P \subset \mathbb{R}^d$ and $q \in \text{conv}(P)$ there exist $k \leqslant d+1$ points $p_1, \dots, p_k \in P$ such that $q \in \text{conv}(p_1, \dots, p_k)$.*

Exercise 4.14. *Prove Theorem 4.13.*

Theorem 4.15 (Separation Theorem). *Any two compact convex sets $C, D \subset \mathbb{R}^d$ with $C \cap D = \emptyset$ can be separated strictly by a hyperplane, that is, there exists a hyperplane h such that C and D lie in the opposite open halfspaces bounded by h.*

Proof. Consider the distance function $\delta : C \times D \to \mathbb{R}$ with $(c, d) \mapsto \|c - d\|$. Since $C \times D$ is compact and δ is continuous and strictly bounded from below by 0, the function δ attains its minimum at some point $(c_0, d_0) \in C \times D$ with $\delta(c_0, d_0) > 0$. Let h be the

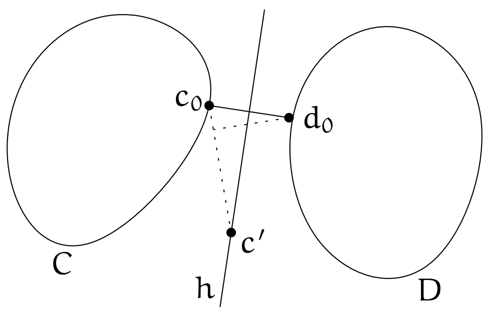


Figure 4.2: *The disjoint compact convex sets* C *and* D *have a separating hyperplane* h.

hyperplane perpendicular to the line segment $\overline{c_0 d_0}$ and passing through the midpoint of c_0 and d_0. (See Figure 4.2.)

If there was a point, say, c' in $C \cap h$, then by convexity of C the whole line segment $\overline{c_0 c'}$ lies in C and some point along this segment is closer to d_0 than is c_0, in contradiction to the choice of c_0. The figure shown to the right depicts the situation in $\mathbb{R}^2$. If, say, C has points on both sides of h, then by convexity of C it has also a point on h, but we just saw that there is no such point. Therefore, C and D must lie in different open halfspaces bounded by h. □

The statement above is wrong for arbitrary (not necessarily compact) convex sets. However, if the separation is not required to be strict (the hyperplane may intersect the sets), then such a separation always exists, with the proof being a bit more involved (cf. [8], but also check the errata on Matoušek's webpage).

Exercise 4.16. *Show that the Separation Theorem does not hold in general, if not both of the sets are convex.*

Exercise 4.17. *Prove or disprove:*

a) The convex hull of a compact subset of $\mathbb{R}^d$ *is compact.*

b) The convex hull of a closed subset of $\mathbb{R}^d$ *is closed.*

Altogether we obtain various equivalent definitions for the convex hull, summarized in the following theorem.

Theorem 4.18. *For a compact set* $P \subset \mathbb{R}^d$ *we can characterize* conv(P) *equivalently as one of*

1. *the smallest (w. r. t. set inclusion) convex subset of* $\mathbb{R}^d$ *that contains* P;
2. *the set of all convex combinations of points from* P;
3. *the set of all convex combinations formed by* $d+1$ *or fewer points from* P;
4. *the intersection of all convex supersets of* P;
5. *the intersection of all closed halfspaces containing* P.

Exercise 4.19. *Prove Theorem 4.18.*

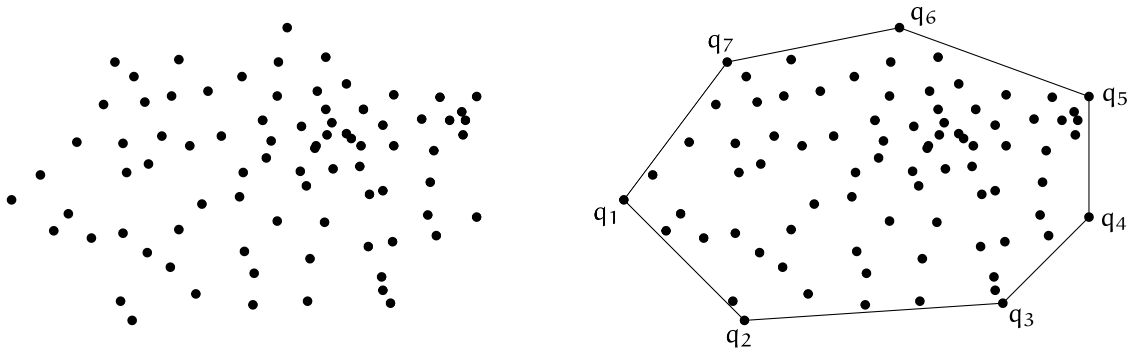
4.3 Planar Convex Hull

Although we know by now what is the convex hull of a point set, it is not yet clear how to construct it algorithmically. As a first step, we have to find a suitable representation for convex hulls. In this section we focus on the problem in $\mathbb{R}^2$, where the convex hull of a finite point set forms a convex polygon. A convex polygon is easy to represent, for instance, as a sequence of its vertices in counterclockwise orientation. In higher dimensions finding a suitable representation for convex polytopes is a much more delicate task.

Problem 4.20 (Convex hull).

Input: $P = \{p_1, \dots, p_n\} \subset \mathbb{R}^2$, $n \in \mathbb{N}$.

Output: Sequence $(q_1, \dots, q_h)$, $1 \leqslant h \leqslant n$, of the vertices of $\operatorname{conv}(P)$ (ordered counterclockwise).



(a) Input. **(b)** Output.

Figure 4.3: *Convex Hull of a set of points in* $\mathbb{R}^2$.

Another possible algorithmic formulation of the problem is to ignore the structure of the convex hull and just consider it as a point set.

Problem 4.21 (Extremal points).

Input: $P = \{p_1, \dots, p_n\} \subset \mathbb{R}^2$, $n \in \mathbb{N}$.

Output: Set $Q \subseteq P$ of the vertices of $\operatorname{conv}(P)$.

Degeneracies. A couple of further clarifications regarding the above problem definitions are in order.

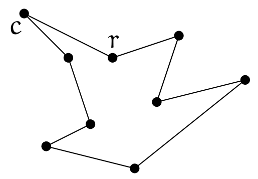
First of all, for efficiency reasons an input is usually specified as a sequence of points. Do we insist that this sequence forms a set or are duplications of points allowed?

What if three points are collinear? Are all of them considered extremal? According to our definition from above, they are not and that is what we will stick to. But note that there may be cases where one wants to include all such points, nevertheless.

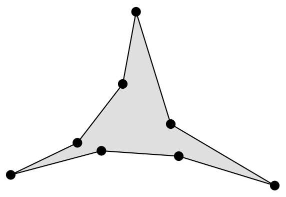
By the Separation Theorem, every extremal point p can be separated from the convex hull of the remaining points by a halfplane. If we take such a halfplane and translate its defining line such that it passes through p, then all points from P other than p should lie in the resulting open halfplane. In $\mathbb{R}^2$ it turns out convenient to work with the following "directed" reformulation.

Proposition 4.22. *A point* $p \in P = \{p_1, \dots, p_n\} \subset \mathbb{R}^2$ *is* **extremal** *for* $P \iff$ *there is a directed line* g *through* p *such that* $P \setminus \{p\}$ *is (strictly) to the left of* g.

The *interior angle* at a vertex v of a polygon P is the angle between the two edges of P incident to v whose corresponding angular domain lies in P°. If this angle is smaller than π, the vertex is called *convex*; if the angle is larger than π, the vertex is called *reflex*. For instance, the vertex c in the polygon depicted to the right is a convex vertex, whereas the vertex labeled r is a reflex vertex.



Exercise 4.23.
A set $S \subset \mathbb{R}^2$ *is* star-shaped *if there exists a point* $c \in S$*, such that for every point* $p \in S$ *the line segment* $\overline{cp}$ *is contained in* S*. A simple polygon with exactly three convex vertices is called a pseudotriangle (see the example shown on the right).*



In the following we consider subsets of $\mathbb{R}^2$*. Prove or disprove:*

a) Every convex vertex of a simple polygon lies on its convex hull.

b) Every star-shaped set is convex.

c) Every convex set is star-shaped.

d) The intersection of two convex sets is convex.

e) The union of two convex sets is convex.

f) The intersection of two star-shaped sets is star-shaped.

g) The intersection of a convex set with a star-shaped set is star-shaped.

h) Every triangle is a pseudotriangle.

i) Every pseudotriangle is star-shaped.

4.4 Trivial algorithms

One can compute the extremal points using Carathéodory's Theorem as follows: Test for every point $p \in P$ whether there are $q, r, s \in P \setminus \{p\}$ such that p is inside the triangle with vertices q, r, and s. Runtime $O(n^4)$.

Another option, inspired by the Separation Theorem: test for every pair $(p, q) \in P^2$ whether all points from $P \setminus \{p, q\}$ are to the left of the directed line through p and q (or on the line segment $\overline{pq}$). Runtime $O(n^3)$.

Exercise 4.24. *Let $P = (p_0, \ldots, p_{n-1})$ be a sequence of n points in $\mathbb{R}^2$. Someone claims that you can check by means of the following algorithm whether or not P describes the boundary of a convex polygon in counterclockwise order:*

```
bool is_convex($p_0, \ldots, p_{n-1}$) {
    for $i = 0, \ldots, n-1$:
        if ($p_i$, $p_{(i+1) \bmod n}$, $p_{(i+2) \bmod n}$) form a rightturn:
            return false;
    return true;
}
```

Disprove the claim and describe a correct algorithm to solve the problem.

Exercise 4.25. *Let $P \subset \mathbb{R}^2$ be a convex polygon, given as an array* `p[0]...p[n-1]` *of its n vertices in counterclockwise order.*

a) Describe an $O(\log(n))$ time algorithm to determine whether a point q lies inside, outside or on the boundary of P.

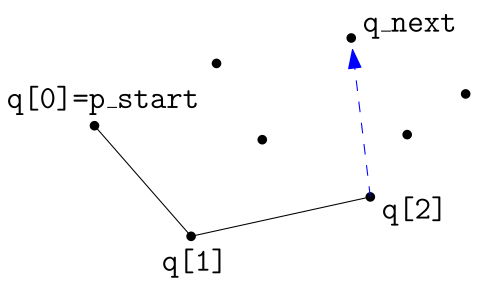
b) Describe an $O(\log(n))$ time algorithm to find a (right) tangent to P from a query point q located outside P. That is, find a vertex $p[i]$, such that P is contained in the closed halfplane to the left of the oriented line $qp[i]$.

4.5 Jarvis' Wrap

We are now ready to describe a first simple algorithm to construct the convex hull. It is inspired by Proposition 4.22 and works as follows:

> Find a point p_1 that is a vertex of $\text{conv}(P)$ (e.g., the one with smallest x-coordinate). "Wrap" P starting from p_1, i.e., always find the next vertex of $\text{conv}(P)$ as the one that is rightmost with respect to the direction given by the previous two vertices.

Besides comparing x-coordinates, the only geometric primitive needed is an *orientation* test: Denote by `rightturn`(p, q, r), for three points $p, q, r \in \mathbb{R}^2$, the predicate that is true if and only if r is (strictly) to the right of the oriented line pq.



Code for Jarvis' Wrap.

`p[0..N)` contains a sequence of N points.
`p_start` point with smallest x-coordinate.
`q_next` some *other* point in `p[0..N)`.

```
int h = 0;
Point_2 q_now = p_start;
do {
  q[h] = q_now;
  h = h + 1;

  for (int i = 0; i < N; i = i + 1)
    if (rightturn_2(q_now, q_next, p[i]))
      q_next = p[i];

  q_now = q_next;
  q_next = p_start;
} while (q_now != p_start);
```

`q[0,h)` describes a convex polygon bounding the convex hull of `p[0..N)`.

Analysis. For every output point the above algorithm spends n rightturn tests, which is $\Rightarrow O(nh)$ in total.

Theorem 4.26. [7] *Jarvis' Wrap computes the convex hull of n points in $\mathbb{R}^2$ using $O(nh)$ rightturn tests, where h is the number of hull vertices.*

In the worst case we have $h = n$, that is, $O(n^2)$ rightturn tests. Jarvis' Wrap has a remarkable property that is called *output sensitivity*: the runtime depends not only on the size of the input but also on the size of the output. For a huge point set it constructs the convex hull in optimal linear time, if the convex hull consists of a constant number of vertices only. Unfortunately the worst case performance of Jarvis' Wrap is suboptimal, as we will see soon.

Degeneracies. The algorithm may have to cope with various degeneracies.

- Several points have smallest x-coordinate $\Rightarrow$ lexicographic order:

$$(p_x, p_y) < (q_x, q_y) \iff p_x < q_x \vee p_x = q_x \wedge p_y < q_y .$$

- Three or more points collinear $\Rightarrow$ choose the point that is farthest among those that are rightmost.

Predicates. Besides the lexicographic comparison mentioned above, the Jarvis' Wrap (and most other 2D convex hull algorithms for that matter) need one more geometric predicate: the rightturn or—more generally—orientation test. The computation amounts to evaluating a polynomial of degree two, see the exercise below. We therefore say that the orientation test has *algebraic degree* two. In contrast, the lexicographic comparison has degree one only. The algebraic degree not only has a direct impact on the efficiency of a geometric algorithm (lower degree $\leftrightarrow$ less multiplications), but also an indirect one because high degree predicates may create large intermediate results, which may lead to overflows and are much more costly to compute with exactly.

Exercise 4.27. *Prove that for three points $(p_x, p_y), (q_x, q_y), (r_x, r_y) \in \mathbb{R}^2$, the sign of the determinant*

$$\begin{vmatrix} 1 & p_x & p_y \\ 1 & q_x & q_y \\ 1 & r_x & r_y \end{vmatrix}$$

determines if r lies to the right, to the left or on the directed line through p and q.

Exercise 4.28. *The InCircle predicate is: Given three points $p, q, r \in \mathbb{R}^2$ that define a circle C and a fourth point s, is s located inside C or not? The goal of this exercise is to derive an algebraic formulation of the incircle predicate in form of a determinant, similar to the formulation of the orientation test given above in Exercise 4.27. To this end we employ the so-called parabolic lifting map, which will also play a prominent role in the next chapter of the course.*

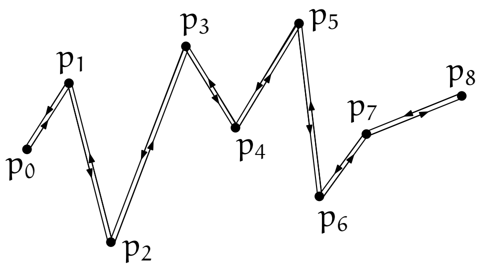
The parabolic lifting map $\ell : \mathbb{R}^2 \to \mathbb{R}^3$ *is defined for a point $p = (x, y) \in \mathbb{R}^2$ by $\ell(p) = (x, y, x^2 + y^2)$. For a circle $C \subseteq \mathbb{R}^2$ of positive radius, show that the "lifted circle" $\ell(C) = \{\ell(p) \mid p \in C\}$ is contained in a unique plane $h_C \subseteq \mathbb{R}^3$. Moreover, show that a point $p \in \mathbb{R}^2$ is strictly inside (outside, respectively) of C if and only if the lifted point $\ell(p)$ is strictly below (above, respectively) h_C.*

Use these insights to formulate the InCircle predicate for given points (p_x, p_y), (q_x, q_y), (r_x, r_y), $(s_x, s_y) \in \mathbb{R}^2$ as a determinant.

4.6 Graham Scan (Successive Local Repair)

There exist many algorithms that exhibit a better worst-case runtime than Jarvis' Wrap. Here we discuss only one of them: a particularly elegant and easy-to-implement variant of the so-called *Graham Scan* [6]. This algorithm is referred to as *Successive Local Repair* because it starts with some polygon enclosing all points and then step-by-step repairs the deficiencies of this polygon, by removing nonconvex vertices. It goes as follows:

Sort the points lexicographically to obtain a sequence $p_0, \ldots, p_{n-1}$ and build a corresponding circular sequence $p_0, \ldots, p_{n-1}, \ldots, p_0$ that walks around the point set in anticlockwise direction.



$p_0\, p_1\, p_2\, p_3\, p_4\, p_5\, p_6\, p_7\, p_8\, p_7\, p_6\, p_5\, p_4\, p_3\, p_2\, p_1\, p_0$

As long as there is a (consecutive) triple (p, q, r) such that r is to the right of or on the directed line $\overrightarrow{pq}$, remove q from the sequence.

Code for Graham Scan.

p[0..N) lexicographically sorted sequence of pairwise distinct points, $N \geqslant 2$.

```
q[0] = p[0];
int h = 0;
// Lower convex hull (left to right):
for (int i = 1; i < N; i = i + 1) {
  while (h>0 && !leftturn_2(q[h-1], q[h], p[i]))
    h = h - 1;
  h = h + 1;
  q[h] = p[i];
}

// Upper convex hull (right to left):
for (int i = N-2; i >= 0; i = i - 1) {
  while (!leftturn_2(q[h-1], q[h], p[i]))
    h = h - 1;
  h = h + 1;
  q[h] = p[i];
}
```

q[0,h) describes a convex polygon bounding the convex hull of p[0..N).

Correctness. We argue for the lower convex hull only. The argument for the upper hull is symmetric. A point p is on the lower convex hull of P if there is a rightward directed line g through p such that $P \setminus \{p\}$ is (strictly) to the left of g. A directed line is *rightward* if it forms an absolute angle of at most π with the positive x-axis. (Compare this statement with the one in Proposition 4.22.)

First, we claim that every point that the algorithm discards does not appear on the lower convex hull. A point q_h is discarded only if there exist points q_{h-1} and p_i with $q_{h-1} < q_h < p_i$ (lexicographically) so that $q_{h-1}q_hp_i$ does not form a leftturn. Thus, for every rightward directed line g through q_h at least one of q_{h-1} or p_i lies on or to the right of g. It follows that q_h is not on the lower convex hull, as claimed.

At the end of the (lower hull part of the) algorithm, in the sequence $q_0, \ldots, q_{h-1}$ every consecutive triple $q_iq_{i+1}q_{i+2}$, for $0 \leqslant i \leqslant h-3$, forms a leftturn with $q_i < q_{i+1} < q_{i+2}$. Thus, for every such triple there exists a rightward directed line g through q_{i+1} such that $P \setminus \{p\}$ is (strictly) to the left of g (for instance, take g to be perpendicular to the angular bisector of $\angle q_{i+2}q_{i+1}q_i$). It follows that every inner point of the sequence $q_0, \ldots, q_{h-1}$ is on the lower convex hull. The extreme points q_0 and q_{h-1} are the lexicographically smallest and largest, respectively, point of P, both of which are easily seen to be on the lower convex hull as well. Therefore, $q_0, \ldots, q_{h-1}$ form the lower convex hull of P, which proves the correctness of the algorithm.

Analysis.

Theorem 4.29. *The convex hull of a set $P \subset \mathbb{R}^2$ of n points can be computed using $O(n \log n)$ geometric operations.*

Proof.

1. Sorting and removal of duplicate points: $O(n \log n)$.

2. At the beginning we have a sequence of $2n - 1$ points; at the end the sequence consists of h points. Observe that for every positive orientation test, one point is discarded from the sequence for good. Therefore, we have exactly $2n - h - 1$ such shortcuts/positive orientation tests. In addition there are at most $2n - 2$ negative tests (#iterations of the outer `for` loops). Altogether we have at most $4n - h - 3$ orientation tests.

In total the algorithm uses $O(n \log n)$ geometric operations. Note that the number of orientation tests is linear only, but $O(n \log n)$ lexicographic comparisons are needed. □

4.7 Lower Bound

It is not hard to see that the runtime of Graham Scan is asymptotically optimal in the worst-case.

Theorem 4.30. $\Omega(n \log n)$ *geometric operations are needed to construct the convex hull of n points in $\mathbb{R}^2$ (in the algebraic computation tree model).*

Proof. Reduction from sorting (for which it is known that $\Omega(n \log n)$ comparisons are needed in the algebraic computation tree model). Given n real numbers $x_1, \ldots, x_n$, construct a set $P = \{p_i \mid 1 \leqslant i \leqslant n\}$ of n points in $\mathbb{R}^2$ by setting $p_i = (x_i, x_i^2)$. This construction can be regarded as embedding the numbers into $\mathbb{R}^2$ along the x-axis and then projecting the resulting points vertically onto the unit parabola. The order in which the points appear along the lower convex hull of P corresponds to the sorted order of the x_i. Therefore, if we could construct the convex hull in $o(n \log n)$ time, we could also sort in $o(n \log n)$ time. □

Clearly this reduction does not work for the Extremal Points problem. But using a reduction from Element Uniqueness (see Section 1.1) instead, one can show that $\Omega(n \log n)$ is also a lower bound for the number of operations needed to compute the set of extremal points only. This was first shown by Avis [1] for linear computation trees, then by Yao [10] for quadratic computation trees, and finally by Ben-Or [2] for general algebraic computation trees.

4.8 Chan's Algorithm

Given matching upper and lower bounds we may be tempted to consider the algorithmic complexity of the planar convex hull problem settled. However, this is not really the case: Recall that the lower bound is a worst case bound. For instance, the Jarvis' Wrap runs in $O(nh)$ time an thus beats the $\Omega(n \log n)$ bound in case that $h = o(\log n)$. The question remains whether one can achieve both output dependence and optimal worst case performance at the same time. Indeed, Chan [4] presented an algorithm to achieve this runtime by cleverly combining the "best of" Jarvis' Wrap and Graham Scan. Let us look at this algorithm in detail. The algorithm consists of two steps that are executed one after another.

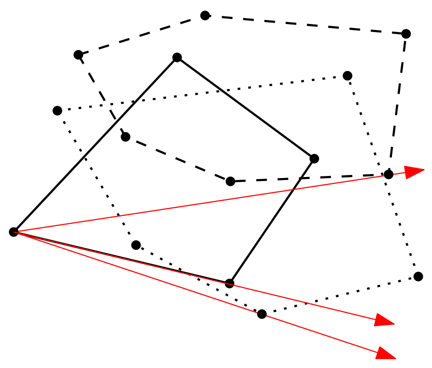
Divide. *Input:* a set $P \subset \mathbb{R}^2$ of n points and a number $H \in \{1, \ldots, n\}$.

1. Divide P into $k = \lceil n/H \rceil$ sets $P_1, \ldots, P_k$ with $|P_i| \leqslant H$.
2. Construct $\mathrm{conv}(P_i)$ for all i, $1 \leqslant i \leqslant k$.

Analysis. Step 1 takes $O(n)$ time. Step 2 can be handled using Graham Scan in $O(H \log H)$ time for any single P_i, that is, $O(n \log H)$ time in total.

Conquer. *Output:* the vertices of $\mathrm{conv}(P)$ in counterclockwise order, if $\mathrm{conv}(P)$ has less than H vertices; otherwise, the message that $\mathrm{conv}(P)$ has at least H vertices.

1. Find the lexicographically smallest point $p_<$ in P.
2. Starting from $p_<$ find the first H points of $\mathrm{conv}(P)$ oriented counterclockwise (simultaneous Jarvis' Wrap on the sequences $\mathrm{conv}(P_i)$).



Determine in every wrap step the point q_i of tangency from the current point of $\text{conv}(P)$ to $\text{conv}(P_i)$, for all $1 \leqslant i \leqslant k$. We have seen in Exercise 4.25 how to compute q_i in $O(\log |\text{conv}(P_i)|) = O(\log H)$ time. Among the k candidates $q_1, \ldots, q_k$ we find the next vertex of $\text{conv}(P)$ in $O(k)$ time.

Analysis. Step 1 takes $O(n)$ time. Step 2 consists of at most H wrap steps. Each wrap step needs $O(k \log H + k) = O(k \log H)$ time, which amounts to $O(Hk \log H) = O(n \log H)$ time for Step 2 in total.

Remark. Using a more clever search strategy instead of many tangency searches one can handle the conquer phase in $O(n)$ time, see Exercise 4.31 below. However, this is irrelevant as far as the asymptotic runtime is concerned, given that already the divide step takes $O(n \log H)$ time.

Exercise 4.31. *Consider k convex polygons $P_1, \ldots P_k$, for some constant $k \in \mathbb{N}$, where each polygon is given as a list of its vertices in counterclockwise orientation. Show how to construct the convex hull of $P_1 \cup \ldots \cup P_k$ in $O(n)$ time, where $n = \sum_{i=1}^{k} n_i$ and n_i is the number of vertices of P_i, for $1 \leqslant i \leqslant k$.*

Searching for h. While the runtime bound for $H = h$ is exactly what we were heading for, it looks like in order to actually run the algorithm we would have to know h, which—in general—we do not. Fortunately we can circumvent this problem rather easily, by applying what is called a *doubly exponential search.* It works as follows.

Call the algorithm from above iteratively with parameter $H = \min\{2^{2^t}, n\}$, for $t = 0, \ldots,$ until the conquer step finds all extremal points of P (i.e., the wrap returns to its starting point).

Analysis: Let 2^{2^s} be the last parameter for which the algorithm is called. Since the previous call with $H = 2^{2^{s-1}}$ did not find all extremal points, we know that $2^{2^{s-1}} < h$, that is, $2^{s-1} < \log h$, where h is the number of extremal points of P. The total runtime is therefore at most

$$\sum_{i=0}^{s} cn \log 2^{2^i} = cn \sum_{i=0}^{s} 2^i = cn(2^{s+1} - 1) < 4cn \log h = O(n \log h),$$

for some constant $c \in \mathbb{R}$. In summary, we obtain the following theorem.

Theorem 4.32. *The convex hull of a set $P \subset \mathbb{R}^2$ of n points can be computed using $O(n \log h)$ geometric operations, where h is the number of convex hull vertices.*

Questions

14. *How is convexity defined? What is the convex hull of a set in $\mathbb{R}^d$?* Give at least three possible definitions and show that they are equivalent.

15. *What is a centerpoint of a finite point set in* $\mathbb{R}^d$*?* State and prove the centerpoint theorem and the two classic theorems used in its proof (Theorem 4.11 along with Theorem 4.9 and Theorem 4.8).
16. *What does it mean to compute the convex hull of a set of points in* $\mathbb{R}^2$*?* Discuss input and expected output and possible degeneracies.
17. *How can the convex hull of a set of* n *points in* $\mathbb{R}^2$ *be computed efficiently?* Describe and analyze (incl. proofs) Jarvis' Wrap, Successive Local Repair, and Chan's Algorithm.
18. *Is there a linear time algorithm to compute the convex hull of* n *points in* $\mathbb{R}^2$*?* Prove the lower bound and define/explain the model in which it holds.
19. *Which geometric primitive operations are used to compute the convex hull of* n *points in* $\mathbb{R}^2$*?* Explain the two predicates and how to compute them.

References

[1] David Avis, Comments on a lower bound for convex hull determination. *Inform. Process. Lett.*, **11**, 3, (1980), 126.

[2] Michael Ben-Or, Lower bounds for algebraic computation trees. In *Proc. 15th Annu. ACM Sympos. Theory Comput.*, pp. 80–86, 1983.

[3] Constantin Carathéodory, Über den Variabilitätsbereich der Fourierschen Konstanten von positiven harmonischen Funktionen. *Rendiconto del Circolo Matematico di Palermo*, **32**, (1911), 193–217.

[4] Timothy M. Chan, Optimal output-sensitive convex hull algorithms in two and three dimensions. *Discrete Comput. Geom.*, **16**, 4, (1996), 361–368.

[5] Herbert Edelsbrunner, *Algorithms in combinatorial geometry*, vol. 10 of *EATCS Monographs on Theoretical Computer Science*, Springer, 1987.

[6] Ronald L. Graham, An efficient algorithm for determining the convex hull of a finite planar set. *Inform. Process. Lett.*, **1**, 4, (1972), 132–133.

[7] Ray A. Jarvis, On the identification of the convex hull of a finite set of points in the plane. *Inform. Process. Lett.*, **2**, 1, (1973), 18–21.

[8] Jiří Matoušek, *Lectures on discrete geometry*, Springer-Verlag, New York, NY, 2002.

[9] Johann Radon, Mengen konvexer Körper, die einen gemeinsamen Punkt enthalten. *Math. Annalen*, **83**, 1–2, (1921), 113–115.

[10] Andrew C. Yao, A lower bound to finding convex hulls. *J. ACM*, **28**, 4, (1981), 780–787.

Chapter 5

Delaunay Triangulations

In Chapter 3 we have discussed triangulations of simple polygons. A triangulation nicely partitions a polygon into triangles, which allows, for instance, to easily compute the area or a guarding of the polygon. Another typical application scenario is to use a triangulation T for interpolation: Suppose a function f is defined on the vertices of the polygon P, and we want to extend it "reasonably" and continuously to P°. Then for a point $p \in P^\circ$ find a triangle t of T that contains p. As p can be written as a convex combination $\sum_{i=1}^{3} \lambda_i v_i$ of the vertices v_1, v_2, v_3 of t, we just use the same coefficients to obtain an interpolation $f(p) := \sum_{i=1}^{3} \lambda_i f(v_i)$ of the function values.

If triangulations are a useful tool when working with polygons, they might also turn out useful to deal with other geometric objects, for instance, point sets. But what could be a triangulation of a point set? Polygons have a clearly defined interior, which naturally lends itself to be covered by smaller polygons such as triangles. A point set does not have an interior, except ... Here the notion of convex hull comes handy, because it allows us to treat a point set as a convex polygon. Actually, not really a convex polygon, because points in the interior of the convex hull should not be ignored completely. But one way to think of a point set is as a convex polygon—its convex hull—possibly with some holes—which are points—in its interior. A triangulation should then partition the convex hull while respecting the points in the interior, as shown in the example in Figure 5.1b.

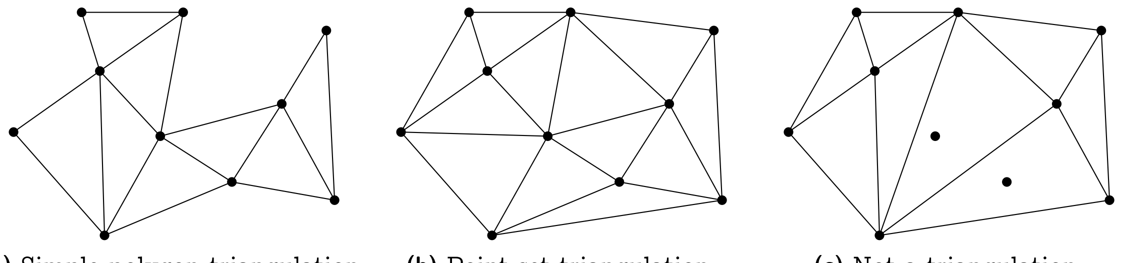

(a) Simple polygon triangulation. **(b)** Point set triangulation. **(c)** Not a triangulation.

Figure 5.1: *Examples of (non-)triangulations.*

In contrast, the example depicted in Figure 5.1c nicely subdivides the convex hull

but should not be regarded a triangulation: Two points in the interior are not respected but simply swallowed by a large triangle.
This interpretation directly leads to the following adaption of Definition 3.7.

Definition 5.1. *A* **triangulation** *of a finite point set* $P \subset \mathbb{R}^2$ *is a collection* $\mathcal{T}$ *of triangles, such that*

(1) $\operatorname{conv}(P) = \bigcup_{T \in \mathcal{T}} T$;

(2) $P = \bigcup_{T \in \mathcal{T}} V(T)$; *and*

(3) for every distinct pair $T, U \in \mathcal{T}$, *the intersection* $T \cap U$ *is either a common vertex, or a common edge, or empty.*

Just as for polygons, triangulations are universally available for point sets, meaning that (almost) every point set admits at least one.

Proposition 5.2. *Every set* $P \subseteq \mathbb{R}^2$ *of* $n \geqslant 3$ *points has a triangulation, unless all points in* P *are collinear.*

Proof. In order to construct a triangulation for P, consider the lexicographically sorted sequence $p_1, \ldots, p_n$ of points in P. Let m be minimal such that $p_1, \ldots, p_m$ are not collinear. We triangulate $p_1, \ldots, p_m$ by connecting p_m to all of $p_1, \ldots, p_{m-1}$ (which *are* on a common line), see Figure 5.2a.

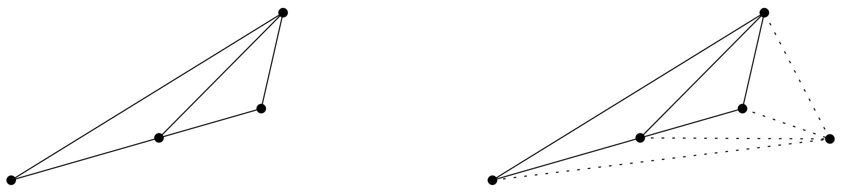
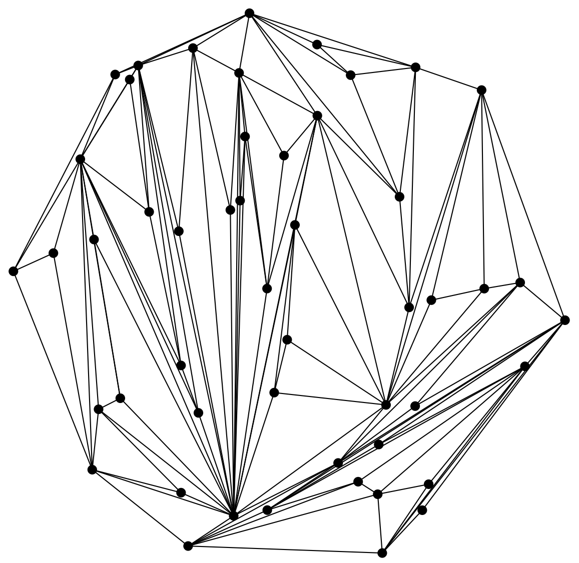

(a) Getting started. **(b)** Adding a point.

Figure 5.2: *Constructing the scan triangulation of* P.

Then we add $p_{m+1}, \ldots, p_n$. When adding p_i, for $i > m$, we connect p_i with all vertices of $C_{i-1} := \operatorname{conv}(\{p_1, \ldots, p_{i-1}\})$ that it "sees", that is, every vertex v of C_{i-1} for which $\overline{p_i v} \cap C_{i-1} = \{v\}$. In particular, among these vertices are the two points of tangency from p_i to C_{i-1}, which shows that we always add triangles (Figure 5.2b) whose union after each step covers C_i. $\square$

The triangulation that is constructed in Proposition 5.2 is called a *scan triangulation.* Such a triangulation (Figure 5.3a (left) shows a larger example) is usually "ugly", though, since it tends to have many long and skinny triangles. This is not just an aesthetic deficit. Having long and skinny triangles means that the vertices of a triangle tend to be spread out far from each other. You can probably imagine that such a behavior is undesirable,


(a) Scan triangulation.

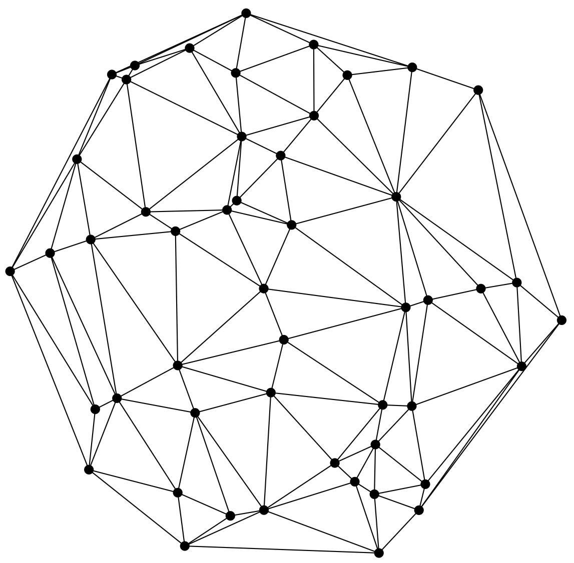

(b) Delaunay triangulation.

Figure 5.3: *Two triangulations of the same set of* 50 *points.*

for instance, in the context of interpolation. In contrast, the *Delaunay triangulation* of the same point set (Figure 5.3b) looks much nicer, and we will discuss in the next section how to get this triangulation.

Exercise 5.3. *Describe an* $O(n \log n)$ *time algorithm to construct a scan triangulation for a set of* n *points in* $\mathbb{R}^2$.

On another note, if you look closely into the SLR-algorithm to compute planar convex hulls that was discussed in Chapter 4, then you will realize that we also could have used this algorithm in the proof of Proposition 5.2. Whenever a point is discarded during SLR, a triangle is added to the polygon that eventually becomes the convex hull.

In view of the preceding chapter, we may regard a triangulation as a plane graph: the vertices are the points in P and there is an edge between two points $p \neq q$, if and only if there is a triangle with vertices p and q. Therefore we can use Euler's formula to determine the number of edges in a triangulation.

Lemma 5.4. *Any triangulation of a set* $P \subset \mathbb{R}^2$ *of* n *points has exactly* $3n - h - 3$ *edges, where* h *is the number of points from* P *on* $\partial\text{conv}(P)$.

Proof. Consider a triangulation T of P and denote by E the set of edges and by F the set of faces of T. We count the number of edge-face incidences in two ways. Denote $\mathcal{I} = \{(e, f) \in E \times F : e \subset \partial f\}$.

On the one hand, every edge is incident to exactly two faces and therefore $|\mathcal{I}| = 2|E|$. On the other hand, every bounded face of T is a triangle and the unbounded face has h edges on its boundary. Therefore, $|\mathcal{I}| = 3(|F| - 1) + h$.

Together we obtain $3|F| = 2|E| - h + 3$. Using Euler's formula ($3n - 3|E| + 3|F| = 6$) we conclude that $3n - |E| - h + 3 = 6$ and so $|E| = 3n - h - 3$. □

In graph theory, the term "triangulation" is sometimes used as a synonym for "maximal planar". But geometric triangulations are different, they are maximal planar in the sense that no straight-line edge can be added without sacrificing planarity.

Corollary 5.5. *A triangulation of a set* $P \subset \mathbb{R}^2$ *of* n *points is maximal planar, if and only if* $\text{conv}(P)$ *is a triangle.*

Proof. Combine Corollary 2.5 and Lemma 5.4. □

Exercise 5.6. *Find for every* $n \geqslant 3$ *a simple polygon* P *with* n *vertices such that* P *has exactly one triangulation.* P *should be in general position, meaning that no three vertices are collinear.*

Exercise 5.7. *Show that every set of* $n \geqslant 5$ *points in general position (no three points are collinear) has at least two different triangulations.*
Hint: Show first that every set of five points in general position contains a convex 4*-hole, that is, a subset of four points that span a convex quadrilateral that does not contain the fifth point.*

5.1 The Empty Circle Property

We will now move on to study the ominous and supposedly nice Delaunay triangulations mentioned above. They are defined in terms of an empty circumcircle property for triangles. The *circumcircle* of a triangle is the unique circle passing through the three vertices of the triangle, see Figure 5.4.

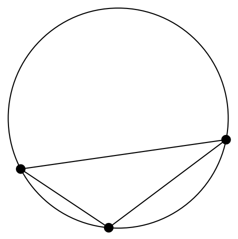


Figure 5.4: *Circumcircle of a triangle.*

Definition 5.8. *A triangulation of a finite point set* $P \subset \mathbb{R}^2$ *is called a* **Delaunay triangulation**, *if the circumcircle of every triangle is empty, that is, there is no point from* P *in its interior.*

Consider the example depicted in Figure 5.5. It shows a Delaunay triangulation of a set of six points: The circumcircles of all five triangles are empty (we also say that the

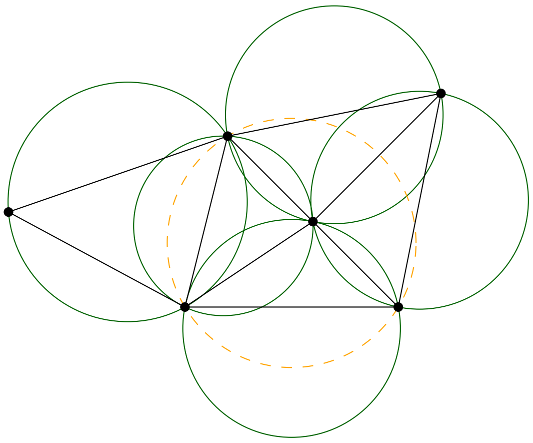
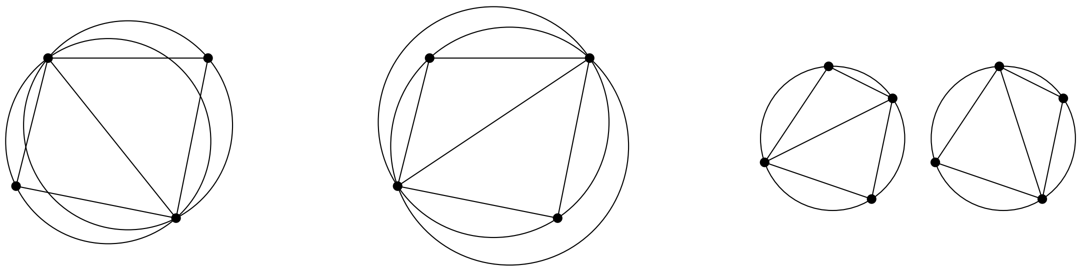


Figure 5.5: *All triangles satisfy the empty circle property.*

triangles satisfy the empty circle property). The dashed circle is not empty, but that is fine, since it is not a circumcircle of any triangle.

It is instructive to look at the case of four points in convex position. Obviously, there are two possible triangulations, but in general, only one of them will be Delaunay, see Figure 5.6a and 5.6b. If the four points are on a common circle, though, this circle is empty; at the same time it is the circumcircle of *all* possible triangles; therefore, both triangulations of the point set are Delaunay, see Figure 5.6c.



(a) Delaunay triangulation. **(b)** Non-Delaunay triangulation. **(c)** Two Delaunay triangulations.

Figure 5.6: *Triangulations of four points in convex position.*

Proposition 5.9. *Given a set $P \subset \mathbb{R}^2$ of four points that are in convex position but not cocircular. Then P has exactly one Delaunay triangulation.*

Proof. Consider a convex polygon $P = pqrs$. There are two triangulations of P: a triangulation $\mathcal{T}_1$ using the edge pr and a triangulation $\mathcal{T}_2$ using the edge qs.

Consider the family $\mathcal{C}_1$ of circles through pr, which contains the circumcircles $C_1 = pqr$ and $C_1' = rsp$ of the triangles in $\mathcal{T}_1$. By assumption s is not on C_1. If s is outside of C_1, then q is outside of C_1': Consider the process of continuously moving from C_1 to C_1' in $\mathcal{C}_1$ (Figure 5.7a); the point q is "left behind" immediately when going beyond C_1 and only the final circle C_1' "grabs" the point s.

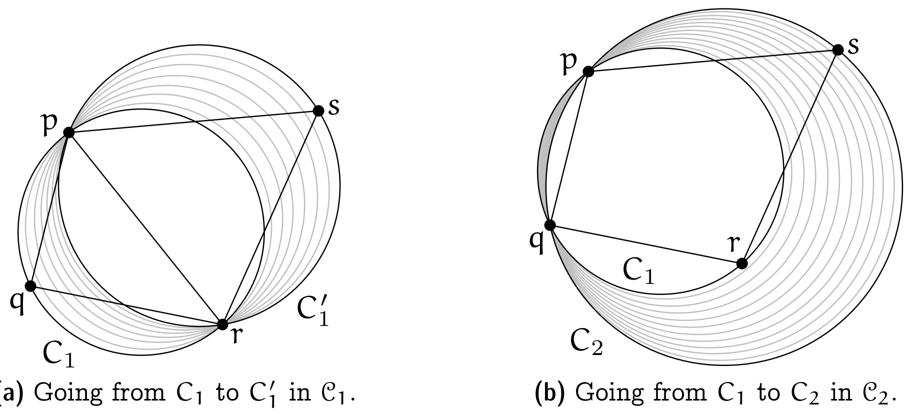

p
s
q
r
C_1
C_1'
(a) Going from C_1 to C_1' in $\mathcal{C}_1$.
p
s
q
r
C_1
C_2
(b) Going from C_1 to C_2 in $\mathcal{C}_2$.

Figure 5.7: *Circumcircles and containment for triangulations of four points.*

Similarly, consider the family $\mathcal{C}_2$ of circles through pq, which contains the circumcircles $C_1 = pqr$ and $C_2 = spq$, the latter belonging to a triangle in $\mathcal{T}_2$. As s is outside of C_1, it follows that r is inside C_2: Consider the process of continuously moving from C_1 to C_2 in $\mathcal{C}_2$ (Figure 5.7b); the point r is on C_1 and remains within the circle all the way up to C_2. This shows that $\mathcal{T}_1$ is Delaunay, whereas $\mathcal{T}_2$ is not.

The case that s is located inside C_1 is symmetric: just cyclically shift the roles of $pqrs$ to $qrsp$. □

Exercise 5.10. *Prove or disprove that every minimum weight triangulation (that is, a triangulation for which the sum of edge lengths is minimum) is a Delaunay triangulation.*

5.2 The Lawson Flip algorithm

It is not clear yet that every point set actually has a Delaunay triangulation (given that not all points are on a common line). In this and the next two sections, we will prove that this is the case. The proof is algorithmic. Here is the *Lawson flip algorithm* for a set P of n points.

1. Compute some triangulation of P (for example, the scan triangulation).
2. While there exists a subtriangulation of four points in convex position that is not Delaunay (like in Figure 5.6b), replace this subtriangulation by the other triangulation of the four points (Figure 5.6a).

We call the replacement operation in the second step a *(Lawson) flip*.

Theorem 5.11. *Let $P \subseteq \mathbb{R}^2$ be a set of n points, equipped with some triangulation $\mathcal{T}$. The Lawson flip algorithm terminates after at most $\binom{n}{2} = O(n^2)$ flips, and the resulting triangulation $\mathcal{D}$ is a Delaunay triangulation of P.*

We will prove Theorem 5.11 in two steps: First we show that the program described above always terminates and, therefore, is an algorithm, indeed (Section 5.3). Then we show that the algorithm does what it claims to do, namely the result is a Delaunay triangulation (Section 5.4).

5.3 Termination of the Lawson Flip Algorithm: The Lifting Map

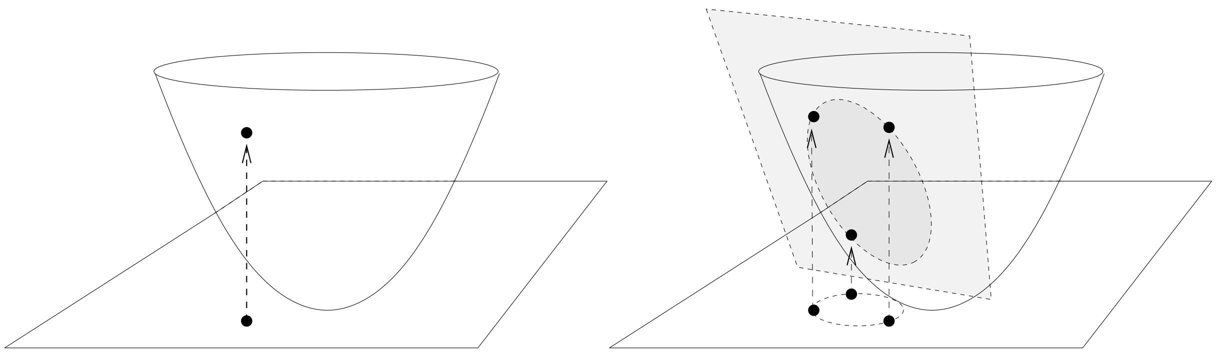
In order to prove Theorem 5.11, we invoke the (parabolic) *lifting map*. This is the following: given a point $p = (x, y) \in \mathbb{R}^2$, its *lifting* $\ell(p)$ is the point

$$\ell(p) = (x, y, x^2 + y^2) \in \mathbb{R}^3.$$

Geometrically, ℓ "lifts" the point vertically up until it lies on the *unit paraboloid*

$$\{(x, y, z) \mid z = x^2 + y^2\} \subseteq \mathbb{R}^3,$$

see Figure 5.8a.



(a) The lifting map.

(b) Points on/inside/outside a circle are lifted to points on/below/above a plane.

Figure 5.8: *The lifting map: circles map to planes.*

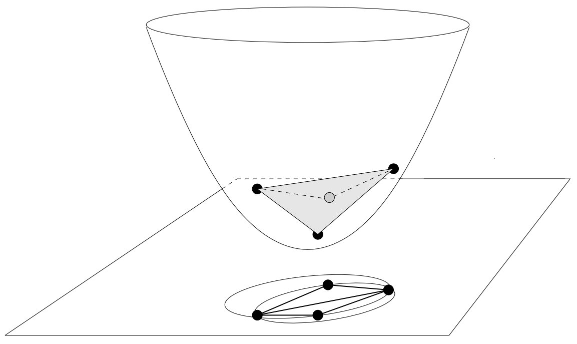
Recall the following important property of the lifting map that we proved in Exercise 4.28. It is illustrated in Figure 5.8b.

Lemma 5.12. *Let* $C \subseteq \mathbb{R}^2$ *be a circle of positive radius. The "lifted circle"* $\ell(C) = \{\ell(p) \mid p \in C\}$ *is contained in a unique plane* $h_C \subseteq \mathbb{R}^3$. *Moreover, a point* $p \in \mathbb{R}^2$ *is strictly inside (outside, respectively) of* C *if and only if the lifted point* $\ell(p)$ *is strictly below (above, respectively)* h_C.

Using the lifting map, we can now prove Theorem 5.11. Let us fix the point set P for this and the next section. First, we need to argue that the algorithm indeed terminates (if you think about it a little, this is not obvious). So let us interpret a flip operation in the lifted picture. The flip involves four points in convex position in $\mathbb{R}^2$, and their lifted images form a tetrahedron in $\mathbb{R}^3$ (think about why this tetrahedron cannot be "flat").

The tetrahedron is made up of four triangles; when you look at it from the top, you see two of the triangles, and when you look from the bottom, you see the other two. In fact, what you see from the top and the bottom are the lifted images of the two possible triangulations of the four-point set in $\mathbb{R}^2$ that is involved in the flip.

Here is the crucial fact that follows from Lemma 5.12: The two top triangles come from the non-Delaunay triangulation before the flip, see Figure 5.9a. The reason is that both top triangles have the respective fourth point below them, meaning that in $\mathbb{R}^2$, the circumcircles of these triangles contain the respective fourth point—the empty circle property is violated. In contrast, the bottom two triangles come from the Delaunay triangulation of the four points: they both have the respective fourth point above them, meaning that in $\mathbb{R}^2$, the circumcircles of the triangles do not contain the respective fourth point, see Figure 5.9b.


(a) Before the flip: the top two triangles of the tetrahedron and the corresponding non-Delaunay triangulation in the plane.

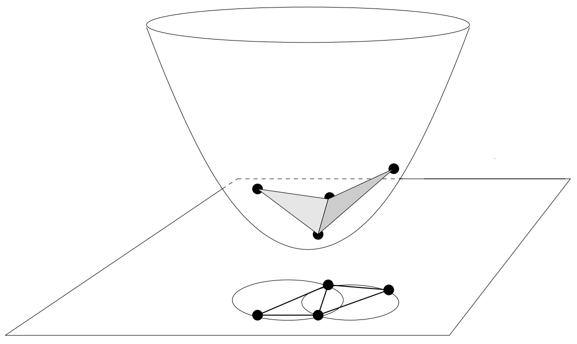

(b) After the flip: the bottom two triangles of the tetrahedron and the corresponding Delaunay triangulation in the plane.

Figure 5.9: *Lawson flip: the height of the surface of lifted triangles decreases.*

In the lifted picture, a Lawson flip can therefore be interpreted as an operation that replaces the top two triangles of a tetrahedron by the bottom two ones. If we consider the lifted image of the current triangulation, we therefore have a surface in $\mathbb{R}^3$ whose pointwise height can only decrease through Lawson flips. In particular, once an edge has been flipped, this edge will be strictly above the resulting surface and can therefore never be flipped a second time. Since n points can span at most $\binom{n}{2}$ edges, the bound on the number of flips follows.

5.4 Correctness of the Lawson Flip Algorithm

It remains to show that the triangulation of P that we get upon termination of the Lawson flip algorithm is indeed a Delaunay triangulation. Here is a first observation telling us that the triangulation is "locally Delaunay".

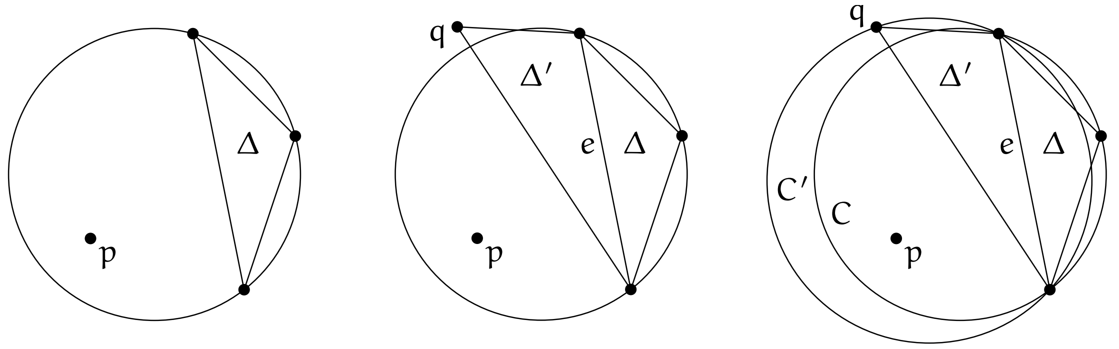
Observation 5.13. *Let Δ, Δ' be two adjacent triangles in the triangulation $\mathcal{D}$ that results from the Lawson flip algorithm. Then the circumcircle of Δ does not have any vertex of Δ' in its interior, and vice versa.*

If the two triangles together form a convex quadrilateral, this follows from the fact that the Lawson flip algorithm did not flip the common edge of Δ and Δ'. If the four vertices are not in convex position, this is basic geometry: given a triangle Δ, its circumcircle C can only contain points of $C \setminus \Delta$ that form a convex quadrilateral with the vertices of Δ.

Now we show that the triangulation is also "globally Delaunay".

Proposition 5.14. *The triangulation $\mathcal{D}$ that results from the Lawson flip algorithm is a Delaunay triangulation.*

Proof. Suppose for contradiction that there is some triangle $\Delta \in \mathcal{D}$ and some point $p \in P$ strictly inside the circumcircle C of Δ. Among all such pairs (Δ, p), we choose one for which the distance of p to Δ is minimal. Note that this distance is positive since $\mathcal{D}$ is a triangulation of P. The situation is as depicted in Figure 5.10a.



(a) A point p inside the circumcircle C of a triangle Δ.
(b) The edge e of Δ closest to p and the second triangle Δ' incident to e.
(c) The circumcircle C' of Δ' also contains p, and p is closer to Δ' than to Δ.

Figure 5.10: *Correctness of the Lawson flip algorithm.*

Now consider the edge e of Δ that is facing p. There must be another triangle Δ' in $\mathcal{D}$ that is incident to the edge e. By the local Delaunay property of $\mathcal{D}$, the third vertex q of Δ' is on or outside of C, see Figure 5.10b. But then the circumcircle C' of Δ' contains the whole portion of C on p's side of e, hence it also contains p; moreover, p is closer to Δ' than to Δ (Figure 5.10c). But this is a contradiction to our choice of Δ and p. Hence there was no (Δ, p), and $\mathcal{D}$ is a Delaunay triangulation. $\square$

Exercise 5.15. *The Euclidean minimum spanning tree (EMST) of a finite point set $P \subset \mathbb{R}^2$ is a spanning tree for which the sum of the edge lengths is minimum (among all spanning trees of P). Show:*

(a) *Every EMST of* P *is a plane graph.*

(b) *Every EMST of* P *contains a closest pair, that is, an edge between two points* $p, q \in P$ *that have minimum distance to each other among all point pairs in* $\binom{P}{2}$.

(c) *Every Delaunay Triangulation of* P *contains an EMST of* P.

Exercise 5.16. (a) *Show that for any two triangulations* T_1 *and* T_2 *on a point set* P, *it is possible to transform* T_1 *into* T_2 *using* $O(n^2)$ *edge flips.*

(b) *Let* D *be a double chain (two convex chains of the same size, facing each other so that for every line* ℓ *through two points on one chain, all points of the other chain are on the same side of* ℓ*). Show that there are two triangulations* T_1 *and* T_2 *on* D *such that at least* $\Omega(n^2)$ *edge flips are needed to transform* T_1 *into* T_2.

(c) *Show that* D *can be constructed in such a way that one of the triangulations from (b), say,* T_1 *is a Delaunay triangulation.*

5.5 The Delaunay Graph

Despite the fact that a point set may have more than one Delaunay triangulation, there are certain edges that are present in every Delaunay triangulation, for instance, the edges of the convex hull.

Definition 5.17. *The* **Delaunay graph** *of* $P \subseteq \mathbb{R}^2$ *consists of all line segments* $\overline{pq}$, *for* $p, q \in P$, *that are contained in every Delaunay triangulation of* P.

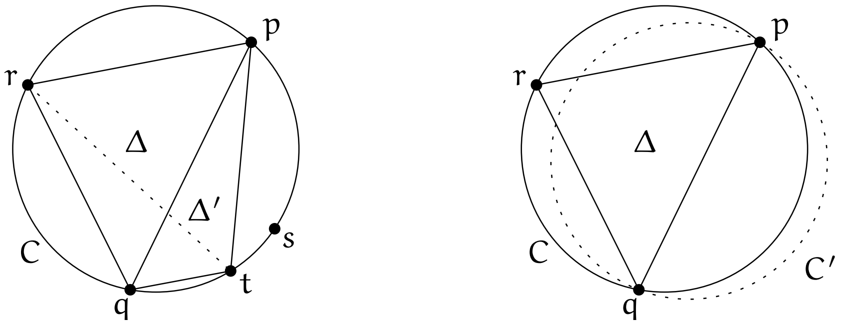
The following characterizes the edges of the Delaunay graph.

Lemma 5.18. *The segment* $\overline{pq}$, *for* $p, q \in P$, *is in the Delaunay graph of* P *if and only if there exists a circle through* p *and* q *that has* p *and* q *on its boundary and all other points of* P *are strictly outside.*

Proof. "$\Rightarrow$": Let pq be an edge in the Delaunay graph of P, and let $\mathcal{D}$ be a Delaunay triangulation of P. Then there exists a triangle $\Delta = pqr$ in $\mathcal{D}$, whose circumcircle C does not contain any point from P in its interior.

If there is a point s on ∂C such that $\overline{rs}$ intersects $\overline{pq}$, then let $\Delta' = pqt$ denote the other ($\neq \Delta$) triangle in $\mathcal{D}$ that is incident to pq (Figure 5.11a). Flipping the edge pq to rt yields another Delaunay triangulation of P that does not contain the edge pq, in contradiction to pq being an edge in the Delaunay graph of P. Therefore, there is no such point s.

Otherwise we can slightly change the circle C by moving away from r while keeping p and q on the circle. As P is a finite point set, we can do such a modification without catching another point from P with the circle. In this way we obtain a circle C' through p and q such that all other points from P are strictly outside C' (Figure 5.12b).



(a) Another point $s \in \partial C$. (b) Moving C away from s.

Figure 5.11: *Characterization of edges in the Delaunay graph (I).*

"$\Leftarrow$": Let $\mathcal{D}$ be a Delaunay triangulation of P. If $\overline{pq}$ is not an edge of $\mathcal{D}$, there must be another edge of $\mathcal{D}$ that crosses $\overline{pq}$ (otherwise, we could add $\overline{pq}$ to $\mathcal{D}$ and still have a plane graph, a contradiction to $\mathcal{D}$ being a triangulation of P). Let rs denote the first edge of $\mathcal{D}$ that intersects the directed line segment $\overline{pq}$.

Consider the triangle Δ of $\mathcal{D}$ that is incident to rs on the side that faces p (given that $\overline{rs}$ intersects $\overline{pq}$ this is a well defined direction). By the choice of rs neither of the other two edges of Δ intersects $\overline{pq}$, and $p \notin \Delta^\circ$ because Δ is part of a triangulation of P. The only remaining option is that p is a vertex of $\Delta = prs$. As Δ is part of a Delaunay triangulation, its circumcircle C_Δ is empty (i.e., ${C_\Delta}^\circ \cap P = \emptyset$).

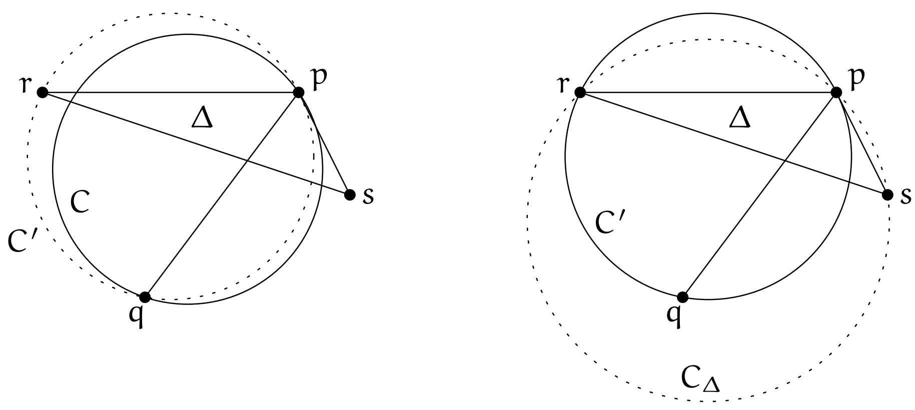
Consider now a circle C through p and q, which exists by assumption. Fixing p and q, expand C towards r to eventually obtain the circle C' through p, q, and r (Figure 5.12a). Recall that r and s are on different sides of the line through p and q. Therefore, s lies strictly outside of C'. Next fix p and r and expand C' towards s to eventually obtain the circle C_Δ through p, r, and s (Figure 5.12b). Recall that s and q are on the same side of the line through p and r. Therefore, $q \in C_\Delta$, which is in contradiction to C_Δ being empty. It follows that there is no Delaunay triangulation of P that does not contain the edge pq. □

The Delaunay graph is useful to prove uniqueness of the Delaunay triangulation in case of general position.

Corollary 5.19. *Let $P \subset \mathbb{R}^2$ be a finite set of points in general position, that is, no four points of P are cocircular. Then P has a unique Delaunay triangulation.* □

5.6 Every Delaunay Triangulation Maximizes the Smallest Angle

Why are we actually interested in Delaunay triangulations? After all, having empty circumcircles is not a goal in itself. But it turns out that Delaunay triangulations satisfy a number of interesting properties. Here we show just one of them.



(a) Expanding C towards r. (b) Expanding C′ towards s.

Figure 5.12: *Characterization of edges in the Delaunay graph (II).*

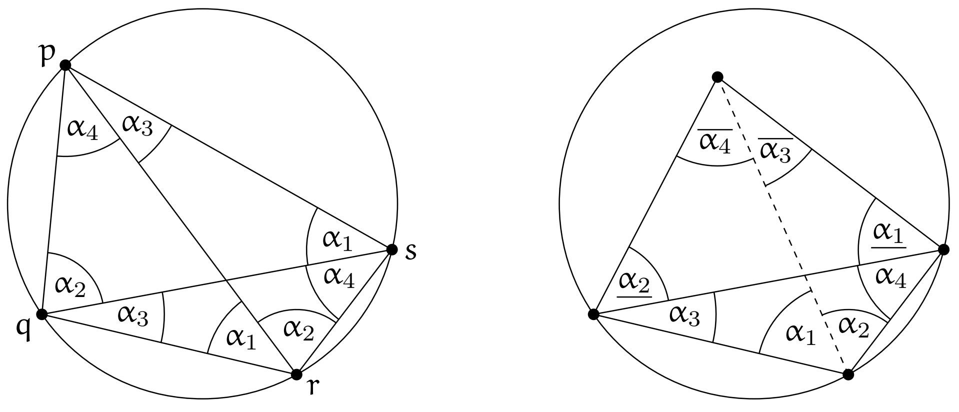
Recall that when we compared a scan triangulation with a Delaunay triangulation of the same point set in Figure 5.3, we claimed that the scan triangulation is "ugly" because it contains many long and skinny triangles. The triangles of the Delaunay triangulation, at least in this example, look much nicer, that is, much closer to an equilateral triangle. One way to quantify this "niceness" is to look at the angles that appear in a triangulation: If all angles are large, then all triangles are reasonably close to an equilateral triangle. Indeed, we will show that Delaunay triangulations maximize the smallest angle among all triangulations of a given point set. Note that this does not imply that there are no long and skinny triangles in a Delaunay triangulation. But if there is a long and skinny triangle in a Delaunay triangulation, then there is an at least as long and skinny triangle in *every* triangulation of the point set.

Given a triangulation $\mathcal{T}$ of P, consider the sorted sequence $A(\mathcal{T}) = (\alpha_1, \alpha_2, \ldots, \alpha_{3m})$ of interior angles, where m is the number of triangles (we have already remarked earlier that m is a function of P only and does not depend on $\mathcal{T}$). Being sorted means that $\alpha_1 \leqslant \alpha_2 \leqslant \cdots \leqslant \alpha_{3m}$. Let $\mathcal{T}, \mathcal{T}'$ be two triangulations of P. We say that $A(\mathcal{T}) < A(\mathcal{T}')$ if there exists some i for which $\alpha_i < \alpha_i'$ and $\alpha_j = \alpha_j'$, for all $j < i$. (This is nothing but the lexicographic order on these sequences.)

Theorem 5.20. *Let $P \subseteq \mathbb{R}^2$ be a finite set of points in general position (not all collinear and no four cocircular). Let $\mathcal{D}^*$ be the unique Delaunay triangulation of P, and let $\mathcal{T}$ be any triangulation of P. Then $A(\mathcal{T}) \leqslant A(\mathcal{D}^*)$.*

In particular, $\mathcal{D}^*$ maximizes the smallest angle among all triangulations of P.

Proof. We know that $\mathcal{T}$ can be transformed into $\mathcal{D}^*$ through the Lawson flip algorithm, and we are done if we can show that each such flip lexicographically increases the sorted angle sequence. A flip replaces six interior angles by six other interior angles, and we will actually show that the smallest of the six angles *strictly* increases under the flip. This implies that the whole angle sequence increases lexicographically.



(a) Four cocircular points and the induced eight angles.

(b) The situation before a flip.

Figure 5.13: *Angle-optimality of Delaunay triangulations.*

Let us first look at the situation of four cocircular points, see Figure 5.13a. In this situation, the *inscribed angle theorem* (a generalization of Thales' Theorem, stated below as Theorem 5.21) tells us that the eight depicted angles come in four equal pairs. For instance, the angles labeled α_1 at s and r are angles on the same side of the chord pq of the circle.

In Figure 5.13b, we have the situation in which we perform a Lawson flip (replacing the solid with the dashed diagonal). By the symbol $\underline{\alpha}$ ($\overline{\alpha}$, respectively) we denote an angle strictly smaller (larger, respectively) than α. Here are the six angles before the flip:

$$\alpha_1 + \alpha_2, \quad \alpha_3, \quad \alpha_4, \quad \underline{\alpha_1}, \quad \underline{\alpha_2}, \quad \overline{\alpha_3} + \overline{\alpha_4}.$$

After the flip, we have

$$\alpha_1, \quad \alpha_2, \quad \overline{\alpha_3}, \quad \overline{\alpha_4}, \quad \underline{\alpha_1} + \alpha_4, \quad \underline{\alpha_2} + \alpha_3.$$

Now, for *every* angle after the flip there is at least one smaller angle before the flip:

$$\begin{aligned}
\alpha_1 &> \underline{\alpha_1}, \\
\alpha_2 &> \underline{\alpha_2}, \\
\overline{\alpha_3} &> \alpha_3, \\
\overline{\alpha_4} &> \alpha_4, \\
\underline{\alpha_1} + \alpha_4 &> \alpha_4, \\
\underline{\alpha_2} + \alpha_3 &> \alpha_3.
\end{aligned}$$

It follows that the smallest angle strictly increases. □

Theorem 5.21 (Inscribed Angle Theorem). *Let C be a circle with center c and positive radius and $p, q \in C$. Then the angle $\angle prq \bmod \pi = \frac{1}{2}\angle pcq$ is the same, for all $r \in C$.*

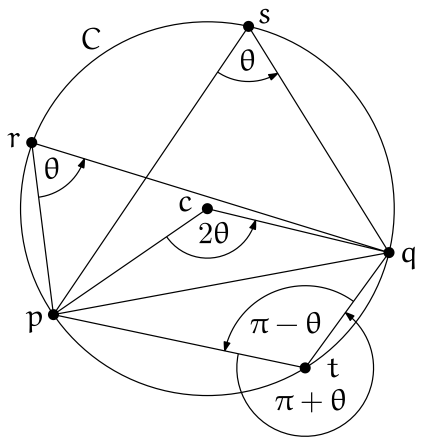
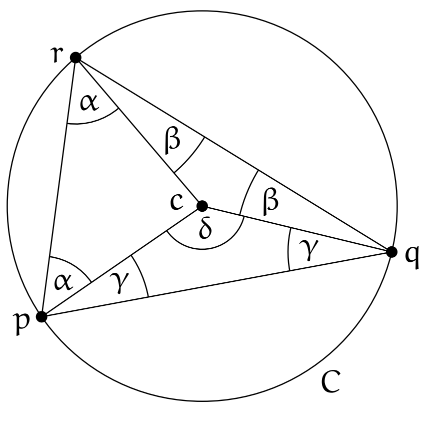


Figure 5.14: *The Inscribed Angle Theorem with* $\theta := \angle prq$.

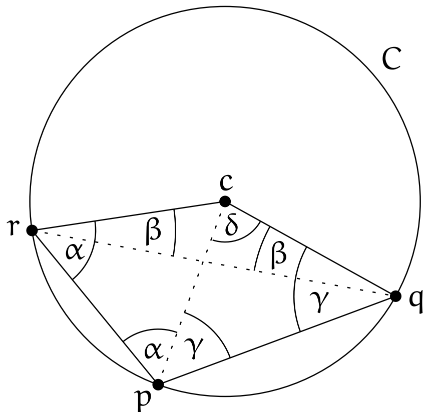
Proof. Without loss of generality we may assume that c is located to the left of or on the oriented line pq.

Consider first the case that the triangle $\Delta = pqr$ contains c. Then Δ can be partitioned into three triangles: pcr, qcr, and cpq. All three triangles are isosceles, because two sides of each form the radius of C. Denote $\alpha = \angle prc$, $\beta = \angle crq$, $\gamma = \angle cpq$, and $\delta = \angle pcq$ (see the figure shown to the right). The angles we are interested in are $\theta = \angle prq = \alpha + \beta$ and δ, for which we have to show that $\delta = 2\theta$.



Indeed, the angle sum in Δ is $\pi = 2(\alpha + \beta + \gamma)$ and the angle sum in the triangle cpq is $\pi = \delta + 2\gamma$. Combining both yields $\delta = 2(\alpha + \beta) = 2\theta$.

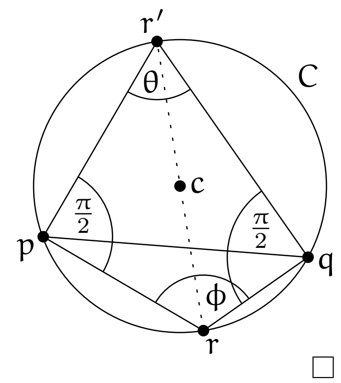
Next suppose that $pqcr$ are in convex position and r is to the left of or on the oriented line pq. Without loss of generality let r be to the left of or on the oriented line qc. (The case that r lies to the right of or on the oriented line pc is symmetric.) Define α, β, γ, δ as above and observe that $\theta = \alpha - \beta$. Again have to show that $\delta = 2\theta$.



The angle sum in the triangle cpq is $\pi = \delta + 2\gamma$ and the angle sum in the triangle rpq is $\pi = (\alpha - \beta) + \alpha + \gamma + (\gamma - \beta) = 2(\alpha + \gamma - \beta)$. Combining both yields $\delta = \pi - 2\gamma = 2(\alpha - \beta) = 2\theta$.

It remains to consider the case that r is to the right of the oriented line pq.

Consider the point r' that is antipodal to r on C, and the quadrilateral $Q = prqr'$. We are interested in the angle ϕ of Q at r. By Thales' Theorem the inner angles of Q at p and q are both $\pi/2$. Hence the angle sum of Q is $2\pi = \theta + \phi + 2\pi/2$ and so $\phi = \pi - \theta$.



□

What happens in the case where the Delaunay triangulation is not unique? The following still holds.

Theorem 5.22. *Let $P \subseteq \mathbb{R}^2$ be a finite set of points, not all on a line. Every Delaunay triangulation $\mathcal{D}$ of P maximizes the smallest angle among all triangulations $\mathcal{T}$ of P.*

Proof. Let $\mathcal{D}$ be some Delaunay triangulation of P. We infinitesimally perturb the points in P such that no four are on a common circle anymore. Then the Delaunay triangulation becomes unique (Corollary 5.19). Starting from $\mathcal{D}$, we keep applying Lawson flips until we reach the unique Delaunay triangulation $\mathcal{D}^*$ of the perturbed point set. Now we examine this sequence of flips on the original *unperturbed* point set. All these flips must involve four cocircular points (only in the cocircular case, an infinitesimal perturbation can change "good" edges into "bad" edges that still need to be flipped). But as Figure 5.13 (a) easily implies, such a "degenerate" flip does not change the smallest of the six involved angles. It follows that $\mathcal{D}$ and $\mathcal{D}^*$ have the same smallest angle, and since $\mathcal{D}^*$ maximizes the smallest angle among all triangulations $\mathcal{T}$ (Theorem 5.20), so does $\mathcal{D}$. □

5.7 Constrained Triangulations

Sometimes one would like to have a Delaunay triangulation, but certain edges are already prescribed, for example, a Delaunay triangulation of a simple polygon. Of course, one cannot expect to be able to get a proper Delaunay triangulation where all triangles satisfy the empty circle property. But it is possible to obtain some triangulation that comes as close as possible to a proper Delaunay triangulation, given that we are forced to include the edges in E. Such a triangulation is called a *constrained Delaunay triangulation*, a formal definition of which follows.

Let $P \subseteq \mathbb{R}^2$ be a finite point set and $G = (P, E)$ a geometric graph with vertex set P (we consider the edges $e \in E$ as line segments). A triangulation $\mathcal{T}$ of P *respects* G if it contains all segments $e \in E$. A triangulation $\mathcal{T}$ of P that respects G is said to be a *constrained Delaunay triangulation* of P with respect to G if the following holds for every triangle Δ of $\mathcal{T}$:

> The circumcircle of Δ contains only points $q \in P$ in its interior that are not visible from the interior of Δ. A point $q \in P$ is *visible* from the interior of

Δ if there exists a point p in the interior of Δ such that the line segment $\overline{pq}$ does not intersect any segment $e \in E$. We can thus imagine the line segments of E as "blocking the view".

For illustration, consider the simple polygon and its constrained Delaunay triangulation shown in Figure 5.15. The circumcircle of the shaded triangle Δ contains a whole other triangle in its interior. But these points cannot be seen from Δ°, because all possible connecting line segments intersect the blocking polygon edge e of Δ.

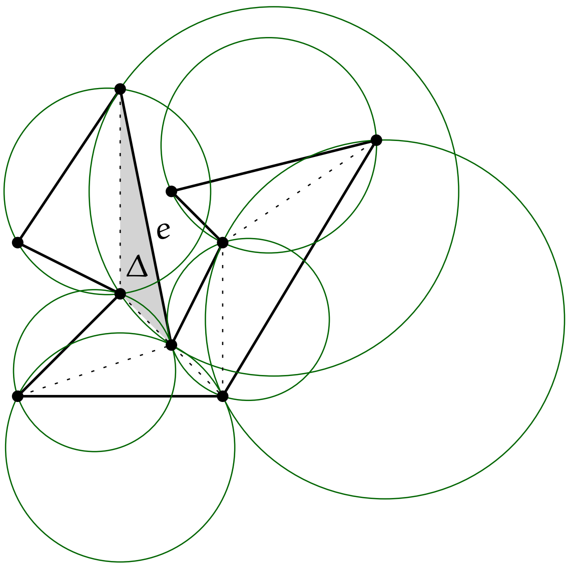


Figure 5.15: *Constrained Delaunay triangulation of a simple polygon.*

Theorem 5.23. *For every finite point set* P *and every plane graph* $G = (P, E)$*, there exists a constrained Delaunay triangulation of* P *with respect to* G.

Exercise 5.24. *Prove Theorem 5.23. Also describe a polynomial algorithm to construct such a triangulation.*

Questions

20. *What is a triangulation?* Provide the definition and prove a basic property: every triangulation with the same set of vertices and the same outer face has the same number of triangles.
21. *What is a triangulation of a point set?* Give a precise definition.
22. *Does every point set (not all points on a common line) have a triangulation?* You may, for example, argue with the scan triangulation.
23. *What is a Delaunay triangulation of a set of points*? Give a precise definition.

24. *What is the Delaunay graph of a point set?* Give a precise definition and a characterization.
25. *How can you prove that every set of points (not all on a common line) has a Delaunay triangulation?* You can for example sketch the Lawson flip algorithm and the Lifting Map, and use these to show the existence.
26. *When is the Delaunay triangulation of a point set unique?* Show that general position is a sufficient condition. Is it also necessary?
27. *What can you say about the "quality" of a Delaunay triangulation?* Prove that every Delaunay triangulation maximizes the smallest interior angle in the triangulation, among the set of all triangulations of the same point set.

Chapter 6

Delaunay Triangulation: Incremental Construction

In the last lecture, we have learned about the Lawson flip algorithm that computes a Delaunay triangulation of a given n-point set $P \subseteq \mathbb{R}^2$ with $O(n^2)$ Lawson flips. One can actually implement this algorithm to run in $O(n^2)$ time, and there are point sets where it may take $\Omega(n^2)$ flips.

In this lecture, we will discuss a different algorithm. The final goal is to show that this algorithm can be implemented to run in $O(n \log n)$ time. Throughout this lecture we assume that P is in general position (no 3 points on a line, no 4 points on a common circle), so that the Delaunay triangulation is unique (Corollary 5.19). There are techniques to deal with non-general position, but we don't discuss them here.

6.1 Incremental construction

The idea is to build the Delaunay triangulation of P by inserting one point after another according to a random permutation of the remaining vertices, say $p_1, p_2, \ldots, p_n$.

To avoid special cases, we enhance the point set P with three artificial points p_0, p_{-1} and p_{-2} "far out" such that the convex hull of $P \cup \{p_0, p_{-1}, p_{-2}\}$ consists only of the three artificial points. Because the convex hull of the resulting point set is a triangle; later, we can remove these extra points and their incident edges to obtain $\mathcal{DT}(P)$. The incremental algorithm starts off with the Delaunay triangulation of the three artificial points which consists of one big triangle enclosing all other points. (In our figures, we suppress the far-away points, since they are merely a technicality.)

For $1 \leqslant s \leqslant n$, let $P_s := \{p_1, \ldots, p_s\}$ and $P_s^* = P_s \cup \{p_0, p_{-1}, p_{-2}\}$. Throughout, we always maintain the Delaunay triangulation of the point set P_{s-1}^* containing the points inserted so far, and when the next point p_s comes along, we update the triangulation to the Delaunay triangulation of P_s^*. Let $\mathcal{DT}(s)$ denote the Delaunay triangulation of P_s^*.

Now assume that we have already built $\mathcal{DT}(s-1)$, and we next insert p_s. Here is the outline of the update step.

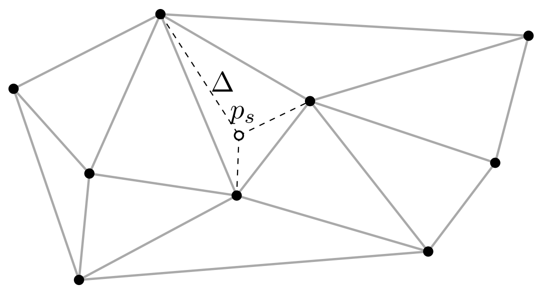


Figure 6.1: *Inserting* p_s *into* $\mathcal{DT}(s-1)$*: Step 1*

1. Find the triangle $\Delta = \Delta(p, q, r)$ of $\mathcal{DT}(s-1)$ that contains p_s, and replace it with the three triangles resulting from connecting p_s with all three vertices p, q, r; see Figure 6.1. We now have a triangulation $\mathcal{T}$ of P^*_s.

2. Perform Lawson flips on $\mathcal{T}$ until $\mathcal{DT}(s)$ is obtained; see Figure 6.2

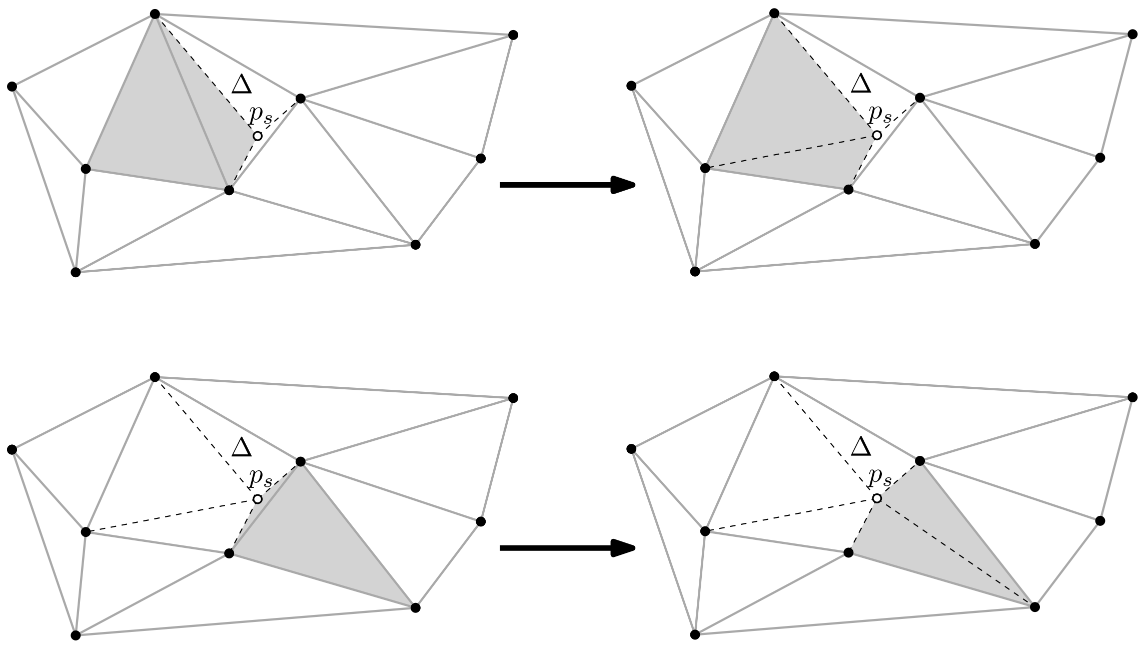


Figure 6.2: *Inserting* p_s *into* $\mathcal{DT}(s-1)$*: Step 2*

How to organize the Lawson flips. The Lawson flips can be organized quite systematically, since we always know the candidates for "bad" edges that may still have to be flipped. Initially (after step 1), only the three edges of Δ can be bad, since these are the only edges for which an incident triangle has changed (by inserting p_s in Step 1). Each of the three new edges is good, since the 4 vertices of its two incident triangles are not in convex position.

Now we have the following invariant (part (a) certainly holds in the first flip):

(a) In every flip, the convex quadrilateral Q in which the flip happens has exactly two edges incident to p_s, and the flip generates a new edge incident to p_s.

(b) Only the two edges of Q that are *not* incident to p_s can become bad after the flip.

We will prove part (b) in the next lemma. The invariant then follows since (b) entails (a) in the next flip. This means that we can maintain a queue of potentially bad edges that we process in turn. A good edge will be removed from the queue, and a bad edge will be flipped and replaced according to (b) with two new edges in the queue. In this way, we never flip edges incident to p_s; the next lemma proves that this is correct and at the same time establishes part (b) of the invariant.

Lemma 6.1. *Every edge incident to p_s that is created during the update is an edge of the Delaunay graph of P^*_s and thus an edge that will be in $\mathcal{DT}(s)$. It easily follows that edges incident to p_s will never become bad during the update step.*[1]

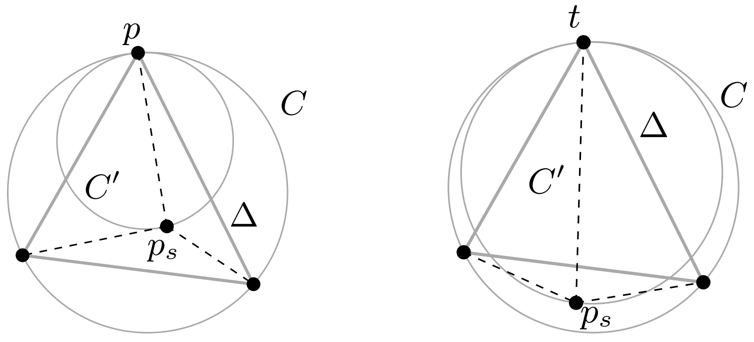
Proof. Let us consider one of the first three new edges, $\overline{p_s p}$, say. Since the triangle Δ has a circumcircle C strictly containing only p_s (Δ is in $\mathcal{DT}(s-1)$), we can shrink that circumcircle to a circle C' through p_s and p with no interior points, see Figure 6.3 (a). This proves that $\overline{p_s p}$ is in the Delaunay graph. If $\overline{p_s t}$ is an edge created by a flip, a similar argument works. The flip destroys exactly one triangle Δ of $\mathcal{DT}(s-1)$. Its circumcircle C contains p_s only, and shrinking it yields an empty circle C' through p_s and t. Thus, $\overline{p_s t}$ is in the Delaunay graph also in this case. □

6.2 The History Graph

What can we say about the performance of the incremental construction? Not much yet. First of all, we did not specify how we find the triangle Δ of $\mathcal{DT}(s-1)$ that contains the point p_s to be inserted. Doing this in the obvious way (checking all triangles) is not good, since already the find steps would then amount to $O(n^2)$ work throughout the whole algorithm. Here is a smarter method, based on the *history graph*.

Definition 6.2. *For a given $1 \leqslant s \leqslant n$, the history graph $\mathcal{H}_{s-1}$ of P^*_{s-1} is a directed acyclic graph whose vertices are all triangles that have ever been created during the incremental construction of $\mathcal{DT}(s-1)$. There is a directed edge from Δ to Δ' whenever Δ has been destroyed during an insertion step, Δ' has been created during the same insertion step, and Δ overlaps with Δ' in its interior.*

[1]If such an edge was bad, it could be flipped, but then it would be "gone forever" according to the lifting map interpretation from the previous lecture.



(a) New edge $\overline{p_s p}$ incident to p_s created in Step 1 **(b)** New edge $\overline{p_s t}$ incident to p_s created in Step 2

Figure 6.3: *Newly created edges incident to* p_s *are in the Delaunay graph*

It follows that the history graph $\mathcal{H}_{s-1}$ contains triangles of outdegrees 3, 2 and 0. The ones of outdegree 0 are clearly the triangles of $\mathcal{DT}(s-1)$.

The triangles of outdegree 3 are the ones that have been destroyed during Step 1 of an insertion. For each such triangle Δ, its three outneighbors are the three new triangles that have replaced it, see Figure 6.4.

The triangles of outdegree 2 are the ones that have been destroyed during Step 2 of an insertion. For each such triangle Δ, its two outneighbors are the two new triangles created during the flip that has destroyed Δ, see Figure 6.5.

The history graph $\mathcal{H}_{s-1}$ can be built during the incremental construction at asymptotically no extra cost; but it may need extra space since it keeps all triangles ever created. Given the history graph $\mathcal{H}_{s-1}$, we can search for the triangle Δ of $\mathcal{DT}(s-1)$ that contains p_s, as follows. We start from the big triangle $\triangle(p_0, p_{-1}, p_{-2})$; this one certainly contains p_s. Then we follow a directed path in the history graph. If the current triangle still has outneighbors, we find the unique outneighbor containing p_s and continue the search with this neighbor. If the current triangle has no outneighbors anymore, it is in $\mathcal{DT}(s-1)$ and contains p_s—we are done. Thus, the complexity of finding the triangle containing p_s is linear on the length of the path followed in the history graph.

Types of triangles in the history graph. After each insertion of a point p_s, several triangles are created and added to the history graph. It is important to note that these triangles come in two types: Some of them are valid Delaunay triangles of $\mathcal{DT}(s)$, and they survive to the next stage of the incremental construction. Other triangles are immediately destroyed by subsequent Lawson flips, because they are not Delaunay triangles of $\mathcal{DT}(S)$.

Note that, whenever a Lawson flip is performed, one of the two triangles destroyed is always a “valid” triangle from a previous iteration, and the other one is an “ephemeral" triangle that was created at this iteration. The ephemeral triangle is always the one that has p_s, the newly inserted point, as a vertex.

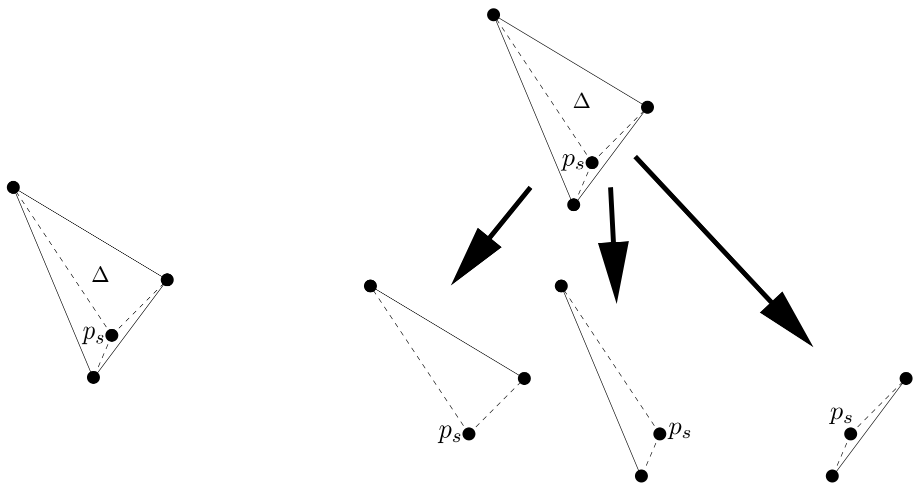


Figure 6.4: *The history graph: one triangle gets replaced by three triangles*

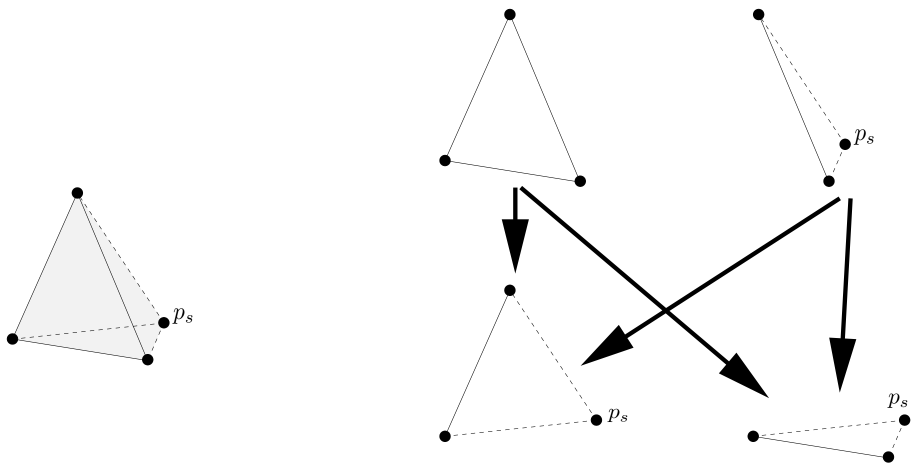


Figure 6.5: *The history graph: two triangles get replaced by two triangles*

6.3 Analysis of the algorithm

To formalize the above intuition, we observe the following.

Observation 6.3. *Given $\mathcal{DT}(s-1)$ and the triangle Δ of $\mathcal{DT}(s-1)$ that contains p_s, we can build $\mathcal{DT}(s)$ in time proportional to the degree of p_s in $\mathcal{DT}(s)$, which is the number of triangles of $\mathcal{DT}(s)$ containing p_s. Moreover, the total number of triangles created throughout this insertion is at most twice the degree of p_s in $\mathcal{DT}(s)$.*

Indeed, since every flip generates exactly one new triangle incident to p_s, the number of flips is the degree of p_s minus three. Step 1 of the update takes constant time, and since also every flip can be implemented in constant time, the observation follows.

Using this result, we can prove the following bound on the expected size of the history graph.

Lemma 6.4. *The expected number of nodes in the history graph is at most $9n+1$.*

Proof. Before start inserting points of P, our history graph consists only of the artificial triangle $\triangle(p_0p_{-1}p_{-2})$. In the s-th iteration of the algorithm, we insert the point p_s. At this point, we first split the triangle $\triangle(p_ip_jp_k)$ containing p_s into three new triangles, $i, j, k < s$. This splitting adds three new vertices to the history graph, and three new Delaunay edges incident to p_s, namely $\overline{p_sp_i}$, $\overline{p_sp_j}$ and $\overline{p_sp_k}$. In addition, we use Lawson flips until obtaining $\mathcal{DT}(s)$. By Observation 6.3, we know that if p_s has degree d_s in $\mathcal{DT}(s)$, then the total number of triangles created throughout the insertion of p_s is at most $2d_s$. Here is where we use *backwards analysis* to bound the value of the random variable d_s. Because $\mathcal{DT}(s)$ is a triangulation with $s+3$ points, it has $3(s+3)-6$ edges. If we exclude the three edges of the convex hull, we get that the sum of the degree of all interior vertices in $\mathcal{DT}(s)$ adds up to at most $2(3(s+3)-9)=6s$. This means, that the expected degree of a random point of P_s (i.e., not including p_0, p_{-1} or p_{-2}) is at most 6. In summary, we get that

$$E[\text{number of triangles created in iteration } s] \leqslant E[2d_s-3] = 2E[d_s]-3 \leqslant 2\cdot 6-3 = 9.$$

Because in the first step we create only one triangle, namely $\triangle(p_0p_{-1}p_{-2})$, and since the expected number of triangles created in each insertion step is at most 9, we get by linearity of expectation that the total expected number of triangles created is at most $9n+1$. □

Note that we cannot say that all insertions create a number of triangle close to 9, i.e., there could be some very costly insertions throughout. However, the average is constant which provides us with a linear expected total value. As a summation of independent random variables, the size of the history graph is indeed concentrated around its mean, which can be shown using standard Chernoff bounds. We proceed now to prove our main result.

Theorem 6.5. *The Delaunay triangulation of a set* P *of* n *points in the plane can be computed in* $O(n \log n)$ *expected time, using* $O(n)$ *expected storage.*

Proof. We have already established the correctness of the algorithm. For the storage, we note that only the history graph could use more than linear storage, however Lemma 6.4 proves that its expected size is $O(n)$ yielding the desired bound on the storage.

To bound the running time of the algorithm, we first ignore the time used during the point location queries, i.e., the time used during the insertion of each point to find the triangle that contains it. Ignoring this, the running time of the algorithm is proportional to the number of triangles created. From Lemma 6.4 we know that only $O(n)$ triangles are created in expectation. That is, only $O(n)$ additional expected time is needed.

It remains to account for the point location queries. That is, given $1 \leqslant s \leqslant n$, we are interested in the expected time needed to locate p_s in the triangulation $\mathcal{DT}(s-1)$. Recall that we do this by using the history graph. We start from its root, the triangle $\triangle(p_0, p_{-1}, p_{-2})$, and then traverse a path in this graph that finishes on a node corresponding to the triangle of $\mathcal{DT}(s-1)$ that contains p_s. Since the out-degree of all nodes in the history graph is $O(1)$, the running time of the point location query is proportional to the number of nodes visited. Recall that each internal node of this path corresponds to a triangle that was created at an earlier stage, but that has been destroyed and contains p_s. A triangle $\triangle$ could only be already destroyed if a point p_l lying in its circumcircle was inserted before p_s. Because of this, we introduce the following notation. Given a triangle $\triangle$, let $K(\triangle)$ be the subset of points of P that lie in the circumcircle of $\triangle$. With this notation, we can say that during the insertion of p_s, the time needed to locate it in $\mathcal{DT}(s-1)$ is at most linear on the number of triangles $\triangle$ with $p_s \in K(\triangle)$. One can see that each triangle $\triangle$ can be charged at most once for each of the points of P in $K(\triangle)$. Therefore, the total running time for all point location steps during the construction is

$$O\left(n + \sum_{\triangle} |K(\triangle)|\right),$$

where the summation is taken over all Delaunay triangles $\triangle$ created by the algorithm. We shall prove below that the expected value of this expression is $O(n \log n)$, which will conclude our proof. □

It remains to provide a bound on the expected size of the sets $K(\triangle)$ throughout the running time of the algorithm. Note that for $\mathcal{DT}(1)$, we would expect $K(\triangle)$ to be roughly n for each of its triangles, while for $\mathcal{DT}(n)$, we know that $K(\triangle) = 0$ for all its triangles. In the middle, we would like the values to interpolate nicely giving something close to $K(\triangle) \approx O(n/s)$ for the triangles in $\mathcal{DT}(s)$. While this is not exactly the case, it provides a good intuition. We will show that the average behaves in this way.

Lemma 6.6. *It holds that*

$$E\left[n + \sum_{\triangle} |K(\Delta)|\right] = O(n \log n),$$

where the summation is taken over all Delaunay triangles $\triangle$ created by the algorithm.

Proof. Let τ_s be the set of triangles of $\mathcal{DT}(s)$ that are not part of $\mathcal{DT}(s-1)$, i.e., the set of triangles incident to p_s in $\mathcal{DT}(s)$. Using this notation, we first rewrite the above expression as follows:

$$\sum_{\triangle} |K(\Delta)| = \sum_{s=1}^{n} \left(\sum_{\triangle \in \tau_s} |K(\triangle)| \right). \tag{6.7}$$

This holds because each triangle created by the algorithm is created in some iteration and hence, belongs to some set τ_s for some $1 \leqslant s \leqslant n$.

For a point q in the plane, let $\varphi_s(q)$ denote the number of triangles $\triangle$ of $\mathcal{DT}(s)$ such that $q \in K(\triangle)$. In other words, we can think of placing the circumcircles of all triangles of $\mathcal{DT}(s)$ in the plane and then count how many circles enclose q. Let also $\varphi^*_s(q)$ denote the number of triangles $\triangle$ of τ_s such that $q \in K(\triangle)$. That is, we place the circumcircles of all triangles incident to p_s in $\mathcal{DT}(s)$ and count how many of them enclose q.

Then, we notice that for $1 \leqslant s \leqslant n$, the summation $\sum_{\triangle \in \tau_s} |K(\triangle)|$ counts the number of points of P that lie inside the circumcircles of the triangles in τ_s. Because these circumcircles belong to $\mathcal{DT}(s)$, they are empty of points of P_s, and hence, all points lying inside these circumcircles belong to $P \setminus P_s$. Thus, we get that

$$\sum_{\triangle \in \tau_s} |K(\triangle)| = \sum_{q \in P \setminus P_s} \varphi^*_s(q). \tag{6.8}$$

To analyze the expected value of $\varphi^*_s(q)$, we use conditional expectation. That is, we condition on P_s being a specific set, and then later, take the weighted average of all those expectations. Thus, we fix the set P_s and assume that $P_s = \overline{P_s}$ for some arbitrary subset $\overline{P_s}$ of P with s elements. With this assumption, the triangulation $\mathcal{DT}(s)$ is fixed and hence, $\varphi^*_s(q)$ depends only on which among the elements of $P_s = \overline{P_s}$ is the last one. Since the order of insertion of the elements of P_s is random (random permutation chosen uniformly at random), we get that for a triangle $\triangle$ of $\mathcal{DT}(s)$, this triangle is incident to the random point p_s with probability $3/s$. Therefore, if we let $\chi_{\triangle,s}$ be an indicator random variable that is one if and only if $\triangle$ is incident to p_s (i.e., $\triangle \in \tau_s$), we get that $\Pr[\chi_{\triangle,s} = 1] = 3/s$. Using this, we get that for a point $q \in P \setminus P_s$,

$$E[\varphi^*_s(q)] = \sum_{\substack{q \in K(\triangle), \\ \triangle \in \mathcal{DT}(s)}} E[\chi_{\triangle,s} \cdot \varphi_s(q)] = \frac{3}{s} \cdot \varphi_s(q).$$

Plugging this into (6.8) and taking expectation, we get that by linearity of expectation, the following holds

$$E\left[\sum_{\triangle \in \tau_s} |K(\triangle)|\right] = \sum_{q \in P \setminus P_s} E[\varphi^*_s(q)] = \frac{3}{s}\left(\sum_{q \in P \setminus P_s} \varphi_s(q) \right). \tag{6.9}$$

Additionally, because any $q \in P \setminus P_s$ is equally likely to be p_{s+1}, i.e., each point of $P \setminus P_s$ is p_{s+1} with probability $1/(n-s)$, we have that

$$E[\varphi_s(p_{s+1})] = \frac{1}{n-s}\left(\sum_{q \in P \setminus P_s} \varphi_s(q)\right).$$

Which implies by arranging the terms that

$$\sum_{q \in P \setminus P_s} \varphi_s(q) = (n-s) \cdot E[\varphi_s(p_{s+1})].$$

Plugging this back into (6.9), we get that

$$E\left[\sum_{\triangle \in \tau_s} |K(\triangle)|\right] = \frac{3(n-s)}{s} E[\varphi_s(p_{s+1})]. \tag{6.10}$$

Recall that $\varphi_s(p_{s+1})$ is the number of triangles $\triangle$ of $\mathcal{DT}(s)$ whose circumcircle encloses p_{s+1}, i..e, $p_{s+1} \in K(\triangle)$. However, these triangles of $\mathcal{DT}(s)$ are exactly the ones that will be destroyed by the insertion of p_{s+1}. Moreover, by Observation 6.3, the triangles destroyed are at most twice the number of triangles of $\mathcal{DT}(s+1)$ incident to p_{s+1}, i..e., the number of triangles in τ_{s+1}. Therefore, we get that $\varphi_s(p_{s+1}) = O(|\tau_{s+1}|)$. Plugging this into (6.10), we get that

$$E\left[\sum_{\triangle \in \tau_s} |K(\triangle)|\right] = O\left(\frac{n-s}{s} \cdot E[|\tau_{s+1}|]\right).$$

Recall that so far, we have assumed that $P_s = \overline{P_s}$. To remove this assumption, we can take the average over all possible different sets $\overline{P_s}$ and all permutations of P. Since all sets and all permutations are equally likely, the average of all of them stays the same. However, now that we are not conditioning the probability, we know that $E[\tau_{s+1}] \leqslant 9$ by the proof of Lemma 6.4. Thus, the previous expression yields that

$$E\left[\sum_{\triangle \in \tau_s} |K(\triangle)|\right] = O\left(\frac{n-s}{s}\right).$$

Summing over all values of s and by linearity of expectation, we get the expected value of (6.7)

$$E\left[\sum_{s=1}^{n}\left(\sum_{\triangle \in \tau_s} |K(\triangle)|\right)\right] = O\left(\sum_{s=1}^{n} \frac{n-s}{s}\right) \leqslant O\left(n \sum_{s=1}^{n} \frac{1}{s}\right) = O(n \log n).$$

$\square$

Exercise 6.11. *For a sequence of* n *pairwise distinct numbers* $y_1, \dots, y_n$ *consider the sequence of pairs* $(\min\{y_1, \dots, y_i\}, \max\{y_1, \dots, y_i\})_{i=0,1,\dots,n}$ *(*$\min \emptyset := +\infty, \max \emptyset := -\infty$*). How often do these pairs change in expectation if the sequence is permuted randomly, each permutation appearing with the same probability? Determine the expected value.*

Exercise 6.12. *Given a set* P *of* n *points in convex position represented by the clockwise sequence of the vertices of its convex hull, provide an algorithm to compute its Delaunay triangulation in* $O(n)$ *time.*

Questions

28. *How can we efficiently compute the three artificial points* p_0, p_{-1} *and* p_{-2} *whose convex hull contains all points of* P*, while keeping their coordinates "small".*
29. *Describe the algorithm for the incremental construction of* $\mathcal{DT}(P)$: how do we find the triangle containing the point p_s to be inserted into $\mathcal{DT}(s-1)$? How do we transform $\mathcal{DT}(P_{s-1})$ into $\mathcal{DT}(s)$? How many steps does the latter transformation take, in terms of $\mathcal{DT}(s)$?
30. *What are the two types of triangles that the history graph contains?*

Chapter 7

Voronoi Diagrams

7.1 Post Office Problem

Suppose there are n post offices $p_1, \ldots p_n$ in a city. Someone who is located at a position q within the city would like to know which post office is closest to him.[1] Modeling the city as a planar region, we think of $p_1, \ldots p_n$ and q as points in the plane. Denote the set of post offices by $P = \{p_1, \ldots p_n\}$.

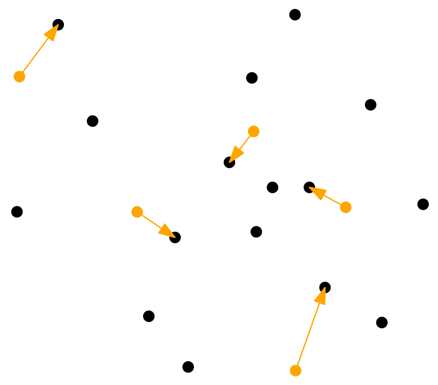


Figure 7.1: *Closest post offices for various query points.*

While the locations of post offices are known and do not change so frequently, we do not know in advance for which—possibly many—query locations the closest post office is to be found. Therefore, our long term goal is to come up with a data structure on top of P that allows to answer any possible query efficiently. The basic idea is to apply a so-called *locus approach*: we partition the query space into regions on which the answer is the same. In our case, this amounts to partition the plane into regions such that for all points within a region the same point from P is closest (among all points from P).

[1]Another—possibly historically more accurate—way to think of the problem: You want to send a letter to a person living at q. For this you need to know the corresponding zip code, which is the code of the post office closest to q.

As a warmup, consider the problem for two post offices $p_i, p_j \in P$. For which query locations is the answer p_i rather than p_j? This region is bounded by the bisector of p_i and p_j, that is, the set of points which have the same distance to both points.

Proposition 7.1. *For any two distinct points in $\mathbb{R}^d$ the bisector is a hyperplane, that is, in $\mathbb{R}^2$ it is a line.*

Proof. Let $p = (p_1, \ldots, p_d)$ and $q = (q_1, \ldots, q_d)$ be two points in $\mathbb{R}^d$. The bisector of p and q consists of those points $x = (x_1, \ldots, x_d)$ for which

$$\begin{aligned}
||p-x|| = ||q-x|| &\iff ||p-x||^2 = ||q-x||^2 \\
&\iff \sum_{i=1}^{d}(p_i - x_i)^2 = \sum_{i=1}^{d}(q_i - x_i)^2 \\
&\iff \sum_{i=1}^{d} {p_i}^2 - 2\sum_{i=1}^{d} p_i x_i + \sum_{i=1}^{d} {x_i}^2 = \sum_{i=1}^{d} {q_i}^2 - 2\sum_{i=1}^{d} q_i x_i + \sum_{i=1}^{d} {x_i}^2 \\
&\iff \sum_{i=1}^{d} {p_i}^2 - \sum_{i=1}^{d} {q_i}^2 = 2\sum_{i=1}^{d}(p_i - q_i)x_i \\
&\iff ||p||^2 - ||q||^2 = 2(p-q)^\top x\,.
\end{aligned}$$

As p and q are distinct, this is the equation of a hyperplane. □

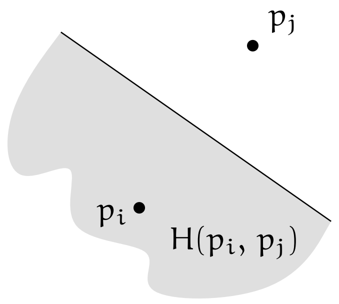


Figure 7.2: *The bisector of two points.*

Denote by $H(p_i, p_j)$ the closed halfspace bounded by the bisector of p_i and p_j that contains p_i. In $\mathbb{R}^2$, the region $H(p_i, p_j)$ is a halfplane; see Figure 7.2.

Exercise 7.2.

a) What is the bisector of a line ℓ and a point $p \in \mathbb{R}^2 \setminus \ell$, that is, the set of all points $x \in \mathbb{R}^2$ with $||x - p|| = ||x - \ell|| = \min_{q \in \ell} ||x - q||$?

b) For two points $p \neq q \in \mathbb{R}^2$, what is the region that contains all points whose distance to p is exactly twice their distance to q?

7.2 Voronoi Diagram

As it turns out, understanding the situation for two points essentially tells us everything we need to know for the general case. The structure obtained by applying the locus approach to the nearest neighbor problem is called *Voronoi diagram*. In fact, this approach works for a variety of distance functions and spaces [2, 7]. So, Voronoi diagram should be considered a family of structures rather than a single specific one. Without further qualification, the underlying distance function is the Euclidean metric. In the following we define and study the Voronoi diagram for a given set $P = \{p_1, \ldots, p_n\}$ of points in $\mathbb{R}^2$.

Definition 7.3. *For* $p_i \in P$ *denote the* **Voronoi cell** $V_P(i)$ *of* p_i *by*

$$V_P(i) := \left\{ q \in \mathbb{R}^2 : \|q - p_i\| \leqslant \|q - p\| \text{ for all } p \in P \right\} .$$

Proposition 7.4.

$$V_P(i) = \bigcap_{j \neq i} H(p_i, p_j) .$$

Proof. For $j \neq i$ we have $\|q - p_i\| \leqslant \|q - p_j\| \iff q \in H(p_i, p_j)$. □

Corollary 7.5. $V_P(i)$ *is non-empty and convex.*

Proof. According to Proposition 7.4, the region $V_P(i)$ is the intersection of a finite number of halfplanes and hence convex. As $p_i \in V_P(i)$, we have $V_P(i) \neq \emptyset$. □

Observe that every point of the plane lies in some Voronoi cell but no point lies in the interior of two Voronoi cells. Therefore these cells form a *subdivision* of the plane (a partition[2] into interior-disjoint simple polygons). See Figure 7.3 for an example.

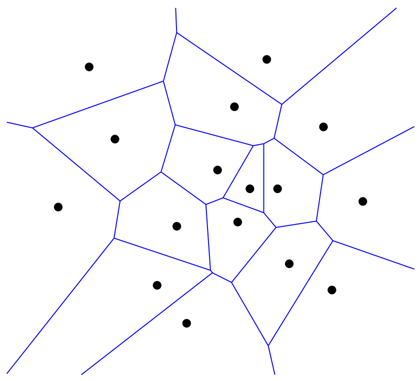


Figure 7.3: *Example: The Voronoi diagram of a point set.*

[2]Strictly speaking, to obtain a partition, we treat the shared boundaries of the polygons as separate entities.

Definition 7.6. *The* **Voronoi Diagram** $\mathrm{VD}(P)$ *of a set* $P = \{p_1, \ldots, p_n\}$ *of points in* $\mathbb{R}^2$ *is the subdivision of the plane induced by the Voronoi cells* $V_P(i)$, *for* $i = 1, \ldots, n$. *Denote by* $\mathrm{VV}(P)$ *the set of vertices, by* $\mathrm{VE}(P)$ *the set of edges, and by* $\mathrm{VR}(P)$ *the set of regions (faces) of* $\mathrm{VD}(P)$.

Lemma 7.7. *For every vertex* $v \in \mathrm{VV}(P)$ *the following statements hold.*

a) v *is the common intersection of at least three edges from* $\mathrm{VE}(P)$*;*

b) v *is incident to at least three regions from* $\mathrm{VR}(P)$*;*

c) v *is the center of a circle* $C(v)$ *through at least three points from* P *such that*

d) $D(v)^\circ \cap P = \emptyset$, *where* $D(v)$ *denotes the disk bounded by* $C(v)$.

Proof. Consider a vertex $v \in \mathrm{VV}(P)$. As all Voronoi cells are convex, $k \geqslant 3$ of them must be incident to v. This proves Part a) and b).

Without loss of generality let these cells be $V_P(i)$, for $1 \leqslant i \leqslant k$; see Figure 7.4. Denote by e_i, $1 \leqslant i \leqslant k$, the edge incident to v that bounds $V_P(i)$ and $V_P((i \bmod k)+1)$.

For any $i = 1, \ldots, k$ we have $v \in e_i \Rightarrow \|v - p_i\| = \|v - p_{(i \bmod k)+1}\|$. In other words, $p_1, p_2, \ldots, p_k$ are cocircular, which proves Part c).

Part d): Suppose there exists a point $p_\ell \in D(v)^\circ$. Then the vertex v is closer to p_ℓ than it is to any of $p_1, \ldots, p_k$, in contradiction to the fact that v is contained in all of $V_P(1), \ldots, V_P(k)$. □

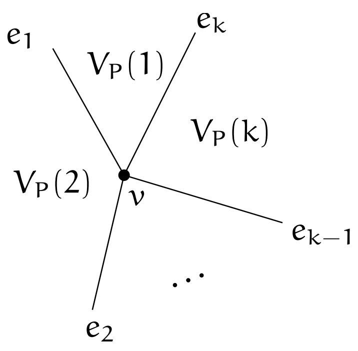


Figure 7.4: *Voronoi regions around* v.

Corollary 7.8. *If* P *is in general position (no four points from* P *are cocircular), then for every vertex* $v \in \mathrm{VV}(P)$ *the following statements hold.*

a) v *is the common intersection of exactly three edges from* $\mathrm{VE}(P)$*;*

b) v *is incident to exactly three regions from* $\mathrm{VR}(P)$*;*

c) v *is the center of a circle* $C(v)$ *through exactly three points from* P *such that* *d)* $D(v)^\circ \cap P = \emptyset$, *where* $D(v)$ *denotes the disk bounded by* $C(v)$. □

Lemma 7.9. *There is an unbounded Voronoi edge bounding* $V_P(i)$ *and* $V_P(j)$ $\iff$ $\overline{p_ip_j} \cap P = \{p_i, p_j\}$ *and* $\overline{p_ip_j} \subseteq \partial\mathrm{conv}(P)$, *where the latter denotes the boundary of the convex hull of* P.

Proof. Denote by $b_{i,j}$ the bisector of p_i and p_j, and let $\mathcal{D}$ denote the family of disks centered at some point on $b_{i,j}$ and passing through p_i (and p_j). There is an unbounded Voronoi edge bounding $V_P(i)$ and $V_P(j)$ $\iff$ there is a ray $\rho \subset b_{i,j}$ such that $\|r-p_k\| > \|r-p_i\|$ $(=\|r-p_j\|)$, for every $r \in \rho$ and every $p_k \in P$ with $k \notin \{i,j\}$. Equivalently, there is a ray $\rho \subset b_{i,j}$ such that for every point $r \in \rho$ the disk $C \in \mathcal{D}$ centered at r does not contain any point from P in its interior (Figure 7.5).

The latter statement implies that the open halfplane H, whose bounding line passes through p_i and p_j and such that H contains the infinite part of ρ, contains no point from P in its interior. Therefore, $\overline{p_ip_j}$ appears on $\partial\mathrm{conv}(P)$ and $\overline{p_ip_j}$ does not contain any $p_k \in P$, for $k \neq i,j$.

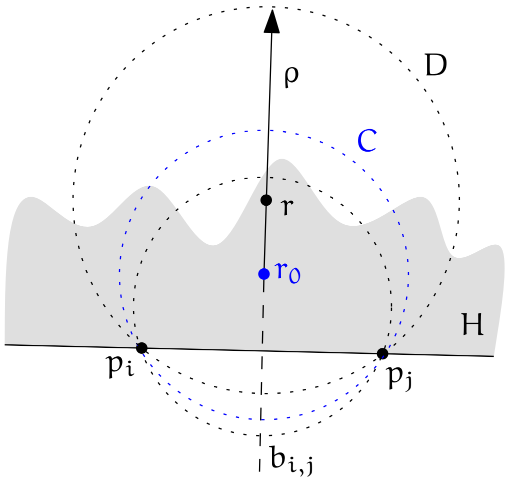


Figure 7.5: *The correspondence between* $\overline{p_ip_j}$ *appearing on* $\partial\mathrm{conv}(P)$ *and a family* $\mathcal{D}$ *of empty disks centered at the bisector of* p_i *and* p_j.

Conversely, suppose that $\overline{p_ip_j}$ appears on $\partial\mathrm{conv}(P)$ and $\overline{p_ip_j} \cap P = \{p_i, p_j\}$. Then some halfplane H whose bounding line passes through p_i and p_j contains no point from P in its interior. In particular, the existence of H together with $\overline{p_ip_j} \cap P = \{p_i, p_j\}$ implies that there is some disk $C \in \mathcal{D}$ such that $C \cap P = \{p_i, p_j\}$. Denote by r_0 the center of C and let ρ denote the ray starting from r_0 along $b_{i,j}$ such that the infinite part of ρ is contained in H. Consider any disk $D \in \mathcal{D}$ centered at a point $r \in \rho$ and observe that $D \setminus H \subseteq C \setminus H$. As neither H nor C contain any point from P in their respective interior, neither does D. This holds for every D, and we have seen above that this statement is equivalent to the existence of an unbounded Voronoi edge bounding $V_P(i)$ and $V_P(j)$. □

7.3 Duality

A *straight-line dual* of a plane graph G is a graph G' defined as follows: Choose a point for each face of G and connect any two such points by a straight edge, if the corresponding faces share an edge of G. Observe that this notion depends on the embedding; that is why the straight-line dual is defined for a plane graph rather than for an abstract graph. In general, G' may have edge crossings, which may also depend on the choice of representative points within the faces. However, for Voronoi diagrams there is a particularly natural choice of representative points such that G' is plane: the points from P.

Theorem 7.10 (Delaunay [3]). *The straight-line dual of* $\mathrm{VD}(P)$ *for a set* $P \subset \mathbb{R}^2$ *of* $n \geqslant 3$ *points in general position (no three points from* P *are collinear and no four points from* P *are cocircular) is a triangulation: the unique Delaunay triangulation of* P.

Proof. By Lemma 7.9, the convex hull edges appear in the straight-line dual T of $\mathrm{VD}(P)$ and they correspond exactly to the unbounded edges of $\mathrm{VD}(P)$. All remaining edges of $\mathrm{VD}(P)$ are bounded, that is, both endpoints are Voronoi vertices. Consider some $v \in \mathrm{VV}(P)$. According to Corollary 7.8(b), v is incident to exactly three Voronoi regions, which, therefore, form a triangle $\triangle(v)$ in T. By Corollary 7.8(d), the circumcircle of $\triangle(v)$ does not contain any point from P in its interior. Hence $\triangle(v)$ appears in the (unique by Corollary 5.19) Delaunay triangulation of P.

Conversely, for any triangle $p_i p_j p_k$ in the Delaunay triangulation of P, by the empty circle property the circumcenter c of $p_i p_j p_k$ has p_i, p_j, and p_k as its closest points from P. Therefore, $c \in \mathrm{VV}(P)$ and—as above—the triangle $p_i p_j p_k$ appears in T. □

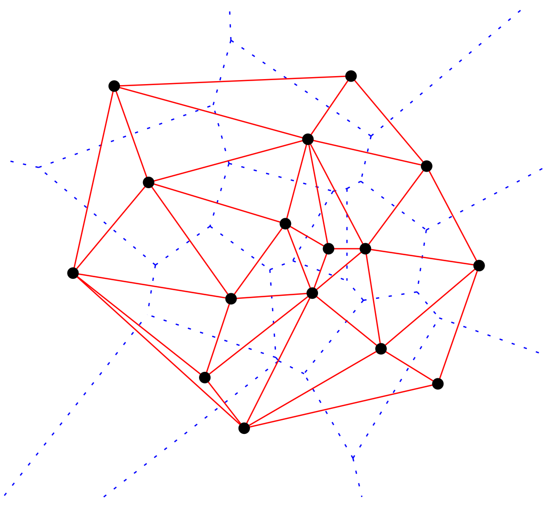


Figure 7.6: *The Voronoi diagram of a point set and its dual Delaunay triangulation.*

It is not hard to generalize Theorem 7.10 to general point sets. In this case, a Voronoi vertex of degree k is mapped to a convex polygon with k cocircular vertices. Any triangulation of such a polygon yields a Delaunay triangulation of the point set.

Corollary 7.11. $|\mathrm{VE}(P)| \leqslant 3n - 6$ *and* $|\mathrm{VV}(P)| \leqslant 2n - 5$.

Proof. Every edge in $\mathrm{VE}(P)$ corresponds to an edge in the dual Delaunay triangulation. The latter is a plane graph on n vertices, which by Corollary 2.5 has at most $3n-6$ edges and at most $2n-4$ faces. Only the bounded faces correspond to a vertex in $\mathrm{VD}(P)$. □

Corollary 7.12. *For a set* $P \subset \mathbb{R}^2$ *of* n *points, the Voronoi diagram of* P *can be constructed in expected* $O(n \log n)$ *time and* $O(n)$ *space.*

Proof. We have seen that a Delaunay triangulation T for P can be obtained using randomized incremental construction in the given time and space bounds. As T is a plane graph, its number of vertices, edges, and faces all are linear in n. Therefore, the straight-line dual of T—which by Theorem 7.10 is the desired Voronoi diagram—can be computed in $O(n)$ additional time and space. □

Exercise 7.13. *Consider the Delaunay triangulation* T *for a set* $P \subset \mathbb{R}^2$ *of* $n \geqslant 3$ *points in general position. Prove or disprove:*

a) Every edge of T *intersects its dual Voronoi edge.*

b) Every vertex of $\mathrm{VD}(P)$ *is contained in its dual Delaunay triangle.*

Exercise 7.14. *Given a plane graph that forms the Voronoi diagram a some unknown point set* P. *Can you compute* P *along with the Delaunay triangulation of* P *in linear time?*

7.4 Lifting Map

Recall the lifting map that we used in Section 5.3 to prove that the Lawson Flip Algorithm terminates. Denote by $\mathcal{U}\colon z = x^2 + y^2$ the unit paraboloid in $\mathbb{R}^3$. The lifting map $\ell\colon \mathbb{R}^2 \to \mathcal{U}$ with $\ell\colon p = (p_x, p_y) \mapsto (p_x, p_y, {p_x}^2 + {p_y}^2)$ is the projection of the x/y-plane onto $\mathcal{U}$ in direction of the z-axis.

For $p \in \mathbb{R}^2$ let H_p denote the plane of tangency to $\mathcal{U}$ in $\ell(p)$. Denote by $h_p\colon \mathbb{R}^2 \to H_p$ the projection of the x/y-plane onto H_p in direction of the z-axis (see Figure 7.7).

Lemma 7.15. $\|\ell(q) - h_p(q)\| = \|p - q\|^2$, *for any points* $p, q \in \mathbb{R}^2$.

Exercise 7.16. *Prove Lemma 7.15.* Hint: *First determine the equation of the tangent plane* H_p *to* $\mathcal{U}$ *in* $\ell(p)$.

Theorem 7.17. *For* $p = (p_x, p_y) \in \mathbb{R}^2$ *denote by* H_p *the plane of tangency to the unit paraboloid* $\mathcal{U} = \{(x, y, z) : z = x^2 + y^2\} \subset \mathbb{R}^3$ *in* $\ell(p) = (p_x, p_y, {p_x}^2 + {p_y}^2)$. *Let* $\mathcal{H}(P) := \bigcap_{p \in P} H_p^+$ *denote the intersection of all halfspaces above the planes* H_p, *for* $p \in P$. *Then the vertical projection of* $\partial\mathcal{H}(P)$ *onto the* x/y*-plane forms the Voronoi Diagram of* P *(the faces of* $\partial\mathcal{H}(P)$ *correspond to Voronoi regions, the edges to Voronoi edges, and the vertices to Voronoi vertices).*

Proof. For any point $q \in \mathbb{R}^2$, the vertical line through q intersects every plane H_p, $p \in P$. By Lemma 7.15 the topmost plane intersected belongs to the point from P that is closest to q. □

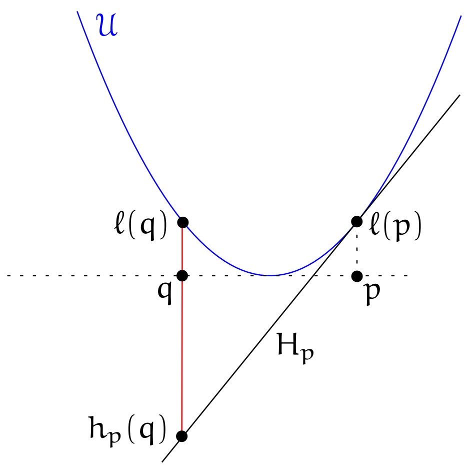


Figure 7.7: *Lifting map interpretation of the Voronoi diagram in a two-dimensional projection.*

7.5 Planar Point Location

One last bit is still missing in order to solve the post office problem optimally.

Theorem 7.18. *Given a triangulation* T *for a set* $P \subset \mathbb{R}^2$ *of* n *points, one can build in* $O(n)$ *time an* $O(n)$ *size data structure that allows for any query point* $q \in \text{conv}(P)$ *to find in* $O(\log n)$ *time a triangle from* T *containing* q.

The data structure we will employ is known as *Kirkpatrick's hierarchy.* But before discussing it in detail, let us put things together in terms of the post office problem.

Corollary 7.19 (Nearest Neighbor Search). *Given a set* $P \subset \mathbb{R}^2$ *of* n *points, one can build in expected* $O(n \log n)$ *time an* $O(n)$ *size data structure that allows for any query point* $q \in \text{conv}(P)$ *to find in* $O(\log n)$ *time a nearest neighbor of* q *among the points from* P.

Proof. First construct the Voronoi Diagram V of P in expected $O(n \log n)$ time. It has exactly n convex faces. Every unbounded face can be cut by the convex hull boundary into a bounded and an unbounded part. As we are concerned with query points within $\text{conv}(P)$ only, we can restrict our attention to the bounded parts.[3] Any convex polygon can easily be triangulated in time linear in its number of edges (= number of vertices). As V has at most $3n-6$ edges and every edge appears in exactly two faces, V can be triangulated in $O(n)$ time overall. Label each of the resulting triangles with the point from p, whose Voronoi region contains it, and apply the data structure from Theorem 7.18. $\square$

[3]We even know how to decide in $O(\log n)$ time whether or not a given point lies within $\text{conv}(P)$, see Exercise 4.25.

7.6 Kirkpatrick's Hierarchy

We will now develop a data structure for point location in a triangulation, as described in Theorem 7.18. For simplicity we assume that the triangulation T we work with is a maximal planar graph, that is, the outer face is a triangle as well. This can easily be achieved by an initial normalization step that puts a huge triangle T_h around T and triangulates the region in between T_h and T (in linear time—how?).

The main idea for the data structure is to construct a hierarchy $T_0,\ldots,T_h$ of triangulations, such that

- $T_0 = T$,
- the vertices of T_i are a subset of the vertices of T_{i-1}, for $i = 1, \ldots, h$, and
- T_h is a single triangle only.

Search. For a query point x we can find a triangle from T that contains x as follows.

```
Search($x \in \mathbb{R}^2$)
1. For $i = h, h-1, \ldots, 0$: Find a triangle $t_i$ from $T_i$ that contains $x$.
2. return $t_0$.
```

This search is efficient under the following conditions.

(C1) Every triangle from T_i intersects only few ($\leqslant c$) triangles from T_{i-1}. (These will then be connected via the data structure.)

(C2) h is small ($\leqslant d \log n$).

Proposition 7.20. *The search procedure described above needs* $\leqslant 3cd \log n = O(\log n)$ *orientation tests.*

Proof. For every T_i, $0 \leqslant i < h$, at most c triangles are tested as to whether or not they contain x. Using three orientation tests one can determine whether or not a triangle contains a given point. □

Thinning. Removing a vertex v and all its incident edges from a triangulation creates a non-triangulated hole that forms a star-shaped polygon since all points are visible from v (the star-point). Here we remove vertices of constant degree only and therefore these polygons are of constant size. But even if they were not, it is not hard to triangulate a star-shaped polygon in linear time.

Lemma 7.21. *A star-shaped polygon, given as a sequence of* $n \geqslant 3$ *vertices and a star-point, can be triangulated in* $O(n)$ *time.*

Exercise 7.22. *Prove Lemma 7.21.*

As a side remark, the *kernel* of a simple polygon, that is, the (possibly empty) set of all star-points, can be constructed in linear time as well [8]. A point in the kernel can also be found using linear programming.

Our working plan is to obtain T_i from T_{i-1} by removing several *independent* (pairwise non-adjacent) vertices and re-triangulating. These vertices should

a) have small degree (otherwise the degree within the hierarchy gets too large, that is, we need to test too many triangles on the next level) and

b) be many (otherwise the height h of the hierarchy gets too large).

The following lemma asserts the existence of a sufficiently large set of independent small-degree vertices in every triangulation.

Lemma 7.23. *In every triangulation of n points in $\mathbb{R}^2$ there exists an independent set of at least $\lceil n/18 \rceil$ vertices of maximum degree 8. Moreover, such a set can be found in $O(n)$ time.*

Proof. Let $T = (V, E)$ denote the graph of the triangulation, which we consider as an abstract graph in the following. We may suppose that T is maximal planar, that is, the outer face is a triangle. (Otherwise use Theorem 2.27 to combinatorially triangulate T arbitrarily. An independent set in the resulting graph T' is also independent in T and the degree of a vertex in T' is at least as large as its degree in T.) For $n = 3$ the statement is true. Let $n \geqslant 4$.

By the Euler formula we have $|E| = 3n - 6$, that is,

$$\sum_{v \in V} \deg_T(v) = 2|E| = 6n - 12 < 6n.$$

Let $W \subseteq V$ denote the set of vertices of degree at most 8. Claim: $|W| > n/2$. Suppose $|W| \leqslant n/2$. By Theorem 2.28 we know that T is 3-connected and so every vertex has degree at least three. Therefore

$$\begin{aligned}\sum_{v \in V} \deg_T(v) &= \sum_{v \in W} \deg_T(v) + \sum_{v \in V \setminus W} \deg_T(v) \geqslant 3|W| + 9|V \setminus W| \\ &= 3|W| + 9(n - |W|) = 9n - 6|W| \geqslant 9n - 3n = 6n,\end{aligned}$$

in contradiction to the above.

Construct an independent set U in T as follows (greedily): As long as $W \neq \emptyset$, add an arbitrary vertex $v \in W$ to U and remove v and all its neighbors from W. Assuming that T is represented so that we can obtain the neighborhood of a vertex v in $\deg_T(v)$ time (for instance, using adjacency lists), both W and U can be computed in $O(n)$ time.

Obviously U is independent and all vertices in U have degree at most 8. At each selection step at most 9 vertices are removed from W. Therefore $|U| \geqslant \lceil (n/2)/9 \rceil = \lceil n/18 \rceil$. □

Proof. (of Theorem 7.18)
Construct the hierarchy $T_0, \ldots T_h$ with $T_0 = T$ as follows. Obtain T_i from T_{i-1} by removing an independent set U as in Lemma 7.23 and re-triangulating the resulting holes. By Lemma 7.21 and Lemma 7.23 every step is linear in the number $|T_i|$ of vertices in T_i. The total cost for building the data structure is thus

$$\sum_{i=0}^{h} \alpha|T_i| \leqslant \sum_{i=0}^{h} \alpha n \left(1 - \frac{1}{18}\right)^i \leqslant \sum_{i=0}^{h} \alpha n(17/18)^i < \alpha n \sum_{i=0}^{\infty} (17/18)^i = 18\alpha n \in O(n),$$

for some constant α. Similarly the space consumption is linear.

The number of levels amounts to $h = \log_{18/17} n < 12.2 \log n$. Thus by Proposition 7.20 the search needs at most $3 \cdot 8 \cdot \log_{18/17} n < 292 \log n$ orientation tests. □

Improvements. As the name suggests, the hierarchical approach discussed above is due to David Kirkpatrick [6]. The constant 292 that appears in the search time is somewhat large. There has been a whole line of research trying to improve it using different techniques.

- Sarnak and Tarjan [9]: $4 \log n$.
- Edelsbrunner, Guibas, and Stolfi [4]: $3 \log n$.
- Goodrich, Orletsky, and Ramaiyer [5]: $2 \log n$.
- Adamy and Seidel [1]: $1 \log n + 2\sqrt{\log n} + O(\sqrt[4]{\log n})$.

Exercise 7.24. *Let $\{p_1, p_2, \ldots, p_n\}$ be a set of points in the plane, which we call* obstacles*. Imagine there is a disk of radius r centered at the origin which can be moved around the obstacles but is not allowed to intersect them (touching the boundary is ok). Is it possible to move the disk out of these obstacles? See the example depicted in Figure 7.8 below.*

More formally, the question is whether there is a (continuous) path $\gamma : [0,1] \longrightarrow \mathbb{R}^2$ with $\gamma(0) = (0,0)$ and $\|\gamma(1)\| \geqslant \max\{\|p_1\|, \ldots, \|p_n\|\}$, such that at any time $t \in [0,1]$ and $\|\gamma(t) - p_i\| \geqslant r$, for any $1 \leqslant i \leqslant n$. Describe an algorithm to decide this question and to construct such a path—if one exists—given arbitrary points $\{p_1, p_2, \ldots, p_n\}$ and a radius $r > 0$. Argue why your algorithm is correct and analyze its running time.

Exercise 7.25. *This exercise is about an application from* Computational Biology*: You are given a set of disks $P = \{a_1, .., a_n\}$ in $\mathbb{R}^2$, all with the same radius $r_a > 0$. Each of these disks represents an atom of a protein. A water molecule is represented by a disc with radius $r_w > r_a$. A water molecule cannot intersect the interior of any protein atom, but it can be tangent to one. We say that an atom $a_i \in P$ is* accessible *if there exists a placement of a water molecule such that it is tangent to a_i and does not intersect the interior of any other atom in P. Given P, find an $O(n \log n)$ time algorithm which determines all atoms of P that are inaccessible.*

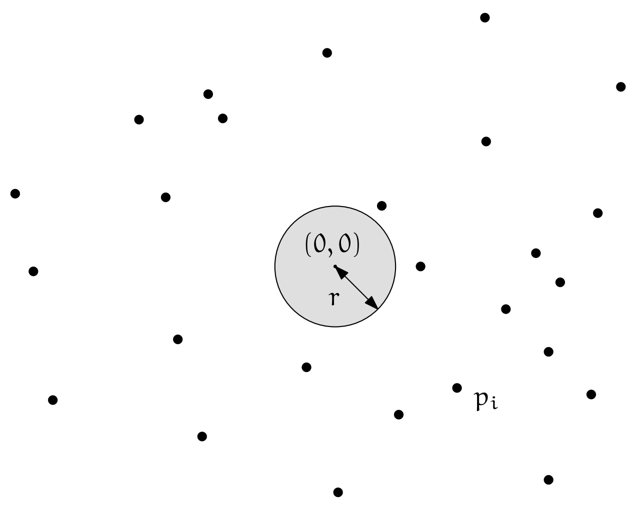


Figure 7.8: *Motion planning: Illustration for Exercise 7.24.*

Exercise 7.26. *Let* $P \subset \mathbb{R}^2$ *be a set of* n *points. Describe a data structure to find in* $O(\log n)$ *time a point in* P *that is furthest from a given query point* q *among all points in* P.

Exercise 7.27. *Show that the bounds given in Theorem 7.18 are optimal in the algebraic computation tree model.*

Questions

31. *What is the Voronoi diagram of a set of points in* $\mathbb{R}^2$*?* Give a precise definition and explain/prove the basic properties: convexity of cells, why is it a subdivision of the plane?, Lemma 7.7, Lemma 7.9.
32. *What is the correspondence between the Voronoi diagram and the Delaunay triangulation for a set of points in* $\mathbb{R}^2$*?* Prove duality (Theorem 7.10) and explain where general position is needed.
33. *How to construct the Voronoi diagram of a set of points in* $\mathbb{R}^2$*?* Describe an $O(n \log n)$ time algorithm, for instance, via Delaunay triangulation.
34. *How can the Voronoi diagram be interpreted in context of the lifting map?* Describe the transformation and prove its properties to obtain a formulation of the Voronoi diagram as an intersection of halfspaces one dimension higher.
35. *What is the Post-Office Problem and how can it be solved optimally?* Describe the problem and a solution using linear space, $O(n \log n)$ preprocessing, and $O(\log n)$ query time.
36. *How does Kirkpatrick's hierarchical data structure for planar point location work exactly?* Describe how to build it and how the search works, and prove the

runtime bounds. In particular, you should be able to state and prove Lemma 7.23 and Theorem 7.18.

References

[1] Udo Adamy and Raimund Seidel, On the exaxt worst case query complexity of planar point location. *J. Algorithms*, **37**, (2000), 189–217.

[2] Franz Aurenhammer, Voronoi diagrams: A survey of a fundamental geometric data structure. *ACM Comput. Surv.*, **23**, 3, (1991), 345–405.

[3] Boris Delaunay, Sur la sphère vide. A la memoire de Georges Voronoi. *Izv. Akad. Nauk SSSR, Otdelenie Matematicheskih i Estestvennyh Nauk*, **6**, (1934), 793–800.

[4] Herbert Edelsbrunner, Leonidas J. Guibas, and Jorge Stolfi, Optimal point location in a monotone subdivision. *SIAM J. Comput.*, **15**, 2, (1986), 317–340.

[5] Michael T. Goodrich, Mark W. Orletsky, and Kumar Ramaiyer, Methods for achieving fast query times in point location data structures. In *Proc. 8th ACM-SIAM Sympos. Discrete Algorithms*, pp. 757–766, 1997.

[6] David G. Kirkpatrick, Optimal search in planar subdivisions. *SIAM J. Comput.*, **12**, 1, (1983), 28–35.

[7] Rolf Klein, *Concrete and abstract Voronoi diagrams*, vol. 400 of *Lecture Notes Comput. Sci.*, Springer, 1989.

[8] Der-Tsai Lee and Franco P. Preparata, An optimal algorithm for finding the kernel of a polygon. *J. ACM*, **26**, 3, (1979), 415–421.

[9] Neil Sarnak and Robert E. Tarjan, Planar point location using persistent search trees. *Commun. ACM*, **29**, 7, (1986), 669–679.

Chapter 8

Line Arrangements

During the course of this lecture we encountered several situations where it was convenient to assume that a point set is "in general position". In the plane, general position usually amounts to no three points being collinear and/or no four of them being cocircular. This raises an algorithmic question: How can we test for n given points whether or not three of them are collinear? Obviously, we can test all triples in $O(n^3)$ time. Can we do better? Yes, we can! Using a detour through the so-called dual plane, we will see that this problem can be solved in $O(n^2)$ time. However, the exact algorithmic complexity of this innocent-looking problem is not known. In fact, to determine this complexity is one of the major open problems in theoretical computer science.

We will get back to the complexity theoretic problems and ramifications at the end of this chapter. But first let us discuss how to obtain a quadratic time algorithm to test whether n given points in the plane are in general position. This algorithm is a nice application of the projective duality transform, as defined below. Such transformations are very useful because they allow us to gain a new perspective on a problem by formulating it in a different but equivalent form. Sometimes such a *dual* form of the problem is easier to work with and—given that it is equivalent to the original *primal* form—any solution to the dual problem can be translated back into a solution to the primal problem.

So what is this duality transform about? Observe that points and hyperplanes in $\mathbb{R}^d$ are very similar objects, given that both can be described using d coordinates/parameters. It is thus tempting to match these parameters to each other and so create a mapping between points and hyperplanes. In $\mathbb{R}^2$ hyperplanes are lines and the standard *projective duality* transform maps a point $p = (p_x, p_y)$ to the line $p^* : y = p_x x - p_y$ and a non-vertical line $g : y = mx + b$ to the point $g^* = (m, -b)$.

Proposition 8.1. *The standard projective duality transform is*

- *incidence preserving:* $p \in g \iff g^* \in p^*$ *and*
- *order preserving:* p *is above* $g \iff g^*$ *is above* p^*.

Exercise 8.2. *Prove Proposition 8.1.*

Exercise 8.3. *Describe the image of the following point sets under this mapping*

a) a halfplane

b) $k \geqslant 3$ *collinear points*

c) a line segment

d) the boundary points of the upper convex hull of a finite point set.

Another way to think of duality is in terms of the parabola $\mathcal{P} : y = \frac{1}{2}x^2$. For a point p on $\mathcal{P}$, the dual line p^* is the tangent to $\mathcal{P}$ at p. For a point p not on $\mathcal{P}$, consider the vertical projection p' of p onto $\mathcal{P}$: the slopes of p^* and p'^* are the same, just p^* is shifted by the difference in y-coordinates.

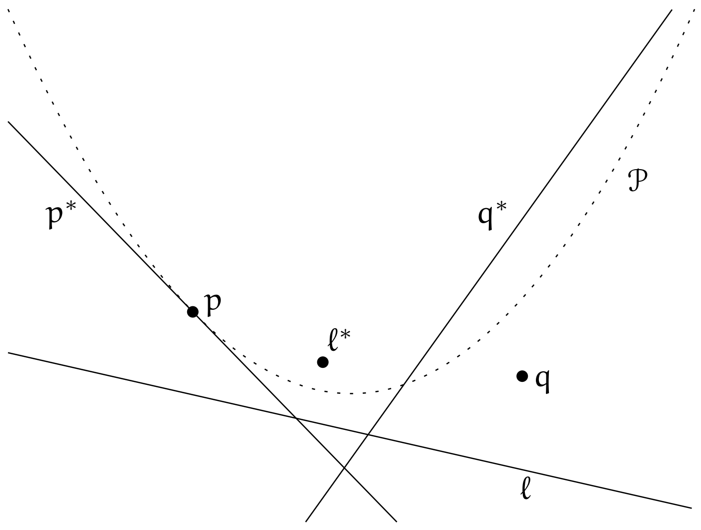


Figure 8.1: *Point* $\leftrightarrow$ *line duality with respect to the parabola* $\mathcal{P} : y = \frac{1}{2}x^2$.

The question of whether or not three points in the primal plane are collinear transforms to whether or not three lines in the dual plane meet in a point. This question in turn we will answer with the help of *line arrangements*, as defined below.

8.1 Arrangements

The subdivision of the plane induced by a finite set L of lines is called the **arrangement** $\mathcal{A}(L)$. We may imagine the creation of this subdivision as a recursive process, defined by the given set L of lines. As a first step, remove all lines (considered as point sets) from the plane $\mathbb{R}^2$. What remains of $\mathbb{R}^2$ are a number of open connected components (possibly only one), which we call the (2-dimensional) **cells** of the subdivision. In the next step, from every line in L remove all the remaining lines (considered as point sets). In this way every line is split into a number of open connected components (possibly only one), which collectively form the (1-dimensional cells or) **edges** of the subdivision. What remains of the lines are the (0-dimensional cells or) **vertices** of the subdivision, which are intersection points of lines from L.

Observe that all cells of the subdivision are intersections of halfplanes and thus convex. A line arrangement is **simple** if no two lines are parallel and no three lines meet in a point. Although lines are unbounded, we can regard a line arrangement a bounded object by (conceptually) putting a sufficiently large box around that contains all vertices. Such a box can be constructed in $O(n \log n)$ time for n lines.

Exercise 8.4. *How?*

Moreover, we can view a line arrangement as a planar graph by adding an additional vertex at "infinity", that is incident to all rays which leave this bounding box. For algorithmic purposes, we will mostly think of an arrangement as being represented by a doubly connected edge list (DCEL), cf. Section 2.2.1.

Theorem 8.5. *A simple arrangement $\mathcal{A}(L)$ of n lines in $\mathbb{R}^2$ has $\binom{n}{2}$ vertices, n^2 edges, and $\binom{n}{2} + n + 1$ faces/cells.*

Proof. Since all lines intersect and all intersection points are pairwise distinct, there are $\binom{n}{2}$ vertices.

The number of edges we count using induction on n. For $n = 1$ we have $1^2 = 1$ edge. By adding one line to an arrangement of $n - 1$ lines we split $n - 1$ existing edges into two and introduce n new edges along the newly inserted line. Thus, there are in total $(n-1)^2 + 2n - 1 = n^2 - 2n + 1 + 2n - 1 = n^2$ edges.

The number f of faces can now be obtained from Euler's formula $v - e + f = 2$, where v and e denote the number of vertices and edges, respectively. However, in order to apply Euler's formula we need to consider $\mathcal{A}(L)$ as a planar graph and take the symbolic "infinite" vertex into account. Therefore,

$$f = 2 - \left(\binom{n}{2} + 1\right) + n^2 = 1 + \frac{1}{2}(2n^2 - n(n-1)) = 1 + \frac{1}{2}(n^2 + n) = 1 + \binom{n}{2} + n\,.$$

□

The **complexity** of an arrangement is simply the total number of vertices, edges, and faces (in general, cells of any dimension).

Exercise 8.6. *Consider a set of lines in the plane with no three intersecting in a common point. Form a graph G whose vertices are the intersection points of the lines and such that two vertices are adjacent if and only if they appear consecutively along one of the lines. Prove that $\chi(G) \leqslant 3$, where $\chi(G)$ denotes the chromatic number of the graph G. In other words, show how to color the vertices of G using at most three colors such that no two adjacent vertices have the same color.*

8.2 Construction

As the complexity of a line arrangement is quadratic, there is no need to look for a subquadratic algorithm to construct it. We will simply construct it incrementally, inserting the lines one by one. Let $\ell_1, \dots, \ell_n$ be the order of insertion.

At Step i of the construction, locate ℓ_i in the leftmost cell of $\mathcal{A}(\{\ell_1, \dots, \ell_{i-1}\})$ it intersects. (The halfedges leaving the infinite vertex are ordered by slope.) This takes $O(i)$ time. Then traverse the boundary of the face F found until the halfedge h is found where ℓ_i leaves F (see Figure 8.2 for illustration). Insert a new vertex at this point, splitting F and h and continue in the same way with the face on the other side of h.

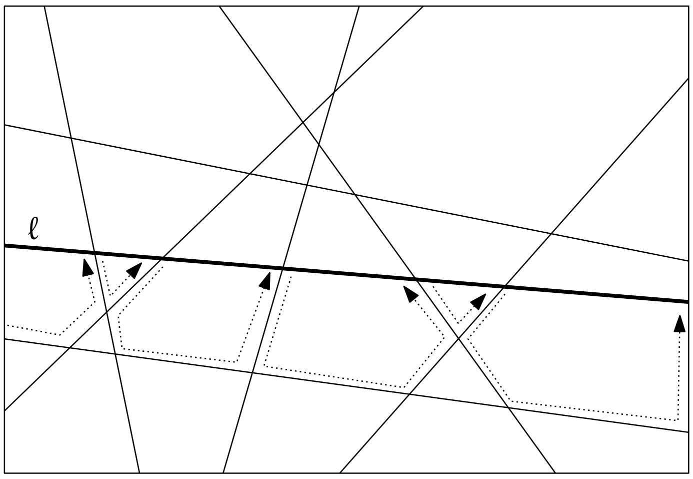


Figure 8.2: *Incremental construction: Insertion of a line ℓ. (Only part of the arrangement is shown in order to increase readability.)*

The insertion of a new vertex involves splitting two halfedges and thus is a constant time operation. But what is the time needed for the traversal? The complexity of $\mathcal{A}(\{\ell_1, \dots, \ell_{i-1}\})$ is $\Theta(i^2)$, but we will see that the region traversed by a single line has linear complexity only.

8.3 Zone Theorem

For a line ℓ and an arrangement $\mathcal{A}(L)$, the **zone** $Z_{\mathcal{A}(L)}(\ell)$ of ℓ in $\mathcal{A}(L)$ is the set of cells from $\mathcal{A}(L)$ whose closure intersects ℓ.

Theorem 8.7. *Given an arrangement $\mathcal{A}(L)$ of n lines in $\mathbb{R}^2$ and a line ℓ (not necessarily from L), the total number of edges in all cells of the zone $Z_{\mathcal{A}(L)}(\ell)$ is at most $10n$.*

Proof. Without loss of generality suppose that ℓ is horizontal (rotate the plane accordingly). For each cell of $Z_{\mathcal{A}(L)}(\ell)$ split its boundary at its topmost vertex and at its bottommost vertex and orient all edges from bottom to top, horizontal edges from left to right. Those edges that have the cell to their right are called *left-bounding* for the cell and those edges that have the cell to their left are called right-bounding. For instance, for the cell depicted in Figure 8.3, all left-bounding edges are shown blue and bold.

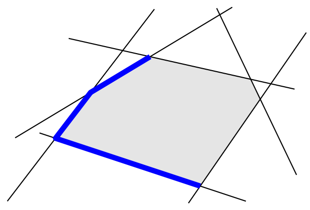


Figure 8.3: *Left-bounding edges (blue and bold) of a cell.*

We will show that there are at most $5n$ left-bounding edges in $Z_{\mathcal{A}(L)}(\ell)$ by induction on n. By symmetry, the same bound holds also for the number of right-bounding edges in $Z_{\mathcal{A}(L)}(\ell)$.

For $n = 1$, there is at most one (exactly one, unless ℓ is parallel to and lies above the only line in L) left-bounding edge in $Z_{\mathcal{A}(L)}(\ell)$ and $1 \leqslant 5n = 5$. Assume the statement is true for $n - 1$.

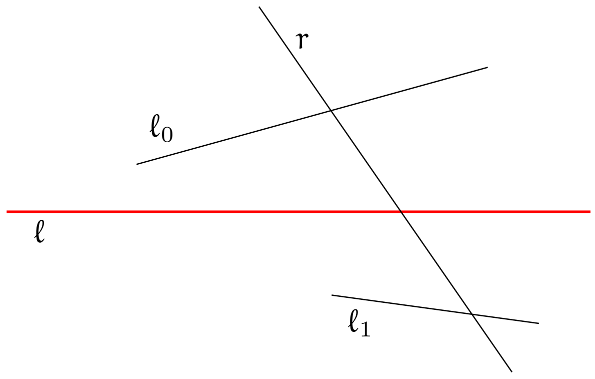


Figure 8.4: *At most three new left-bounding edges are created by adding* r *to* $\mathcal{A}(L\backslash\{r\})$.

If no line from L intersects ℓ, then all lines in $L \cup \{\ell\}$ are horizontal and there is at most $1 < 5n$ left-bounding edge in $Z_{\mathcal{A}(L)}(\ell)$. Else assume first that there is a single rightmost line r from L intersecting ℓ and the arrangement $\mathcal{A}(L \setminus \{r\})$. By the induction hypothesis there are at most $5n - 5$ left-bounding edges in $Z_{\mathcal{A}(L\setminus\{r\})}(\ell)$. Adding r back adds at most three new left-bounding edges: At most two edges (call them ℓ_0 and ℓ_1) of the rightmost cell of $Z_{\mathcal{A}(L\setminus\{r\})}(\ell)$ are intersected by r and thereby split in two. Both of these two edges may be left-bounding and thereby increase the number of left-bounding edges by at most two. In any case, r itself contributes exactly one more left-bounding edge to that cell. The line r cannot contribute a left-bounding edge to any cell other than the rightmost: to the left of r, the edges induced by r form right-bounding edges only and to the right of r all other cells touched by r (if any) are shielded away from ℓ by one of ℓ_0 or ℓ_1. Therefore, the total number of left-bounding edges in $Z_{\mathcal{A}(L)}(\ell)$ is bounded from above by $3 + 5n - 5 < 5n$.

If there are several rightmost lines that intersect ℓ in the same point, we consider these lines in an arbitrary order. Using the same line of arguments as in the previous case, it can be observed that we add at most five left-bounding edges when adding a line r' after having added a line r, where both r and r' pass through the rightmost intersection point on ℓ. Apart from r, the line r' intersects at most two left-bounding edges ℓ_0 and ℓ_1 of cells in the zone of ℓ. There are two new left-bounding segments on r', and at most one additional on r. Hence, the number of left-bounding edges in this case is at most $5 + 5n - 5 = 5n$. □

Corollary 8.8. *The arrangement of n lines in $\mathbb{R}^2$ can be constructed in optimal $O(n^2)$ time and space.*

Proof. Use the incremental construction described above. In Step i, for $1 \leqslant i \leqslant n$, we do a linear search among $i - 1$ elements to find the starting face and then traverse (part of) the zone of the line ℓ_i in the arrangement $\mathcal{A}(\{\ell_1, \dots, \ell_{i-1}\})$. By Theorem 8.7 the complexity of this zone and hence the time complexity of Step i altogether is $O(i)$. Overall we obtain $\sum_{i=1}^{n} ci = O(n^2)$ time (and space), for some constant $c > 0$, which is optimal by Theorem 8.5. □

The corresponding bounds for hyperplane arrangements in $\mathbb{R}^d$ are $\Theta(n^d)$ for the complexity of a simple arrangement and $O(n^{d-1})$ for the complexity of a zone of a hyperplane.

Exercise 8.9. *For an arrangement $\mathcal{A}$ of a set of n lines in $\mathbb{R}^2$, let*

$$\mathcal{F} := \bigcup_{C \text{ is a bounded cell of } \mathcal{A}} \overline{C}$$

denote the union of the closure of all bounded cells. Show that the complexity (number of vertices and edges of the arrangement lying on the boundary) of $\mathcal{F}$ is $O(n)$.

8.4 The Power of Duality

The real beauty and power of line arrangements becomes apparent in context of projective point $\leftrightarrow$ line duality. It is often convenient to assume that no two points in the primal have the same x-coordinate so that no line defined by any two points is vertical (and hence becomes an infinite point in the dual). This degeneracy can be tested for by sorting according to x-coordinate (in $O(n \log n)$ time) and resolved by rotating the whole plane by some sufficiently small angle. In order to select the rotation angle it is enough to determine the line of maximum absolute slope that passes through two points. Then we can take, say, half of the angle between such a line and the vertical direction. As the line of maximum slope through any given point can be found in linear time, the overall maximum can be obtained in $O(n^2)$ time.

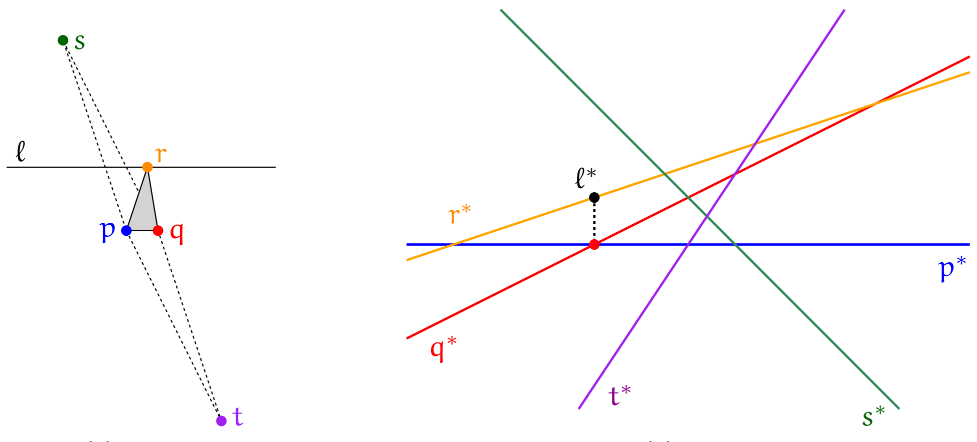
The following problems can be solved in $O(n^2)$ time and space by constructing the dual arrangement.

General position test. Given n points in $\mathbb{R}^2$, are any three of them collinear? (Dual: do any three lines of the dual arrangement meet in a single point?)

Minimum area triangle. Given a set $P \subset \mathbb{R}^2$ of n points, what is the minimum area triangle spanned by any three (pairwise distinct) points of P? Let us make the problem easier by fixing two distinct points $p, q \in P$ and ask for a minimum area triangle pqr, where $r \in P \setminus \{p, q\}$. With pq fixed, the area of pqr is determined by the distance between r and the line pq. Thus, we want to find a point $r \in P \setminus \{p, q\}$ of minimum distance to pq. Equivalently, we want to find

a closest line ℓ parallel to pq so that ℓ passes through some point $r \in P \setminus \{p, q\}$. $(\star)$

Consider the set $P^* = \{p^* : p \in P\}$ of dual lines and their arrangement $\mathcal{A}$. In $\mathcal{A}$ the statement $(\star)$ translates to "a closest point ℓ^* with the same x-coordinate as the vertex $p^* \cap q^*$ of $\mathcal{A}$ that lies on some line $r^* \in P^*$." See Figure 8.5 for illustration.



(a) primal **(b)** dual

Figure 8.5: *Minimum area triangle spanned by two fixed points* p, q.

In other words, for the vertex $v = p^* \cap q^*$ of $\mathcal{A}$ we want to know what is a first line from P^* that is hit by a vertical ray—upward or downward—emanating from v. Of course, in the end we want this information not only for such a single vertex (which provides the minimum area triangle for fixed p, q) but for *all* vertices of $\mathcal{A}$, that is, for all possible pairs of fixed vertices $p, q \in \binom{P}{2}$. Luckily, all this information can easily be maintained over the incremental construction of $\mathcal{A}$. When inserting a line ℓ, this new line may become the first line hit by some vertical rays from vertices of the already computed partial arrangement. However, only vertices in the zone of ℓ may be affected. This zone is traversed, anyway, during the insertion of ℓ. So, during the traversal we can also check possibly update the information for vertices that lie vertically above or below a new edge of the arrangement, with no extra cost asymptotically.

In this way obtain $O(n^2)$ candidate triangles by constructing the arrangement of the n dual lines in $O(n^2)$ time. The smallest among those candidates can be determined by a straightforward minimum selection (comparing the area of the corresponding triangles).

Exercise 8.10. *A set* P *of* n *points in the plane is said to be in* ε-general position *for* $\varepsilon > 0$ *if no three points of the form*

$$p + (x_1, y_1), q + (x_2, y_2), r + (x_3, y_3)$$

are collinear, where $p, q, r \in P$ *and* $|x_i|, |y_i| < \varepsilon$*, for* $i \in \{1, 2, 3\}$*. In words:* P *remains in general position under changing point coordinates by less than* ε *each.*

Give an algorithm with runtime $O(n^2)$ *for checking whether a given point set* P *is in* ε*-general position.*

8.5 Rotation Systems—Sorting all Angular Sequences

Recall the notion of a combinatorial embedding from Chapter 2. It is specified by the circular order of edges along the boundary of each face or—equivalently, dually—around each vertex. In a similar way we can also give a combinatorial description of the geometry of a finite point set $P \subset \mathbb{R}^2$ using its **rotation system**. This is nothing else but a combinatorial embedding of the complete geometric (straight line) graph on P, specified by the circular order of edges around vertices.[1]

For a given set P of n points, it is trivial to construct the corresponding rotation system in $O(n^2 \log n)$ time, by sorting each of the n lists of neighbors independently. The following theorem describes a more efficient, in fact optimal, algorithm.

Theorem 8.11. *Consider a set* P *of* n *points in the plane. For a point* $q \in P$ *let* $c_P(q)$ *denote the circular sequence of points from* $S \setminus \{q\}$ *ordered counterclockwise around* q *(in order as they would be encountered by a ray sweeping around* q*). The rotation system of* P*, consisting of all* $c_P(q)$*, for* $q \in P$*, collectively can be obtained in* $O(n^2)$ *time.*

Proof. Consider the projective dual P^* of P. An angular sweep around a point $q \in P$ in the primal plane corresponds to a traversal of the line q^* from left to right in the dual plane. (A collection of lines through a single point q corresponds to a collection of points on a single line q^* and slope corresponds to x-coordinate.) Clearly, the sequence of intersection points along all lines in P^* can be obtained by constructing the arrangement in $O(n^2)$ time. In the primal plane, any such sequence corresponds to an order of the remaining points according to the slope of the connecting line; to construct the circular sequence of points as they are encountered around q, we have to split the sequence obtained from the dual into those points that are to the left of q and those that are to the right of q; concatenating both yields the desired sequence. □

[1]As these graphs are not planar for $|P| \geqslant 5$, we do not have the natural dual notion of faces as in the case of planar graphs.

8.6 Segment Endpoint Visibility Graphs

A fundamental problem in motion planning is to find a short(est) path between two given positions in some domain, subject to certain constraints. As an example, suppose we are given two points $p, q \in \mathbb{R}^2$ and a set $S \subset \mathbb{R}^2$ of obstacles. What is the shortest path between p and q that avoids S?

Observation 8.12. *The shortest path (if it exists) between two points that does not cross a finite set of finite polygonal obstacles is a polygonal path whose interior vertices are obstacle vertices.*

One of the simplest type of obstacle conceivable is a line segment. In general the plane may be disconnected with respect to the obstacles, for instance, if they form a closed curve. However, if we restrict the obstacles to pairwise disjoint line segments then there is always a free path between any two given points. Apart from start and goal position, by the above observation we may restrict our attention concerning shortest paths to straight line edges connecting obstacle vertices, in this case, segment endpoints.

Definition 8.13. *Consider a set S of n disjoint line segments in $\mathbb{R}^2$. The* **segment endpoint visibility graph** *$\mathcal{V}(S)$ is a geometric straight line graph defined on the segment endpoints. Two segment endpoints p and q are connected by an edge in $\mathcal{V}(S)$ if and only if*

- *the line segment $\overline{pq}$ is in S or*
- *$\overline{pq} \cap s \subseteq \{p, q\}$ for every segment $s \in S$.*

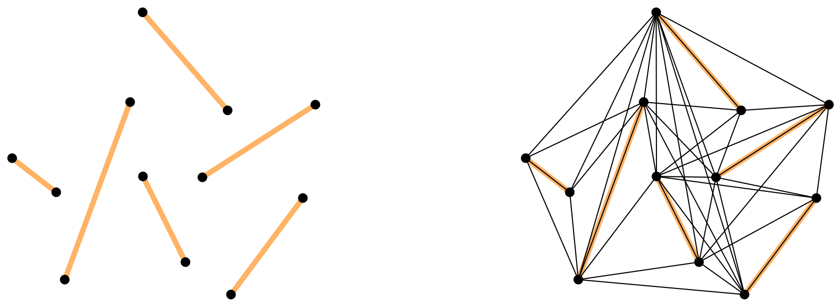


Figure 8.6: *A set of disjoint line segments and their endpoint visibility graph.*

If all segments are on the convex hull, the visibility graph is complete. If they form parallel chords of a convex polygon, the visibility graph consists of copies of K_4, glued together along opposite edges and the total number of edges is linear only.

These graphs also appear in the context of the following question: Given a set of disjoint line segments, is it possible to connect them to form (the boundary of) a simple polygon? It is easy to see that this is not possible in general: Just take three parallel chords of a convex polygon (Figure 8.7a). However, if we do not insist that the segments appear on the boundary, but allow them to be diagonals or epigonals, then it is always possible [11, 12]. In other words, the segment endpoint visibility graph of disjoint line segments is Hamiltonian, unless all segments are collinear. It is actually essential to allow epigonals and not only diagonals [9, 20] (Figure 8.7b).

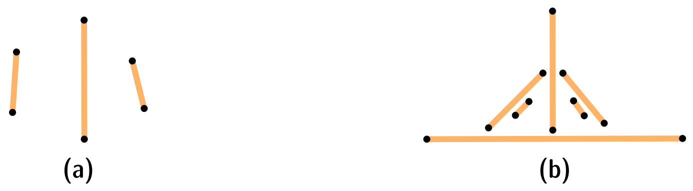


Figure 8.7: *Sets of disjoint line segments that do not allow certain polygons.*

Constructing $\mathcal{V}(S)$ for a given set S of disjoint segments in a brute force way takes $O(n^3)$ time. (Take all pairs of endpoints and check all other segments for obstruction.)

Theorem 8.14 (Welzl [21]). *The segment endpoint visibility graph of n disjoint line segments can be constructed in worst case optimal $O(n^2)$ time.*

Proof. As before we assume general position, that is, no three endpoints are collinear and no two have the same x-coordinate. It is no problem to handle such degeneracies explicitly.

We have seen above how all sorted angular sequences can be obtained from the dual line arrangement in $O(n^2)$ time. Topologically sweep the arrangement from left to right (corresponds to changing the slope of the primal rays from $-\infty$ to $+\infty$) while maintaining for each segment endpoint p the segment $s(p)$ it currently "sees" (if any). Initialize by brute force in $O(n^2)$ time (direction vertically downwards). Each intersection of two lines corresponds to two segment endpoints "seeing" each other along the primal line whose dual is the point of intersection. In order to process an intersection, we only need that all preceding (located to the left) intersections of the two lines involved have already been processed. This order corresponds to a topological sort of the arrangement graph where all edges are directed from left to right. (Clearly, this graph is acyclic, that is, it does not contain a directed cycle.) A topological sort can be obtained, for instance, via (reversed) post order DFS in time linear in the size of the graph (number of vertices and edges), which in our case here is $O(n^2)$.

When processing an intersection, there are four cases. Let p and q be the two points involved such that p is to the left of q.

1. The two points belong to the same input segment $\rightarrow$ output the edge pq, no change otherwise.

2. q is obscured from p by $s(p)$ $\rightarrow$ no change.

3. q is endpoint of $s(p)$ $\rightarrow$ output pq and update $s(p)$ to $s(q)$.

4. Otherwise q is endpoint of a segment t that now obscures $s(p) \to$ output pq and update $s(p)$ to t.

Thus any intersection can be processed in constant time and the overall runtime of this algorithm is quadratic. □

8.7 3-Sum

The 3-Sum problem is the following: Given a set S of n integers, does there exist a three-tuple[2] of elements from S that sum up to zero? By testing all three-tuples this can obviously be solved in $O(n^3)$ time. If the tuples to be tested are picked a bit more cleverly, we obtain an $O(n^2)$ algorithm.

Let $(s_1, \ldots, s_n)$ be the sequence of elements from S in increasing order. This sequence can be obtained by sorting in $O(n \log n)$ time. Then we test the tuples as follows.

```
For $i = 1, \ldots, n$ {
    $j = i$, $k = n$.
    While $k \geqslant j$ {
        If $s_i + s_j + s_k = 0$ then exit with triple $s_i, s_j, s_k$.
        If $s_i + s_j + s_k > 0$ then $k = k - 1$ else $j = j + 1$.
    }
}
```

The runtime is clearly quadratic. Regarding the correctness observe that the following is an invariant that holds at the start of every iteration of the inner loop: $s_i + s_x + s_k < 0$, for all $x \in \{i, \ldots, j-1\}$, and $s_i + s_j + s_x > 0$, for all $x \in \{k+1, \ldots, n\}$.

Interestingly, until very recently this was the best algorithm known for 3-Sum. But at FOCS 2014, Grønlund and Pettie [8] presented a deterministic algorithm that solves 3-Sum in $O(n^2(\log\log n/\log n)^{2/3})$ time.

They also give a bound of $O(n^{3/2}\sqrt{\log n})$ on the decision tree complexity of 3-Sum, which since then has been further improved in a series of papers. The latest improvement is due to Kane, Lovett, and Moran [13] who showed that $O(n \log^2 n)$ linear queries suffice (where a query amounts to ask for the sign of the sum of at most six input numbers with coefficients in $\{-1, 1\}$). In this decision tree model, only queries that involve the input numbers are counted, all other computation, for instance, using these query results to analyze the parameter space are for free. In other words, the results on the decision tree complexity of 3-Sum demonstrate that the (supposed) hardness of 3-Sum does not originate from the complexity of the decision tree.

[2]That is, an element of S may be chosen twice or even three times, although the latter makes sense for the number 0 only. : -)

The big open question remains whether an $O(n^{2-\varepsilon})$ algorithm can be achieved. Only in some very restricted models of computation—such as the 3-linear decision tree model[3]—it is known that 3-Sum requires quadratic time [6].

3-Sum hardness There is a whole class of problems that are equivalent to 3-Sum up to sub-quadratic time reductions [7]; such problems are referred to as **3-Sum-hard**.

Definition 8.15. *A problem* P *is* **3-Sum-hard** *if and only if every instance of 3-Sum of size* n *can be solved using a constant number of instances of* P*—each of* $O(n)$ *size—and* $o(n^{2-\varepsilon})$ *additional time, for some* $\varepsilon > 0$.

For instance, it is not hard to show that the following variation of 3-Sum—let us denote it by 3-Sum°—is 3-Sum-hard: Given a set S of n integers, does there exist a three-element subset of S whose elements sum up to zero?

Exercise 8.16. *Show that 3-Sum° is 3-Sum-hard.*

As another example, consider the Problem **GeomBase**: Given n points on the three horizontal lines $y = 0$, $y = 1$, and $y = 2$, is there a non-horizontal line that contains at least three of them?

3-Sum can be reduced to GeomBase as follows. For an instance $S = \{s_1, \ldots, s_n\}$ of 3-Sum, create an instance P of GeomBase in which for each s_i there are three points in P: $(s_i, 0)$, $(-s_i/2, 1)$, and $(s_i, 2)$. If there are any three collinear points in P, there must be one from each of the lines $y = 0$, $y = 1$, and $y = 2$. So suppose that $p = (s_i, 0)$, $q = (-s_j/2, 1)$, and $r = (s_k, 2)$ are collinear. The inverse slope of the line through p and q is $\frac{-s_j/2-s_i}{1-0} = -s_j/2 - s_i$ and the inverse slope of the line through q and r is $\frac{s_k+s_j/2}{2-1} = s_k + s_j/2$. The three points are collinear if and only if the two slopes are equal, that is, $-s_j/2 - s_i = s_k + s_j/2 \iff s_i + s_j + s_k = 0$.

A very similar problem is **General Position**, in which one is given n arbitrary points and has to decide whether any three are collinear. For an instance S of 3-Sum°, create an instance P of General Position by projecting the numbers s_i onto the curve $y = x^3$, that is, $P = \{(a, a^3) \mid a \in S\}$.

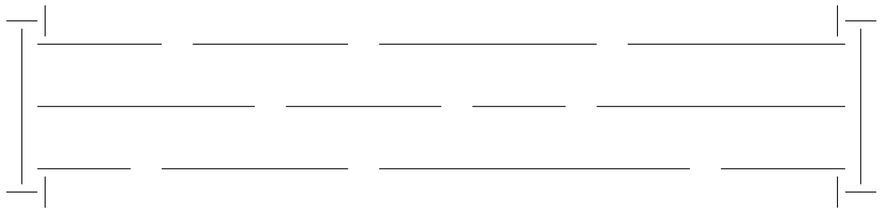
Suppose three of the points, say, (a, a^3), (b, b^3), and (c, c^3) are collinear. This is the case if and only if the slopes of the lines through each pair of them are equal. (Observe that a, b, and c are pairwise distinct.)

$$\begin{aligned}
(b^3 - a^3)/(b - a) &= (c^3 - b^3)/(c - b) \iff \\
b^2 + a^2 + ab &= c^2 + b^2 + bc \iff \\
b &= (c^2 - a^2)/(a - c) \iff \\
b &= -(a + c) \iff \\
a + b + c &= 0\,.
\end{aligned}$$

[3]where a decision depends on the sign of a linear expression in 3 input variables

Minimum Area Triangle is a strict generalization of General Position and, therefore, also 3-Sum-hard.

In **Segment Splitting/Separation**, we are given a set of n line segments and have to decide whether there exists a line that does not intersect any of the segments but splits them into two non-empty subsets. To show that this problem is 3-Sum-hard, we can use essentially the same reduction as for GeomBase, where we interpret the points along the three lines $y = 0$, $y = 1$, and $y = 2$ as sufficiently small "holes". The parts of the lines that remain after punching these holes form the input segments for the Splitting problem. Horizontal splits can be prevented by putting constant size gadgets somewhere beyond the last holes, see the figure below. The set of input segments for the segment



splitting problem requires sorting the points along each of the three horizontal lines, which can be done in $O(n \log n) = o(n^2)$ time. It remains to specify what "sufficiently small" means for the size of those holes. As all input numbers are integers, it is not hard to show that punching a hole of $(x - 1/4, x + 1/4)$ around each input point x is small enough.

In **Segment Visibility**, we are given a set S of n horizontal line segments and two segments $s_1, s_2 \in S$. The question is: Are there two points, $p_1 \in s_1$ and $p_2 \in s_2$ which can see each other, that is, the open line segment $\overline{p_1 p_2}$ does not intersect any segment from S? The reduction from 3-Sum is the same as for Segment Splitting, just put s_1 above and s_2 below the segments along the three lines.

In **Motion Planning**, we are given a robot (line segment), some environment (modeled as a set of disjoint line segments), and a source and a target position. The question is: Can the robot move (by translation and rotation) from the source to the target position, without ever intersecting the "walls" of the environment?

To show that Motion Planning is 3-Sum-hard, employ the reduction for Segment Splitting from above. The three "punched" lines form the doorway between two rooms, each modeled by a constant number of segments that cannot be split, similar to the boundary gadgets above. The source position is in one room, the target position in the other, and to get from source to target the robot has to pass through a sequence of three collinear holes in the door (suppose the doorway is sufficiently small compared to the length of the robot).

Exercise 8.17. *The 3-Sum' problem is defined as follows: given three sets S_1, S_2, S_3 of n integers each, are there $a_1 \in S_1$, $a_2 \in S_2$, $a_3 \in S_3$ such that $a_1 + a_2 + a_3 = 0$? Prove that the 3-Sum' problem and the 3-Sum problem as defined in the lecture ($S_1 = S_2 = S_3$) are equivalent, more precisely, that they are reducible to each other in subquadratic time.*

8.8 Ham Sandwich Theorem

Suppose two thieves have stolen a necklace that contains rubies and diamonds. Now it is time to distribute the prey. Both, of course, should get the same number of rubies and the same number of diamonds. On the other hand, it would be a pity to completely disintegrate the beautiful necklace. Hence they want to use as few cuts as possible to achieve a fair gem distribution.

To phrase the problem in a geometric (and somewhat more general) setting: Given two finite sets R and D of points, construct a line that bisects both sets, that is, in either halfplane defined by the line there are about half of the points from R and about half of the points from D. To solve this problem, we will make use of the concept of levels in arrangements.

Definition 8.18. *Consider an arrangement $\mathcal{A}(L)$ induced by a set L of n non-vertical lines in the plane. We say that a point p is on the* k-**level** *in $\mathcal{A}(L)$ if there are at most $k-1$ lines below and at most $n-k$ lines above p. The 1-level and the n-level are also referred to as* **lower** *and* **upper envelope***, respectively.*

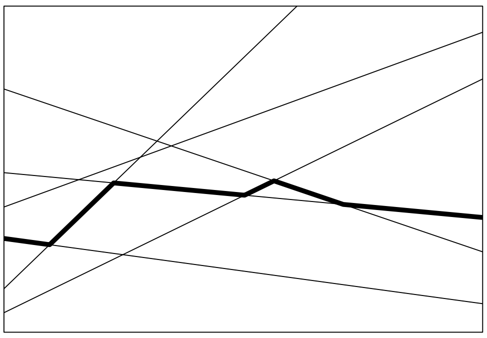


Figure 8.8: *The 3-level of an arrangement.*

Another way to look at the k-level is to consider the lines to be real functions; then the lower envelope is the pointwise minimum of those functions, and the k-level is defined by taking pointwise the k^th-smallest function value.

Theorem 8.19. *Let $R, D \subset \mathbb{R}^2$ be finite sets of points. Then there exists a line that bisects both R and D. That is, in either open halfplane defined by ℓ there are no more than $|R|/2$ points from R and no more than $|D|/2$ points from D.*

Proof. Without loss of generality suppose that both $|R|$ and $|D|$ are odd. (If, say, $|R|$ is even, simply remove an arbitrary point from R. Any bisector for the resulting set is also a bisector for R.) We may also suppose that no two points from $R \cup D$ have the same x-coordinate. (Otherwise, rotate the plane infinitesimally.)

Let R^* and D^* denote the set of lines dual to the points from R and D, respectively. Consider the arrangement $\mathcal{A}(R^*)$. The median level of $\mathcal{A}(R^*)$ defines the bisecting lines for R. As $|R| = |R^*|$ is odd, both the leftmost and the rightmost segment of this level are defined by the same line ℓ_r from R^*, the one with median slope. Similarly there is a corresponding line ℓ_d in $\mathcal{A}(D^*)$.

Since no two points from $R \cup D$ have the same x-coordinate, no two lines from $R^* \cup D^*$ have the same slope, and thus ℓ_r and ℓ_d intersect. Consequently, being piecewise linear continuous functions, the median level of $\mathcal{A}(R^*)$ and the median level of $\mathcal{A}(D^*)$ intersect (see Figure 8.9 for an example). Any point that lies on both median levels corresponds to a primal line that bisects both point sets simultaneously. □

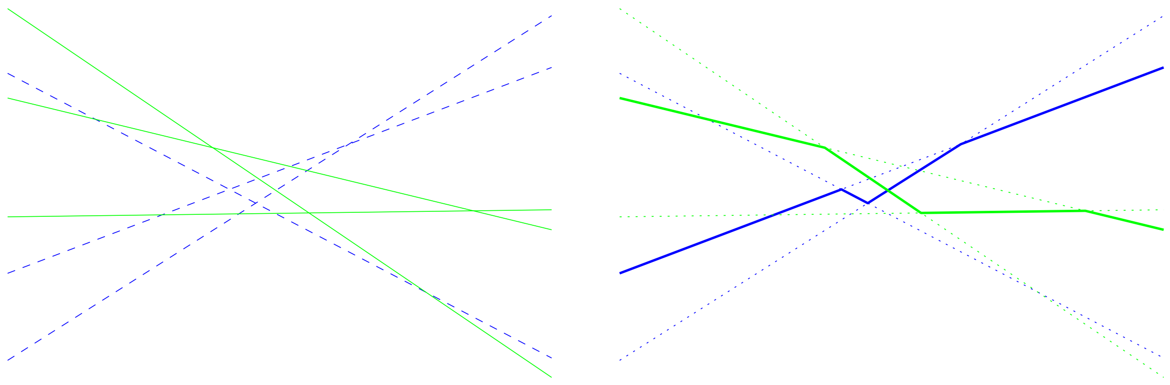


Figure 8.9: *An arrangement of 3 green lines (solid) and 3 blue lines (dashed) and their median levels (marked bold on the right hand side).*

How can the thieves use Theorem 8.19? If they are smart, they drape the necklace along some convex curve, say, a circle. Then by Theorem 8.19 there exists a line that simultaneously bisects the set of diamonds and the set of rubies. As any line intersects the circle at most twice, the necklace is cut at most twice.

However, knowing about the existence of such a line certainly is not good enough. It is easy to turn the proof given above into an $O(n^2)$ algorithm to construct a line that simultaneously bisects both sets. But we can do better...

8.9 Constructing Ham Sandwich Cuts in the Plane

The algorithm outlined below is not only interesting in itself but also because it illustrates one of the fundamental general paradigms for designing optimization algorithms: *prune & search*. The basic idea behind prune & search is to search the space of possible solutions by at each step excluding some part of this space from further consideration. For instance, if at each step a constant fraction of all possible solutions can be discarded and a single step is linear in the number of solutions to be considered, then for the runtime we obtain a recursion of the form

$$T(n) \leqslant cn + T\left(n\left(1-\frac{1}{d}\right)\right) < cn\sum_{i=0}^{\infty}\left(\frac{d-1}{d}\right)^i = cn\frac{1}{1-\frac{d-1}{d}} = cdn\,,$$

that is, a linear time algorithm overall. Another well-known example of prune & search is binary search: every step takes constant time and about half of the possible solutions can be discarded, resulting in a logarithmic runtime overall.

Theorem 8.20 (Edelsbrunner and Waupotitsch [5]). *Let* $R, D \subset \mathbb{R}^2$ *be finite sets of points with* $n = |R| + |D|$*. Then in* $O(n \log n)$ *time one can find a line* ℓ *that simultaneously bisects* R *and* D*. That is, in either open halfplane defined by* ℓ *there are no more than* $|R|/2$ *points from* R *and no more than* $|D|/2$ *points from* D*.*

Proof. We describe a recursive algorithm find$(L_1, k_1, L_2, k_2, (x_1, x_2))$, for sets L_1, L_2 of lines in $\mathbb{R}^2$, non-negative integers k_1 and k_2, and a real interval (x_1, x_2), to find an intersection between the k_1-level of $\mathcal{A}(L_1)$ and the k_2-level of $\mathcal{A}(L_2)$, under the following assumption that is called *odd-intersection property*: the k_1-level of $\mathcal{A}(L_1)$ and the k_2-level of $\mathcal{A}(L_2)$ intersect an odd number of times in (x_1, x_2) and they do not intersect at $x \in \{x_1, x_2\}$. Note that the odd-intersection property is equivalent to saying that the level that is above the other at $x = x_1$ is below the other at $x = x_2$. In the end, we are interested in find$(R^*, (|R| + 1)/2, D^*, (|D| + 1)/2, (-\infty, \infty))$. As argued in the proof of Theorem 8.19, for these arguments the odd-intersection property holds.

First let $L = L_1 \cup L_2$ and find a line μ with median slope in L. Denote by $L_<$ and $L_>$ the lines from L with slope less than and greater than μ, respectively. Using an infinitesimal rotation of the plane if necessary, we may assume without loss of generality that no two points in $R \cup D$ have the same x-coordinate and thus no two lines in L have the same slope. Pair the lines in $L_<$ with those in $L_>$ arbitrarily to obtain an almost perfect matching in the complete bipartite graph on $L_< \cup L_>$. Denote by I the $\lfloor (|L_<| + |L_>|)/2 \rfloor$ points of intersection generated by the pairs chosen, and let j be a point from I with median x-coordinate.

Determine the intersection (j, y_1) of the k_1-level of L_1 with the vertical line $x = j$ and the intersection (j, y_2) of the k_2-level of L_2 with the vertical line $x = j$. If both levels intersect at $x = j$, return the intersection and exit. Otherwise, if $j \in (x_1, x_2)$, then exactly one of the intervals (x_1, j) or (j, x_2) has the odd-intersection property, say[4], (x_1, j). In other words, we can from now on restrict our focus to the halfplane $x \leqslant j$. The case $j \notin (x_1, x_2)$ is no different, except that we simply keep the original interval (x_1, x_2).

In the following it is our goal to discard a constant fraction of the lines in L from further consideration. To this end, let $I_>$ denote the set of points from I with x-coordinate greater than j, and let μ' be a line parallel to μ such that about half of the points from $I_>$ are above μ' (and thus the other about half of points from $I_>$ are below μ'). We consider the four quadrants formed by the two lines $x = j$ and μ'. By assumption the odd-intersection property (for the k_1-level of L_1 and the k_2-level of L_2) holds for the (union of the) left two quadrants. Therefore the odd-intersection property holds for exactly one of the left two quadrants; we call this the *interesting* quadrant. Suppose furthermore that the upper left quadrant Q_2 is interesting. We will later argue how to algorithmically determine the interesting quadrant (see Figure 8.10 for an example).

[4]The other case is completely symmetric and thus will not be discussed here.

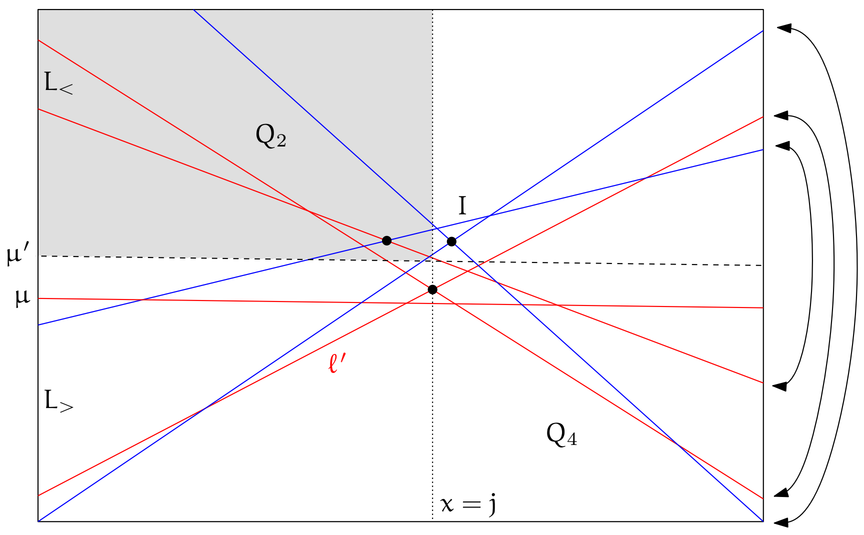


Figure 8.10: *An example with a set* L_1 *of* 4 *red lines and a set* L_2 *of* 3 *blue lines. Suppose that* $k_1 = 3$ *and* $k_2 = 2$*. Then the interesting quadrant is the top-left one (shaded) and the red line* ℓ' *(the line with a smallest slope in* L_1*) would be discarded because it does not intersect the interesting quadrant.*

Then by definition of j and μ' about a quarter of the points from I are contained in the opposite, that is, the lower right quadrant Q_4. Any point in Q_4 is the point of intersection of two lines from L, exactly one of which has slope larger than μ'. As no line with slope larger than μ' that passes through Q_4 can intersect Q_2, any such line can be discarded from further consideration. In this case, the lines discarded pass completely below the interesting quadrant Q_2. For every line discarded in this way from L_1 or L_2, the parameter k_1 or k_2, respectively, has to be decreased by one. In the symmetric case where the lines discarded pass above the interesting quadrant, the parameters k_1 and k_2 stay the same. In any case, about a $1/8$-fraction of all lines in L is discarded. Denote the resulting sets of lines (after discarding) by L'_1 and L'_2, and let k'_1 and k'_2 denote the correspondingly adjusted levels.

We want to apply the algorithm recursively to compute an intersection between the k'_1-level of L'_1 and the k'_2-level of L'_2. However, discarding lines changes the arrangement and its levels. As a result, it is not clear that the odd-intersection property holds for the k'_1-level of L'_1 and the k'_2-level of L'_2 on the interval (x_1, j), or even on the original interval (x_1, x_2). Note that we do know that these levels intersect in the interesting quadrant, and this intersection persists because none of the involved lines is removed. However, it is conceivable that the removal of lines changes the parity of intersections in the non-interesting quadrant of the interval under consideration. Luckily, this issue can easily be resolved as a part of the algorithm to determine the interesting quadrant, which we will discuss next. More specifically, we will show how to determine a subinterval $(x'_1, x'_2) \subseteq (x_1, x_2)$ on which the odd-intersection property holds for the k'_1-level of L'_1

and the k_2'-level of L_2'.

So let us argue how to determine the interesting quadrant, that is, how to test whether the k_1-level of L_1 and the k_2-level of L_2 intersect an odd number of times in $S_{(x_1,j)} \cap H_\mu^+$, where $S_{(x_1,j)}$ is the vertical strip $(x_1, j) \times \mathbb{R}$ and H_μ^+ is the open halfplane above μ'. For this it is enough to trace μ' through the arrangement $\mathcal{A}(L)$ while keeping track of the position of the two levels of interest. Initially, at $x = x_1$ we know which level is above the other. At every intersection of one of the two levels with μ', we can check whether the ordering is still consistent with that initial ordering. For instance, if both were above μ' initially and the level that was above the other intersects μ' first, we can deduce that there must be an intersection of the two levels above μ'. As the relative position of the two levels is reversed at $x = x_2$, at some point an inconsistency, that is, the presence of an intersection will be detected and we will be able to tell whether it is above or below μ'. (There could be many more intersections between the two levels, but finding just one intersection is good enough.) Along with this above/below information we also obtain a suitable interval (x_1', x_2') for which the odd-intersection property holds because the levels of interest do not change in that interval.

The trace of μ' in $\mathcal{A}(L)$ can be computed by a sweep along μ', which amounts to computing all intersections of μ' with the lines from L and sorting them by x-coordinate. During the sweep we keep track of the number of lines from L_1 below μ' and the number of lines from L_2 below μ'. At every point of intersection, these counters can be adjusted and any intersection with one of the two levels of interest is detected. Therefore computing the trace takes $O(|L| \log |L|)$ time. This step dominates the whole algorithm, noting that all other operations are based on rank-i element selection, which can be done in linear time [4]. Altogether, we obtain as a recursion for the runtime

$$T(n) \leqslant cn \log n + T(7n/8) = O(n \log n). \qquad \square$$

You can also think of the two point sets as a discrete distribution of a ham sandwich that is to be cut fairly, that is, in such a way that both parts have the same amount of ham and the same amount of bread. That is where the name "ham sandwich cut" comes from. The theorem generalizes both to higher dimension and to more general types of measures (here we study the discrete setting only where we simply count points). These generalizations can be proven using the *Borsuk-Ulam Theorem*, which states that any continuous map from S^d to $\mathbb{R}^d$ must map some pair of antipodal points to the same point. For a proof of both theorems and many applications see Matoušek's book [17].

Theorem 8.21. *Let $P_1, \ldots, P_d \subset \mathbb{R}^d$ be finite sets of points. Then there exists a hyperplane H that simultaneously bisects all of $P_1, \ldots, P_d$. That is, in either open halfspace defined by H there are no more than $|P_i|/2$ points from P_i, for every $i \in \{1, \ldots, d\}$.*

This implies that the thieves can fairly distribute a necklace consisting of d types of gems using at most d cuts.

In the plane, a ham sandwich cut can be found in linear time using a sophisticated prune and search algorithm by Lo, Matoušek and Steiger [16]. But in higher dimension, the algorithmic problem gets harder. In fact, already for $\mathbb{R}^3$ the complexity of finding a ham sandwich cut is wide open: The best algorithm known, from the same paper by Lo et al. [16], has runtime $O(n^{3/2}\log^2 n/\log^* n)$ and no non-trivial lower bound is known. If the dimension d is not fixed, it is both NP-hard and W[1]-hard[5] in d to decide the following question [15]: Given $d \in \mathbb{N}$, finite point sets $P_1, \ldots, P_d \subset \mathbb{R}^d$, and a point $p \in \bigcup_{i=1}^d P_i$, is there a ham sandwich cut through p?

Exercise 8.22. *The goal of this exercise is to develop a data structure for halfspace range counting.*

a) *Given a set $P \subset \mathbb{R}^2$ of n points in general position, show that it is possible to partition this set by two lines such that each region contains at most $\lceil \frac{n}{4} \rceil$ points.*

b) *Design a data structure of size $O(n)$ that can be constructed in time $O(n \log n)$ and allows you, for any halfspace h, to output the number of points $|P \cap h|$ of P contained in this halfspace h in time $O(n^\alpha)$, for some $0 < \alpha < 1$.*

Exercise 8.23. *Prove or disprove the following statement: Given three finite sets A, B, C of points in the plane, there is always a circle or a line that bisects A, B and C simultaneously (that is, no more than half of the points of each set are inside or outside the circle or on either side of the line, respectively).*

8.10 Davenport-Schinzel Sequences

The complexity of a simple arrangement of n lines in $\mathbb{R}^2$ is $\Theta(n^2)$ and so every algorithm that uses such an arrangement explicitly needs $\Omega(n^2)$ time. However, there are many scenarios in which we do not need the whole arrangement but only some part of it. For instance, to construct a ham sandwich cut for two sets of points in $\mathbb{R}^2$ one needs the median levels of the two corresponding line arrangements only. As mentioned in the previous section, the relevant information about these levels can actually be obtained in linear time. Similarly, in a motion planning problem where the lines are considered as obstacles we are only interested in the cell of the arrangement we are located in. There is no way to ever reach any other cell, anyway.

This chapter is concerned with analyzing the complexity—that is, the number of vertices and edges—of a single cell in an arrangement of n curves in $\mathbb{R}^2$. In case of a line arrangement this is mildly interesting only: Every cell is convex and any line can appear at most once along the cell boundary. On the other hand, it is easy to construct an example in which there is a cell C such that every line appears on the boundary ∂C.

But when we consider arrangement of line segments rather than lines, the situation changes in a surprising way. Certainly a single segment can appear several times along the boundary of a cell, see the example in Figure 8.11. Make a guess: What is the maximal complexity of a cell in an arrangement of n line segments in $\mathbb{R}^2$?

[5]Essentially this means that it is unlikely to be solvable in time $O(f(d)p(n))$, for an arbitrary function f and a polynomial p.

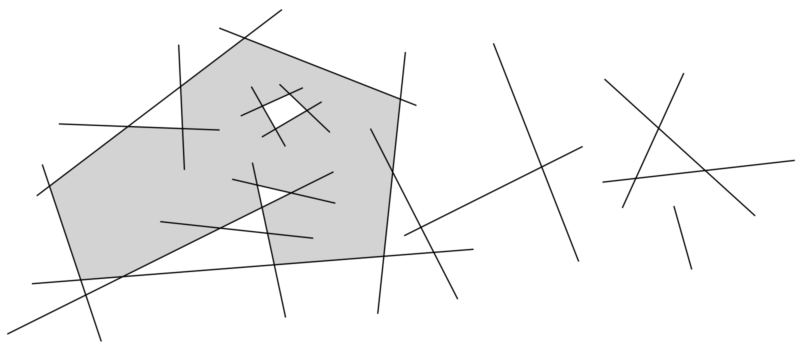


Figure 8.11: *A single cell in an arrangement of line segments.*

You will find out the correct answer soon, although we will not prove it here. But my guess would be that it is rather unlikely that your guess is correct, unless, of course, you knew the answer already. : -)

For a start we will focus on one particular cell of any arrangement that is very easy to describe: the lower envelope or, intuitively, everything that can be seen vertically from below. To analyze the complexity of lower envelopes we use a combinatorial description using strings with forbidden subsequences, so-called Davenport-Schinzel sequences. These sequences are of independent interest, as they appear in a number of combinatorial problems [2] and in the analysis of data structures [19]. The techniques used apply not only to lower envelopes but also to arbitrary cells of arrangements.

Definition 8.24. *An* (n, s)-**Davenport-Schinzel sequence**, *for* $n, s \in \mathbb{N}$, *is a sequence over an alphabet* A *of size* n *in which*

- *no two consecutive characters are the same and*
- *there is no alternating subsequence of the form* $\ldots a \ldots b \ldots a \ldots b \ldots$ *of* $s + 2$ *characters, for any* $a, b \in A$.

Let $\lambda_s(n)$ *be the length of a longest* (n, s)*-Davenport-Schinzel sequence.*

For example, $abcbacb$ is a $(3, 4)$-DS sequence but not a $(3, 3)$-DS sequence because it contains the subsequence $bcbcb$.

Proposition 8.25. $\lambda_s(m) + \lambda_s(n) \leqslant \lambda_s(m + n)$.

Proof. On the left hand side, we consider two Davenport-Schinzel sequences, one over an alphabet A of size m and another over an alphabet B of size n. We may suppose that $A \cap B = \emptyset$ (for each character $x \in A \cap B$ introduce a new character x' and replace all occurrences of x in the second sequence by x'). Concatenating both sequences yields a Davenport-Schinzel sequence over the alphabet $A \cup B$ of size $m + n$. □

Let us now see how Davenport-Schinzel sequences are connected to lower envelopes. Consider a collection $\mathcal{F} = \{f_1, \ldots, f_n\}$ of real-valued continuous functions that are defined on a common interval $I \subset \mathbb{R}$. The *lower envelope* $\mathcal{L}_{\mathcal{F}}$ of $\mathcal{F}$ is defined as the pointwise minimum of the functions f_i, $1 \leqslant i \leqslant n$, over I. Suppose that any pair f_i, f_j, $1 \leqslant i < j \leqslant n$, intersects in at most s points. Then I can be decomposed into a finite sequence $I_1, \ldots, I_\ell$ of (maximal connected) pieces on each of which a single function from $\mathcal{F}$ defines $\mathcal{L}_{\mathcal{F}}$. Define the sequence $\phi(\mathcal{F}) = (\phi_1, \ldots, \phi_\ell)$, where f_{ϕ_i} is the function from $\mathcal{F}$ which defines $\mathcal{L}_{\mathcal{F}}$ on I_i.

Observation 8.26. *$\phi(\mathcal{F})$ is an (n, s)-Davenport-Schinzel sequence.*

In the case of line segments the above statement does not hold because a set of line segments is in general not defined on a common real interval.

Proposition 8.27. *Let $\mathcal{F}$ be a collection of n real-valued continuous functions, each of which is defined on some real interval. If any two functions from $\mathcal{F}$ intersect in at most s points then $\phi(\mathcal{F})$ is an $(n, s+2)$-Davenport-Schinzel sequence.*

Proof. Let us first argue that we may suppose that the functions in $\mathcal{F}$ are piecewise quadratic. Denote by P the set of points that are vertices of the arrangement induced by the graphs of the functions in $\mathcal{F}$. In other words, P is the set of all endpoints of functions and intersection points of two functions from $\mathcal{F}$. Let $x_1, \ldots, x_m$ denote the sequence of x-coordinates of points from P, sorted in increasing order.

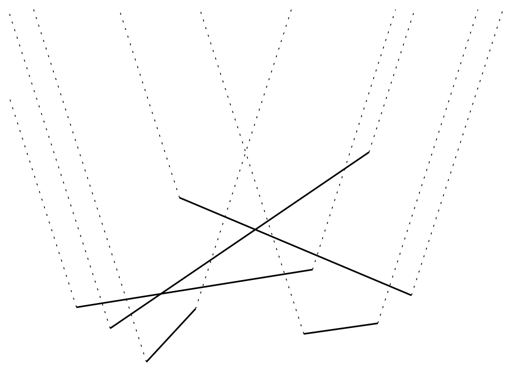
Consider the set $\mathcal{F}_i \subseteq \mathcal{F}$ of functions that are defined on the interval $[x_i, x_{i+1}]$, for $i \in \{1, \ldots, m-1\}$. By definition of P, no two functions intersect within the interval (x_i, x_{i+1}), that is, throughout (x_i, x_{i+1}) the functions from $\mathcal{F}_i$ maintain the same total (y-)order. We describe how to replace each function in $\mathcal{F}_i$ on $[x_i, x_{i+1}]$ by a quadratic function such that this order and therefore the combinatorics of the induced arrangement remains unchanged.

Consider a function $f \in \mathcal{F}_i$. If f is the only function in $\mathcal{F}_i$ that passes through both points $(x_i, f(x_i))$ and $(x_{i+1}, f(x_{i+1}))$, then replace f by the line segment connecting these two points. Otherwise, consider the set F of all functions in $\mathcal{F}_i$ that pass through both points, and replace each function by a parabolic arc connecting the two points. These parabolic arcs can be chosen so that they are pairwise disjoint within the open interval (x_i, x_{i+1}) and all of them are infinitesimally close to the line segment between $(x_i, f(x_i))$ and $(x_{i+1}, f(x_{i+1}))$, while maintaining the total order among the functions in F within the interval (x_i, x_{i+1}). The described replacement does not introduce any new intersection (that is why we used different parabolic arcs) and maintains the order of functions at any x-coordinate. In particular, $\phi(\mathcal{F})$ remains the same.

Now we are ready to prove the actual statement. The idea is to extend the functions to be defined on all of $\mathbb{R}$, without changing $\phi(\mathcal{F})$ too much, and then resort to Observation 8.26.

Consider any function $f \in \mathcal{F}$ defined on an interval $[a, b]$. Extend f to $\mathbb{R}$ using almost vertical rays pointing upward; from a use a ray of sufficiently small slope, from b use a ray of sufficiently large slope. For all functions use the same slope on these two extensions such that no extensions in the same direction intersect. By *sufficiently small/large* we mean that for any extension ray there is no endpoint of a function nor an intersection point of two functions in the open angular wedge bounded by the extension ray and the vertical ray starting from the same source. (There may be such points *on* the vertical ray, but not in the open wedge between the two rays.) As any two functions intersect only a finite number of times, there is only a finite number of endpoints and intersection points to consider, and so "sufficiently small/large" is well-defined.

Denote the resulting collection of functions totally defined on $\mathbb{R}$ by $\mathcal{F}'$. If the extension rays are sufficiently close to vertical then $\phi(\mathcal{F}') = \phi(\mathcal{F})$. (Recall that by our reasoning from above we may regard each function as a parabolic arc or a line segment locally.)



For any $f \in \mathcal{F}'$ a single extension ray can create at most one additional intersection with any $g \in \mathcal{F}'$. (Let $[a_f, b_f]$ and $[a_g, b_g]$ be the intervals on which the function f and g, respectively, was defined originally. Consider the ray r extending f from a_f to the left. If $a_f \in [a_g, b_g]$ then r may create a new intersection with g, if $a_f > b_g$ then r creates a new intersection with the right extension of g from b_g, and if $a_f < a_g$ then r does not create any new intersection with g.)

On the other hand, for any pair s, t of segments, neither the left extension of the leftmost segment endpoint nor the right extension of the rightmost segment endpoint can introduce an additional intersection. Therefore, any pair of segments in $\mathcal{F}'$ intersects at most $s + 2$ times and the claim follows. □

Next we will give an upper bound on the length of Davenport-Schinzel sequences for small s.

Lemma 8.28. *$\lambda_1(n) = n$, $\lambda_2(n) = 2n - 1$, and $\lambda_3(n) \leqslant 2n(1 + \log n)$.*

Proof. $\lambda_1(n) = n$ is obvious. $\lambda_2(n) = 2n - 1$ is given as an exercise. We prove $\lambda_3(n) \leqslant 2n(1 + \log n) = O(n \log n)$.

For $n = 1$ it is $\lambda_3(1) = 1 \leqslant 2$. For $n > 1$ consider any $(n, 3)$-DS sequence σ of length $\lambda_3(n)$. Let a be a character that appears least frequently in σ. Clearly a appears at most $\lambda_3(n)/n$ times in σ. Delete all appearances of a from σ to obtain a sequence σ' on $n - 1$ symbols. But σ' is not necessarily a DS sequence because there may be consecutive appearances of a character b in σ', in case that $\sigma = \ldots bab \ldots$.

Claim: There are at most two pairs of consecutive appearances of the same character in σ'. Indeed, such a pair can be created around the first and last appearance of a in σ only. If any intermediate appearance of a creates a pair bb in σ' then $\sigma = \ldots a \ldots bab \ldots a \ldots$, in contradiction to σ being an $(n, 3)$-DS sequence.

Therefore, one can remove at most two characters from σ' to obtain a $(n-1, 3)$-DS sequence $\tilde{\sigma}$. As the length of $\tilde{\sigma}$ is bounded by $\lambda_3(n-1)$, we obtain $\lambda_3(n) \leqslant \lambda_3(n-1) + \lambda_3(n)/n + 2$. Reformulating yields

$$\underbrace{\frac{\lambda_3(n)}{n}}_{=:f(n)} \leqslant \underbrace{\frac{\lambda_3(n-1)}{n-1}}_{=f(n-1)} + \frac{2}{n-1} \leqslant \underbrace{1}_{=f(1)} + 2\sum_{i=1}^{n-1} \frac{1}{i} = 1 + 2H_{n-1}$$

and together with $2H_{n-1} < 1 + 2\log n$ we obtain $\lambda_3(n) \leqslant 2n(1 + \log n)$. □

Bounds for higher-order Davenport-Schinzel sequences. As we have seen, $\lambda_1(n)$ (no aba) and $\lambda_2(n)$ (no $abab$) are both linear in n. It turns out that for $s \geqslant 3$, $\lambda_s(n)$ is slightly superlinear in n (taking s fixed). The bounds are known almost exactly, and they involve the inverse Ackermann function $\alpha(n)$, a function that grows extremely slowly.

To define the inverse Ackermann function, we first define a hierarchy of functions $\alpha_1(n)$, $\alpha_2(n)$, $\alpha_3(n)$, ... where, for every fixed k, $\alpha_k(n)$ grows much more slowly than $\alpha_{k-1}(n)$:

We first let $\alpha_1(n) = \lceil n/2 \rceil$. Then, for each $k \geqslant 2$, we define $\alpha_k(n)$ to be the number of times we must apply α_{k-1}, starting from n, until we get a result not larger than 1. In other words, $\alpha_k(n)$ is defined recursively by:

$$\alpha_k(n) = \begin{cases} 0, & \text{if } n \leqslant 1; \\ 1 + \alpha_k(\alpha_{k-1}(n)), & \text{otherwise.} \end{cases}$$

Thus, $\alpha_2(n) = \lceil \log_2 n \rceil$, and $\alpha_3(n) = \log^* n$.

Now fix n, and consider the sequence $\alpha_1(n), \alpha_2(n), \alpha_3(n), \ldots$. For every fixed n, this sequence decreases rapidly until it settles at 3. We define $\alpha(n)$ (the inverse Ackermann function) as the function that, given n, returns the smallest k such that $\alpha_k(n)$ is at most 3:

$$\alpha(n) = \min\{k \mid \alpha_k(n) \leqslant 3\}.$$

We leave as an exercise to show that for every fixed k we have $\alpha_k(n) = o(\alpha_{k-1}(n))$ and $\alpha(n) = o(\alpha_k(n))$.

Coming back to the bounds for Davenport-Schinzel sequences, for $\lambda_3(n)$ (no $ababa$) it is known that $\lambda_3(n) = \Theta(n\alpha(n))$ [10]. In fact it is known that $\lambda_3(n) = 2n\alpha(n) \pm O(n\sqrt{\alpha(n)})$ [14, 18]. For $\lambda_4(n)$ (no $ababab$) we have $\lambda_4(n) = \Theta(n \cdot 2^{\alpha(n)})$ [3].

For higher-order sequences the known upper and lower bounds are almost tight, and they are of the form $\lambda_s(n) = n \cdot 2^{\text{poly}(\alpha(n))}$, where the degree of the polynomial in the exponent is roughly $s/2$ [3, 18].

Realizing DS sequences as lower envelopes. There exists a construction of a set of n segments in the plane whose lower-envelope sequence has length $\Omega(n\alpha(n))$. (In fact, the lower-envelope sequence has length $n\alpha(n) - O(n)$, with a leading coefficient of 1; it is an open problem to get a leading coefficient of 2, or prove that this is not possible.)

It is an open problem to construct a set of n parabolic arcs in the plane whose lower-envelope sequence has length $\Omega(n \cdot 2^{\alpha(n)})$.

Generalizations of DS sequences. Also generalizations of Davenport-Schinzel sequences have been studied, for instance, where arbitrary subsequences (not necessarily an alternating pattern) are forbidden. For a word σ and $n \in \mathbb{N}$ define $\mathrm{Ex}(\sigma, n)$ to be the maximum length of a word over $A = \{1, \ldots, n\}^*$ that does not contain a subsequence of the form σ. For example, $\mathrm{Ex}(ababa, n) = \lambda_3(n)$. If σ consists of two letters only, say a and b, then $\mathrm{Ex}(\sigma, n)$ is super-linear if and only if σ contains $ababa$ as a subsequence [1]. This highlights that the alternating forbidden pattern is of particular interest.

Exercise 8.29. *Prove that* $\lambda_2(n) = 2n - 1$.

Exercise 8.30. *Prove that* $\lambda_s(n)$ *is finite for all* s *and* n.

8.11 Constructing lower envelopes

Theorem 8.31. *Let* $\mathcal{F} = \{f_1, \ldots, f_n\}$ *be a collection of real-valued continuous functions defined on a common interval* $I \subset \mathbb{R}$ *such that no two functions from* $\mathcal{F}$ *intersect in more than* s *points. Then the lower envelope* $\mathcal{L}_{\mathcal{F}}$ *can be constructed in* $O(\lambda_s(n) \log n)$ *time. (Assuming that an intersection between any two functions can be constructed in constant time.)*

Proof. Divide and conquer. For simplicity, assume that n is a power of two. Split $\mathcal{F}$ into two equal parts $\mathcal{F}_1$ and $\mathcal{F}_2$ and construct $\mathcal{L}_{\mathcal{F}_1}$ and $\mathcal{L}_{\mathcal{F}_2}$ recursively. The resulting envelopes can be merged using line sweep by processing $2\lambda_s(n/2)+\lambda_s(n) \leqslant 2\lambda_s(n)$ events (the inequality $2\lambda_s(n/2) \leqslant \lambda_s(n)$ is by Proposition 8.25). Here the first term accounts for events generated by the vertices of the two envelopes to be merged. The second term accounts for their intersections, each of which generates a vertex of the resulting envelope. Observe that no sorting is required and the sweep line status structure is of constant size. Therefore, the sweep can be done in time linear in the number of events.

This yields the following recursion for the runtime $T(n)$ of the algorithm. $T(n) \leqslant 2T(n/2) + c\lambda_s(n)$, for some constant $c \in \mathbb{N}$. Using Proposition 8.25 it follows that $T(n) \leqslant c\sum_{i=1}^{\log n} 2^i \lambda_s(n/2^i) \leqslant c\sum_{i=1}^{\log n} \lambda_s(n) = O(\lambda_s(n) \log n)$. □

Exercise 8.32. *Show that every* (n, s)*-Davenport-Schinzel sequence can be realized as the lower envelope of* n *continuous functions from* $\mathbb{R}$ *to* $\mathbb{R}$*, every pair of which intersect at most* s *times.*

Exercise 8.33. *Show that every Davenport-Schinzel sequence of order two can be realized as a lower envelope of* n *parabolas.*

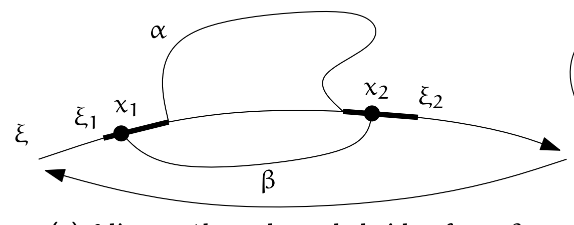
8.12 Complexity of a single face

Theorem 8.34. *Let $\Gamma = \{\gamma_1, \ldots, \gamma_n\}$ be a collection of Jordan arcs in $\mathbb{R}^2$ such that each pair intersects in at most s points, for some $s \in \mathbb{N}$. Then the combinatorial complexity of any single face in the arrangement $\mathcal{A}(\Gamma)$ is $O(\lambda_{s+2}(n))$.*

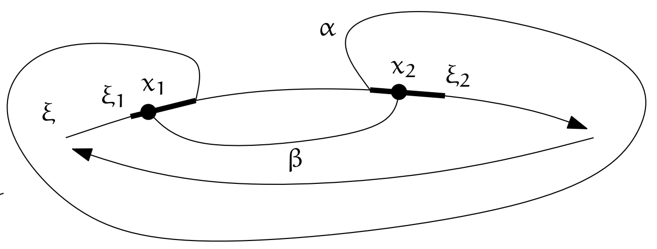
Proof. Consider a face f of $\mathcal{A}(\Gamma)$. In general, the boundary of f might consist of several connected components. But as any single curve can appear in at most one component, by Proposition 8.25 we may suppose that the boundary consists of one component only.

Replace each γ_i by two directed arcs γ_i^+ and γ_i^- that together form a closed curve that is infinitesimally close to γ_i. Denote by S the circular sequence of these oriented curves, in their order along the (oriented) boundary ∂f of f.

Consistency Lemma. Let ξ be one of the oriented arcs γ_i^+ or γ_i^-. The order of portions of ξ that appear in S is consistent with their order along ξ. (That is, for each ξ we can break up the circular sequence S into a linear sequence $S(\xi)$ such that the portions of ξ that correspond to appearances of ξ in $S(\xi)$ appear in the same order along ξ.)



(a) f lies on the unbounded side of $\alpha \cup \beta$.



(b) f lies on the bounded side of $\alpha \cup \beta$.

Figure 8.12: *Cases in the Consistency Lemma.*

Consider two portions ξ_1 and ξ_2 of ξ that appear consecutively in S (that is, there is no other occurrence of ξ in between). Choose points $x_1 \in \xi_1$ and $x_2 \in \xi_2$ and connect them in two ways: first by the arc α following ∂f as in S, and second by an arc β inside the closed curve formed by γ_i^+ or γ_i^-. The curves α and β do not intersect except at their endpoints and they are both contained in the complement of the interior of f. In other words, $\alpha \cup \beta$ forms a closed Jordan curve and f lies either in the interior of this curve or in its exterior. In either case, the part of ξ between ξ_1 and ξ_2 is separated from f by $\alpha \cup \beta$ and, therefore, no point from this part can appear anywhere along ∂f. In other words, ξ_1 and ξ_2 are also consecutive boundary parts in the order of boundary portions along ξ, which proves the lemma.

Break up S into a linear sequence $S' = (s_1, \ldots, s_t)$ arbitrarily. For each oriented arc ξ, consider the sequence $s(\xi)$ of its portions along ∂f in the order in which they appear along ξ. By the Consistency Lemma, $s(\xi)$ corresponds to a subsequence of S, starting at s_k, for some $1 \leqslant k \leqslant t$. In order to consider $s(\xi)$ as a subsequence of S', break up the symbol for ξ into two symbols ξ and ξ' and replace all occurrences of ξ in S' before s_k by ξ'. Doing so for all oriented arcs results in a sequence S^* on at most $4n$ symbols.

Claim: S^* is a $(4n, s+2)$-Davenport-Schinzel sequence.

Clearly no two adjacent symbols in S^* are the same. Suppose S^* contains an alternating subsequence $\sigma = \ldots\xi\ldots\eta\ldots\xi\ldots\eta\ldots$ of length $s+4$. For any occurrence of ξ in this sequence, choose a point from the corresponding part of ∂f. This gives a sequence $x_1, \ldots, x_{\lceil (s+4)/2 \rceil}$ of points on ∂f. These points we can connect in this order by a Jordan arc $C(\xi)$ that stays within the closed curve formed by ξ and its counterpart—except for the points $x_1, \ldots, x_{\lceil (s+4)/2 \rceil}$, which lie on this closed curve. Similarly we may choose points $y_1, \ldots, y_{\lfloor (s+4)/2 \rfloor}$ on ∂f that correspond to the occurrences of η in σ and connect these points in this order by a Jordan arc $C(\eta)$ that stays (except for the points $y_1, \ldots, y_{\lfloor (s+4)/2 \rfloor}$) within the closed curve formed by η and its counterpart.

Now consider any four consecutive elements in σ and the corresponding points x_i, y_i, x_{i+1}, y_{i+1}, which appear in this order—and, thus, can be regarded as connected by an arc—along ∂f. In addition, the points x_i and x_{i+1} are connected by an arc of $C(\xi)$, and similarly y_i and y_{i+1} are connected by an arc of $C(\xi)$. Both arcs, except for their endpoints, lie in the exterior of f. Finally, we can place a point u into the interior of f and connect u by pairwise interior-disjoint arcs to each of x_i, y_i, x_{i+1}, and y_{i+1}, such that the relative interior of these four arcs stays in the interior of f. By construction, no two of these arcs cross (intersect at a point that is not a common endpoint), except possibly for the arcs x_i, x_{i+1} and y_i, y_{i+1} in the exterior of f. In fact, these two arcs must intersect, because otherwise we are facing a plane embedding of K_5, which does not exist (Figure 8.13).

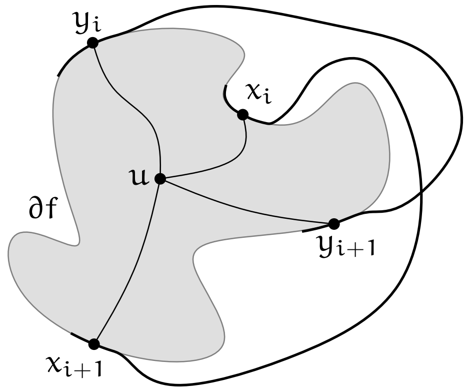


Figure 8.13: *Every quadruple* x_i, y_i, x_{i+1}, y_{i+1} *generates an intersection between the curves* ξ *and* η.

In other words, any quadruple of consecutive elements from the alternating subsequence induces an intersection between the corresponding arcs ξ and η. Clearly these intersection points are pairwise distinct for any pair of distinct quadruples which altogether provides $s+4-3 = s+1$ points of intersection between ξ and η, in contradiction to the assumption that these curves intersect in at most s points. □

Corollary 8.35. *The combinatorial complexity of a single face in an arrangement of* n *line segments in* $\mathbb{R}^2$ *is* $O(\lambda_3(n)) = O(n\alpha(n))$.

Exercise 8.36.

a) Show that for every fixed $k \geqslant 2$ *we have* $\alpha_k(n) = o(\alpha_{k-1}(n))$*; in fact, for every fixed* k *and* j *we have* $\alpha_k(n) = o(\alpha_{k-1}(\alpha_{k-1}(\cdots\alpha_{k-1}(n)\cdots)))$*, with* j *applications of* α_{k-1}.

b) Show that for every fixed k *we have* $\alpha(n) = o(\alpha_k(n))$.

It is a direct consequence of the symmetry in the definition that the property of being a Davenport-Schinzel sequence is invariant under permutations of the alphabet. For instance, $\sigma = bcacba$ is a $(3,3)$-DS sequence over $A = \{a, b, c\}$. Hence the permutation $\pi = (ab)$ induces a $(3,3)$-DS sequence $\pi(\sigma) = acbcab$ and similarly $\pi' = (cba)$ induces another $(3,3)$-DS sequence $\pi'(\sigma) = abcbac$.

When counting the number of Davenport-Schinzel sequences of a certain type we want to count *essentially distinct* sequences only. Therefore we call two sequences over a common alphabet A *equivalent* if and only if one can be obtained from the other by a permutation of A. Then two sequences are *distinct* if and only if they are not equivalent. A typical way to select a representative from each equivalence class is to order the alphabet and demand that the first appearance of a symbol in the sequence follows that order. For example, ordering $A = \{a, b, c\}$ alphabetically demands that the first occurrence of a precedes the first occurrence of b, which in turn precedes the first occurrence of c.

Exercise 8.37. *Let* P *be a convex polygon with* $n+1$ *vertices. Find a bijection between the triangulations of* P *and the set of pairwise distinct* $(n,2)$*-Davenport-Schinzel sequences of maximum length (*$2n-1$*). It follows that the number of distinct maximum* $(n,2)$*-Davenport-Schinzel sequences is exactly* $C_{n-1} = \frac{1}{n}\binom{2n-2}{n-1}$*, which is the* $(n-1)$*-st Catalan number.*

Questions

37. *How can one construct an arrangement of lines in* $\mathbb{R}^2$*?* Describe the incremental algorithm and prove that its time complexity is quadratic in the number of lines (incl. statement and proof of the Zone Theorem).

38. *How can one test whether there are three collinear points in a set of* n *given points in* $\mathbb{R}^2$*?* Describe an $O(n^2)$ time algorithm.

39. *How can one compute the minimum area triangle spanned by three out of* n *given points in* $\mathbb{R}^2$*?* Describe an $O(n^2)$ time algorithm.

40. *What is a ham sandwich cut? Does it always exist? How to compute it?* State and prove the theorem about the existence of a ham sandwich cut in $\mathbb{R}^2$ and sketch the $O(n \log n)$ algorithm by Edelsbrunner and Waupotitsch.

41. *What is the endpoint visibility graph for a set of disjoint line segments in the plane and how can it be constructed?* Give the definition and explain the relation to shortest paths. Describe the $O(n^2)$ algorithm by Welzl, including full proofs of Theorem 8.11 and Theorem 8.14.

42. *Is there a subquadratic algorithm for General Position?* Explain the term 3-Sum hard and its implications and give the reduction from 3-Sum to General Position.

43. *Which problems are known to be 3-Sum-hard?* List at least three problems (other than 3-Sum) and briefly sketch the corresponding reductions.

44. *What is an (n, s) Davenport-Schinzel sequence and how does it relate to the lower envelope of real-valued continuous functions?* Give the precise definition and some examples. Explain the relationship to lower envelopes and how to apply the machinery to partial functions like line segments.

45. *What is the value of $\lambda_1(n)$ and $\lambda_2(n)$?*

46. *What is the asymptotic value of $\lambda_3(n)$, $\lambda_4(n)$, and $\lambda_s(n)$ for larger s?*

47. *What is the combinatorial complexity of the lower envelope of a set of n lines/parabolas/line segments?*

48. *What is the combinatorial complexity of a single face in an arrangement of n line segments?* State the result and sketch the proof (Theorem 8.34).

References

[1] Radek Adamec, Martin Klazar, and Pavel Valtr, Generalized Davenport-Schinzel sequences with linear upper bound. *Discrete Math.*, **108**, (1992), 219–229.

[2] Pankaj K. Agarwal and Micha Sharir, *Davenport-Schinzel sequences and their geometric applications*, Cambridge University Press, New York, NY, 1995.

[3] Pankaj K. Agarwal, Micha Sharir, and Peter W. Shor, Sharp upper and lower bounds on the length of general Davenport-Schinzel sequences. *J. Combin. Theory Ser. A*, **52**, 2, (1989), 228–274.

[4] Manuel Blum, Robert W. Floyd, Vaughan Pratt, Ronald L. Rivest, and Robert E. Tarjan, Time bounds for selection. *J. Comput. Syst. Sci.*, **7**, 4, (1973), 448–461.

[5] Herbert Edelsbrunner and Roman Waupotitsch, Computing a ham-sandwich cut in two dimensions. *J. Symbolic Comput.*, **2**, (1986), 171–178.

[6] Jeff Erickson, Lower bounds for linear satisfiability problems. *Chicago J. Theoret. Comput. Sci.*, **1999**, 8.

[7] Anka Gajentaan and Mark H. Overmars, On a class of $O(n^2)$ problems in computational geometry. *Comput. Geom. Theory Appl.*, **5**, (1995), 165–185.

[8] Allan Grønlund and Seth Pettie, Threesomes, degenerates, and love triangles. *J. ACM*, **65**, 4, (2018), 22:1–22:25.

[9] Branko Grünbaum, Hamiltonian polygons and polyhedra. *Geombinatorics*, **3**, 3, (1994), 83–89.

[10] Sergiu Hart and Micha Sharir, Nonlinearity of Davenport-Schinzel sequences and of generalized path compression schemes. *Combinatorica*, **6**, (1986), 151–177.

[11] Michael Hoffmann, *On the existence of paths and cycles*. Ph.D. thesis, ETH Zürich, 2005.

[12] Michael Hoffmann and Csaba D. Tóth, Segment endpoint visibility graphs are Hamiltonian. *Comput. Geom. Theory Appl.*, **26**, 1, (2003), 47–68.

[13] Daniel M. Kane, Shachar Lovett, and Shay Moran, Near-optimal linear decision trees for k-SUM and related problems. *J. ACM*, **66**, 3, (2019), 16:1–16:18.

[14] Martin Klazar, On the maximum lengths of Davenport-Schinzel sequences. In R. Graham et al., ed., *Contemporary Trends in Discrete Mathematics*, vol. 49 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pp. 169–178, Amer. Math. Soc., Providence, RI, 1999.

[15] Christian Knauer, Hans Raj Tiwary, and Daniel Werner, On the computational complexity of ham-sandwich cuts, Helly sets, and related problems. In *Proc. 28th Sympos. Theoret. Aspects Comput. Sci.*, vol. 9 of *LIPIcs*, pp. 649–660, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2011.

[16] Chi-Yuan Lo, Jiří Matoušek, and William L. Steiger, Algorithms for ham-sandwich cuts. *Discrete Comput. Geom.*, **11**, (1994), 433–452.

[17] Jiří Matoušek, *Using the Borsuk–Ulam theorem*, Springer-Verlag, Berlin, 2003.

[18] Gabriel Nivasch, Improved bounds and new techniques for Davenport-Schinzel sequences and their generalizations. *J. ACM*, **57**, 3, (2010), Article No. 17.

[19] Seth Pettie, Splay trees, Davenport-Schinzel sequences, and the deque conjecture. In *Proc. 19th ACM-SIAM Sympos. Discrete Algorithms*, pp. 1115–1124, 2008.

[20] Masatsugu Urabe and Mamoru Watanabe, On a counterexample to a conjecture of Mirzaian. *Comput. Geom. Theory Appl.*, **2**, 1, (1992), 51–53.

[21] Emo Welzl, Constructing the visibility graph for n line segments in $O(n^2)$ time. *Inform. Process. Lett.*, **20**, (1985), 167–171.

Chapter 9

Counting

We take a tour through several questions in algorithmic and extremal counting of (geometrically defined) combinatorial objects, with emphasis on how they connect to each other in their solutions. Among these problems are (i) counting the number of simplices spanned by $d+1$ points in a finite set of n points in d-space that contain a given point (simplicial depth), (ii) counting the number of facets of the convex hull of n points in d-space, (iii) investigating the minimal number of crossings a drawing of the complete graph with straight line edges in the plane must have, (iv) counting of crossing-free geometric graphs of several types on so-called wheel-sets (point sets in the plane with all but one point extremal),

Notation.

$0 := (0, 0, \ldots, 0)$ is the origin in the considered ambient space.

$\mathbb{N}$ is the set of positive integers, $\mathbb{N}_0 := \mathbb{N} \cup \{0\}$.

$\operatorname{conv}(S)$ denotes the convex hull of a given set S in $\mathbb{R}^d$.

$\binom{S}{k}$ denotes the set of all k-element subsets of a given set S.

Sometimes it may be useful to remember

$$\sum_{i=0}^{n-1} \binom{i}{k-1} = \binom{n}{k} = \binom{n-1}{k} + \binom{n-1}{k-1} .$$

Checkpoints are usually simple facts that are put there to check your understanding of definitions or notions, to be answered perhaps in a minute or two (assuming you have indeed absorbed the definition).

9.1 Introduction

Consider a set $P \subseteq \mathbb{R}^d$ and $q \in \mathbb{R}^d$. A set $A \in \binom{P}{d+1}$ is called q-*embracing simplex* if $q \in \operatorname{conv}(A)$.

The simplicial depth, $\mathrm{sd}_q(P)$, of point q relative to P is the number of q-embracing simplices, i.e.

$$\mathrm{sd}_q(P) := \left| \left\{ A \in \binom{P}{d+1} \;\middle|\; q \in \operatorname{conv}(A) \right\} \right| .$$

This notion, among others, is a possible response to the search for a higher-dimensional counterpart of the notion of a median in $\mathbb{R}^1$. Note here that when specialized to $\mathbb{R}^1$, a median is a point of maximum simplicial depth. We will investigate here this notion, asking questions like:

> *What is the maximum possible simplicial depth a point can have in any set of* n *points in general position?*
>
> *How efficiently can we compute the simplicial depth of a point?*

A second question we want to address is that of the complexity of polytopes in general dimension d.

> *How many facets can a polytope obtained as the convex hull of* n *points have, how few?*
>
> *Given* n *points, how efficiently can we compute the number of facets? Can we do that asymptotically faster than enumerating these facets (which is a hard enough problem per se)?*

A small caveat, in case you are not familiar with this: We know that a 2-dimensional polytope with n vertices has n facets (here edges), and a 3-dimensional polytope with n vertices has at most $2n-4$ facets (if the vertices are in general position, it is exactly this number). So the number of facets is linear in n. This last fact is not true in higher dimension and we will see what the right bounds are.

We will see that these two types of questions about simplicial depth and number of facets of a polytope are very closely related; in some sense, that we will make very explicit, it is the same question.

On the side, we will also indicate how these connect to other problems as indicated in the abstract.

9.2 Embracing k-Sets in the Plane

In this section we investigate simplicial depth in the plane, and generalize by considering arbitrarily large sets with a given point in their convex hull. This will allow us to introduce some of the technicalities in this simpler (planar) context, and we will see later that the extension to larger sets was unavoidable, even when we are interested in simplicial depth only.

Consider a set $P \subseteq \mathbb{R}^2$, with $0 \notin P$ and $P \,\dot\cup\, \{0\}$ in general position (no three on a line); $n := |P|$. This setting will be implicitly assumed throughout the section. For $k \in \mathbb{N}_0$, we define

$$e_k = e_k(P) := \left| \left\{ A \in \binom{P}{k} \;\middle|\; 0 \in \operatorname{conv}(A) \right\} \right| .$$

We call A with $0 \in \operatorname{conv}(A)$ an *embracing* k*-set* – if $|A| = 3$, we call A an *embracing triangle.*

Checkpoint 9.1. e_3 *is the simplicial depth of* 0 *in* P*, i.e.* $e_3 = \mathrm{sd}_0(P)$. $e_0 = e_1 = e_2 = 0$, $e_n \in \{0, 1\}$.

We start a general investigation of $e = (e_0, e_1, \dots, e_n)$. Bounds and algorithms will follow easily. For a preparatory step consider real vectors $x_{0..n-3} = (x_0, x_1, \dots, x_{n-3})$, $y_{0..n-2}$ and $z_{0..n-1}$ satisfying

$$e_k = \sum_{i=0}^{n-3} \binom{i}{k-3} x_i \,, \quad \text{for } k \geqslant 3, \tag{9.2}$$

$$e_k = \sum_{i=0}^{n-2} \binom{i}{k-2} y_i \,, \quad \text{for } k \geqslant 2, \text{ and} \tag{9.3}$$

$$e_k = \sum_{i=0}^{n-1} \binom{i}{k-1} z_i \,, \quad \text{for } k \geqslant 1. \tag{9.4}$$

Observe that $x_{0..n-3}$ exists and is uniquely determined by $e_{3..n}$, since

$$\begin{aligned} e_n &= \tbinom{n-3}{n-3} x_{n-3} && \Rightarrow x_{n-3} = e_n \\ e_{n-1} &= \tbinom{n-4}{n-4} x_{n-4} + \tbinom{n-3}{n-4} x_{n-3} && \Rightarrow x_{n-4} = e_{n-1} - (n-3) \underbrace{x_{n-3}}_{e_n} \\ &\vdots \end{aligned}$$

Similarly, this works for $y_{0..n-2}$ and $z_{0..n-1}$. Thus we have

$$e_{3..n} \underset{\text{each other}}{\overset{\text{determine}}{\longleftrightarrow}} x_{0..n-3}, \quad e_{2..n} \underset{\text{each other}}{\overset{\text{determine}}{\longleftrightarrow}} y_{0..n-2}, \quad e_{1..n} \underset{\text{each other}}{\overset{\text{determine}}{\longleftrightarrow}} z_{0..n-1} \,.$$

For now, it is by no means clear what that should help here. Note also that these facts are true for any vector e, we have not used any of the properties of the specific vector we are interested in. They simply describe one of many possible transformations of a given vector.

9.2.1 Adding a Dimension

Another step that comes across unmotivated: Lift the point set P vertically to a set P' in space, arbitrarily, with the only condition that P' is in general position (no four on a plane).[1] We denote the map by

$$P \ni \quad q = (x, y) \mapsto q' = (x, y, z(q)) \quad \in P' .$$

For an embracing triangle $\Delta = \{p, q, r\}$ in the plane, let β_Δ be the number of points in P' below the plane containing $\Delta' = \{p', q', r'\}$. (Just to avoid confusion: β_Δ clearly depends on the choice of the lifting P'.) Let

$$h_i = h_i(P') := \text{ the number of embracing triangles } \Delta \text{ with } \beta_\Delta = i.$$

Checkpoint 9.5. $\sum_{i=0}^{n-3} h_i = e_3$.

Let us recall here that we are assuming general position for $P \,\dot\cup\, \{0\}$.

Lemma 9.6. $0 \in \text{conv}(P) \iff h_0 = h_{n-3} = 1$.

Proof. ($\Leftarrow$) That's obvious, since $h_0 = 1$ means that there is some embracing triangle, and therefore 0 is in the convex hull of P.

($\Rightarrow$) Note that $0 \in \text{conv}(P)$ iff the z-axis (i.e. the vertical line through 0) intersects $\text{conv}(P')$. There are exactly two facets (triangles because of general position) intersected by the z-axis, the bottom one, Δ_0' has no point in P' below its supporting plane, hence, $\beta_{\Delta_0} = 0$; the top one, Δ_1' has no point in P' above and hence $n - 3$ points in P' below (all but the three points defining the facet), hence, $\beta_{\Delta_1} = n-3$. Since any triple $\Delta' \subseteq P'$ with all points in P' on one side (above or below) must give rise to a facet, it cannot be hit by the z-axis, unless $\Delta' = \Delta_0'$ or $\Delta' = \Delta_1'$. □

Consider an embracing k-set A and its lifting A'. As observed before, in $\mathbb{R}^3$ the vertical line through 0 will intersect the boundary of $\text{conv}(A')$ in two facets. Consider the top facet – its vertices are liftings of some embracing triangle Δ in the plane. We call this Δ the *witness of* (the embracing property of) A. For how many embracing k-sets is Δ the witness?

For Δ to be witness of an embracing k-set B, we must have $\Delta \subseteq B$ and the remaining $k-3$ points in $B \setminus \Delta$ must be chosen so that $B' \setminus \Delta'$ lies below the plane spanned by Δ'. Hence Δ is witness for exactly $\binom{\beta_\Delta}{k-3}$ embracing k-sets. It follows that

$$e_k = \sum_{\Delta \text{ embracing}} \binom{\beta_\Delta}{k-3} = \sum_{i=0}^{n-3} \binom{i}{k-3} h_i \,. \tag{9.7}$$

[1] For example, choose the lifting map $(x, y) \mapsto (x, y, x^2 + y^2)$ ... but stay flexible!

That is, the h_i's are exactly the x_i's defined by equations (9.2). As observed before, we thus have

$$e_{3..n} \underset{\text{each other}}{\overset{\text{determine}}{\longleftrightarrow}} h_{0..n-3} := (h_0, h_1, \ldots, h_{n-3})$$

and therefore the vector $h_{0..n-3}$ is independent of the lifting we chose, i.e. $h_i = h_i(P)$.

A few properties emerge. First note that h (consisting of nonnegative integers, each at most $\binom{n}{3} = O(n^3)$, i.e. $O(\log n)$ bits) is a compact way of representing e (with numbers, some may be exponential in n, with $\Omega(n)$ bits). Also, since it is easy to compute the vector h in $O(n^4)$ time[2], we can compute each entry of e_k in $O(n^4)$ time.

Exercise 9.8. *Show*

$$h_0 = 1 \quad \Leftrightarrow \quad 0 \in \operatorname{conv}(P) \quad \Leftrightarrow \quad h_i \geqslant 1 \text{ for } 0 \leqslant i \leqslant n-3 \ .$$

Exercise 9.9. *Assume* $0 \in \operatorname{conv}(P)$. *(i) What is the minimal possible value of* e_3 *in terms of* $n := |P|$*? (Note that this gives a quantified version of Carathéodory's Theorem.) (ii) What is the minimal possible value of* e_k, $3 \leqslant k \leqslant n$*?*

Exercise 9.10. *What does* $\sum_{i=0}^{n-3} 2^i h_i$ *count?*

Exercise 9.11. *Show* $\sum_{k=3}^{n} (-1)^k e_k = -1$ *provided* $0 \in \operatorname{conv}(P)$.
(Hint: Plug in $\sum_{i=0}^{n-3} \binom{i}{k-3} h_i$ *for* e_k *in this sum and simplify.)*

In a next step we show that the vector h is symmetric.

Lemma 9.12. $h_i = h_{n-3-i}$.

Proof. Define $\hat{h}_i$ in the same way as h_i, except that you count the points *above* (instead of below) the plane through the lifting of an embracing triangle. First, note that $\hat{h}_i = h_{n-3-i}$. And clearly, (with the same witness argument as before)

$$e_k = \sum_{i=0}^{n-3} \binom{i}{k-3} \hat{h}_i \ ,$$

and, therefore,

$$h_i = \hat{h}_i = h_{n-3-i} \ .$$

□

That is, vector $h_{0..n-3}$ is determined by entries $h_0, h_1, \ldots, h_{\lfloor (n-3)/2 \rfloor}$.

Exercise 9.13. *Show* $(n-3)e_3 = 2e_4$.

Exercise 9.14. *Show that if* $|P| = 6$*, then* e_3 *determines* $e_{3..6}$*. How?*

Exercise 9.15. *Show that if* $|P|$ *is even then* e_3 *is even.*

[2]With some basics from computational geometry, in $O(n^3)$ time.

9.2.2 The Upper Bound

We have seen in one of the exercises how the relation between e and h can be useful in proving lower bounds on the e_k's. We need two lemmas towards a proof of upper bounds.

The first lemma states that removing a point in P cannot increase h_j.

Lemma 9.16. *For all* $q \in P$, $h_j(P \setminus \{q\}) \leqslant h_j(P)$.

Proof. Which changes happen to h_j as we remove a point q in P?

- *We lose* embracing triangles Δ with j points below (in the lifting), one of which is q. And we lose embracing triangles Δ, where q is a defining point (i.e. $q \in \Delta$).
- *We keep* embracing triangles Δ with j points below, and q above.
- *We gain* embracing triangles Δ with $j+1$ points below, with q one of those below.

Now move q' (in the lifting) vertically above all planes defined by three points in $P' \setminus \{q'\}$. This does not change the values h_i (since, again, h is independent of the lifting), and the case of "We gain" cannot occur. This gives the lemma. □

Lemma 9.17. $\sum_{q \in P} h_j(P \setminus \{q\}) = (n-j-3)h_j(P) + (j+1)h_{j+1}(P)$.

Proof. A contribution to $\sum_{q \in P} h_j(P \setminus \{q\})$ can come only from triangles Δ with $\beta_\Delta = j$ or $\beta_\Delta = j+1$ (relative to the complete set P and a chosen lifting P').

- If $\beta_\Delta = j$, Δ' remains a triangle with j points below, if q is chosen as one of the $(n-3-j)$ points above.
- If $\beta_\Delta = j+1$, Δ' turns into a triangle with j points below, if q is chosen as one of the $(j+1)$ points below.

Hence the lemma. □

Now we recall the previous Lemma 9.16 to bound the sum in Lemma 9.17:

$$\sum_{q \in P} h_j(P \setminus \{q\}) \leqslant n \cdot h_j(P) \ ,$$

and with this we can derive

$$\begin{aligned} (n-j-3)h_j(P) + (j+1)h_{j+1}(P) &\leqslant n \cdot h_j(P) \\ (j+1)h_{j+1}(P) &\leqslant (j+3)h_j(P) \\ h_{j+1}(P) &\leqslant \frac{j+3}{j+1} h_j(P) \ . \end{aligned}$$

This bound can be iterated until we reach h_0:

$$h_{j+1}(P) \leqslant \tfrac{j+3}{j+1} h_j(P) \leqslant \tfrac{j+3}{j+1}\tfrac{j+2}{j} h_{j-1}(P) \leqslant \underbrace{\frac{j+3}{j+1}\frac{j+2}{j}\cdots\frac{3}{1}}_{=\binom{j+3}{2}} \underbrace{h_0(P)}_{\leqslant 1} \leqslant \binom{j+3}{2} \ .$$

Theorem 9.18. *Let* P *be a set of* n *points in general position.*

(i) For all j, $0 \leqslant j \leqslant n-3$,

$$h_j = h_{n-3-j} \quad \textit{and} \quad h_j \leqslant \binom{j+2}{2}$$

and hence $h_j \leqslant \min\{\binom{j+2}{2}, \binom{n-1-j}{2}\}$.

(ii)

$$e_3 \leqslant \begin{cases} 2\binom{n/2+1}{3} = \frac{n(n^2-4)}{24} & \textit{for } n \textit{ even, and} \\ 2\binom{(n+1)/2}{3} + \binom{(n+1)/2}{2} = \frac{n(n^2-1)}{24} & \textit{for } n \textit{ odd.} \end{cases}$$

Proof. (i) is just a summary of what we have derived so far.

For (ii) we simply plug these bounds into relation (9.7). Suppose first that n is even. Then

$$(h_0, h_1, \dots, h_{n/2-2}) = (h_{n-3}, h_{n-2}, \dots, h_{n/2-1})$$

and, therefore,

$$e_3 = \sum_{i=0}^{n-3} h_i = 2 \sum_{i=0}^{n/2-2} h_i \leqslant 2 \sum_{i=0}^{n/2-2} \binom{i+2}{2} = 2\binom{n/2+1}{3}.$$

Second, if n is odd then

$$(h_0, h_1, \dots, h_{(n-3)/2}) = (h_{n-3}, h_{n-2}, \dots, h_{(n-3)/2})$$

with $h_{(n-3)/2}$ appearing in both sequences. Then

$$\begin{aligned} e_3 = \sum_{i=0}^{n-3} h_i &= 2 \sum_{i=0}^{(n-3)/2-1} h_i + h_{(n-3)/2} \\ &\leqslant 2 \sum_{i=0}^{(n-3)/2-1} \binom{i+2}{2} + \binom{(n+1)/2}{2} \\ &= 2\binom{(n+1)/2}{3} + \binom{(n+1)/2}{2}. \end{aligned}$$

□

There are sets where all these bounds are tight, simultaneously. We find it more convenient to substantiate this claim after some further considerations.

Exercise 9.19. *Show* $e_3 \leqslant \frac{1}{4}\binom{n}{3} + O(n^2)$. *(That is, asymptotically, at most* $1/4$ *of all triangles embrace the origin.)*

Exercise 9.20. *Try to understand the independence of* h *of the actual lifting by observing what happens as you move a single point vertically.*

While we have successfully obtained lower and upper bounds, we will next give a better method for computing the e_k's.

9.2.3 Faster Counting—Another Vector

Call a directed edge $0q$, $q \in P$, an i-*edge*, if i points in P lie to the left of the directed line through $0q$ (directed from 0 to q). Let $\ell_i = \ell_i(P)$ be the number of i-edges of P.

Checkpoint 9.21. $\sum_i \ell_i = n$. *What is the vector* $l = (\ell_0, \ell_1, \ldots, \ell_{n-1})$ *for the case* $0 \notin \mathrm{conv}(P)$*?*

For every nonempty set $A \subseteq P$ with $0 \notin \mathrm{conv}(A)$, $\mathrm{conv}(A)$ has a left and a right tangent from 0. Let $q \in A$ be the touching point of the right tangent. For how many k-element sets $A \subseteq P$ with $0 \notin \mathrm{conv}(A)$ is this point q the right touching point?

Checkpoint 9.22. *This is* $\binom{i}{k-1}$ *if* $0q$ *is an* i-*edge.*

Hence, we have for $1 \leqslant k \leqslant n$:

$$e_k = \underbrace{\binom{n}{k}}_{\sum_{i=0}^{n-1}\binom{i}{k-1}} - \sum_{i=0}^{n-1}\binom{i}{k-1}\ell_i = \sum_{i=0}^{n-1}\binom{i}{k-1}(1-\ell_i)\ . \tag{9.23}$$

We have a combinatorial interpretation of the z_i's in (9.4) and, therefore, numbers ℓ_i satisfying (9.23) are unique.

Exercise 9.24. *Show that* $\ell_i = \ell_{n-1-i}$. *(Hint: Wonder why we chose "left" and not "right".)*

We can compute the vector $l_{0..n-1}$ in $O(n \log n)$ time. For that we rotate a directed line about 0, starting with the horizontal line, say. We always maintain the number of points left of this line, and update this number whenever we sweep over a point $q \in P$. This q may lie ahead of 0 or behind it; depending on this the number increases by 1 or decreases by 1, resp. In this way, with a rotation by 180 degrees, we can compute the "number of points to the left" for every $q \in P$. We need $O(n \log n)$ time to sort the events (encounters with points in P). We initialize the "number to the left" in $O(n)$ time in the beginning, and then update the number in $O(1)$ at each event. This gives $O(n \log n)$ altogether.

Theorem 9.25. *In the plane, the simplicial depth* $\mathrm{sd}_q(P)$ *can be computed in* $O(n \log n)$ *time, provided* $P \,\dot\cup\, \{q\}$ *is in general position.*

Clearly, all entries e_k, $1 \leqslant k \leqslant n$, can be computed based on the vector l. However, keep in mind that the binomial coefficients involved in the sum (9.23) must be determined and that the numbers involved are large (up to n-bit numbers).

Showing that the upper bound in Theorem 9.18 is tight is now actually easy.

If P is the set of vertices of a regular n-gon, n odd, centered at 0, then $\ell_{(n-1)/2} = n$ (and all other ℓ_i's vanish). Therefore,

$$e_3 = \binom{n}{3} - \binom{(n-1)/2}{2} \cdot n = \frac{n(n^2-1)}{24} ,$$

and the case of n odd is shown tight in Theorem 9.18.

For n even, consider the vertices of a regular n-gon centered at 0, and let P be a slightly perturbed set of these vertices so that $P \cup \{0\}$ is in general position. Note that all edges $0q$, $q \in P$, must be $(n/2-1)$- or $(n/2)$-edges. Interestingly, because of the symmetry of the ℓ-vector, we immediately know that $\ell_{n/2-1} = \ell_{n/2} = n/2$ (with all other ℓ_i's vanishing), independent of our perturbation. Now

$$e_3 = \binom{n}{3} - \left(\binom{n/2-1}{2} + \binom{n/2}{2}\right)\frac{n}{2} = \frac{n(n^2-4)}{24} ,$$

and Theorem 9.18 is proven tight also for n even.

A next step is to understand what the possible ℓ-vectors for n points are, and in this way characterize and eventually count all possibilities for l and thus for e.

9.2.4 Characterizing All Possibilities

We start with two observations about properties of l.

Exercise 9.26. *Show that $\ell_{\lfloor (n-1)/2 \rfloor} \geqslant 1$. (There is always a halving edge.)*

Exercise 9.27. *Show that if $\ell_i \geqslant 1$ for some $i \leqslant \lfloor (n-1)/2 \rfloor$, then $\ell_j \geqslant 1$ for all j, $i \leqslant j \leqslant \lfloor (n-1)/2 \rfloor$.*

We summarize our knowledge about l.

Theorem 9.28. *For $n \in \mathbb{N}$, the vector $\mathrm{l}_{0..n-1}$ of an n-point set satisfies the following conditions.*

- *All entries are nonnegative integers.*
- $\sum_{i=0}^{n-1} \ell_i = n$.
- $\ell_i = \ell_{n-1-i}$. *(Symmetry)*
- *If $\ell_i \geqslant 1$ for some $i \leqslant \lfloor (n-1)/2 \rfloor$, then $\ell_j \geqslant 1$ for all j, $i \leqslant j \leqslant \lfloor (n-1)/2 \rfloor$. (remains positive towards the middle)*

Let us call a vector of length n a *legal n-vector* if the conditions of Theorem 9.28 are satisfied. Then (1) is the only legal 1-vector, $(1,1)$ is the only legal 2-vector, and the only legal 3-vectors are $(0,3,0)$ and $(1,1,1)$. The following scheme displays how we derive legal 6-vectors from legal 5-vectors, and how we can derive legal 7-vectors from legal 5- or 6-vectors.

$$\begin{tikzcd}
\overbrace{\begin{matrix}0&0&5&0&0\\0&1&3&1&0\\0&2&1&2&0\\1&1&1&1&1\end{matrix}}^{n=5} \arrow[r, "\text{add 1 in the middle and split}"] \arrow[d, "\text{add 2 in the middle}"'] & \overbrace{\begin{matrix}0&0&3&3&0&0\\0&1&2&2&1&0\\0&2&1&1&2&0\\1&1&1&1&1&1\end{matrix}}^{n=6} \arrow[d, "\text{insert a 1 in the middle}"] \\
\underbrace{\begin{matrix}0&0&0&7&0&0&0\\0&0&1&5&1&0&0\\0&0&2&3&2&0&0\\0&1&1&3&1&1&0\end{matrix}}_{n=7,\text{ with }>1\text{ in the middle}} & \underbrace{\begin{matrix}0&0&3&1&3&0&0\\0&1&2&1&2&1&0\\0&2&1&1&1&2&0\\1&1&1&1&1&1&1\end{matrix}}_{n=7,\text{ with }=1\text{ in the middle}}
\end{tikzcd}$$

Exercise 9.29. *Show that the scheme described, when applied to general n odd, is complete. That is, starting with all legal n-vectors, n odd, we get all legal $(n+1)$-vectors, and from the n- and $(n+1)$-vectors, we get all $(n+2)$-vectors.*

Exercise 9.30. *Show that the number of legal n-vectors is exactly $2^{\lfloor (n-1)/2 \rfloor}$.*

Exercise 9.31. *Show that every legal n-vector is the ℓ-vector of some set of n points in general position.*

With these exercises settled, we have given a complete characterization of all possible ℓ-vectors, thus of all possible e-vectors.

Theorem 9.32. *The number of different e-vectors (or ℓ-vectors) for n points is exactly $2^{\lfloor (n-1)/2 \rfloor}$.*

Exercise 9.33. *Show that $\sum_{i=0}^{j} \ell_i \leqslant j+1$ for all $0 \leqslant j \leqslant \lfloor (n-1)/2 \rfloor$. (Hint: Otherwise, we get into conflict with "remains positive towards the middle").*

9.2.5 Some Add-Ons

We are still missing an interpretation of the y_i's in relations (9.3). We want to leave this as an exercise.

Exercise 9.34. *For a set P of n points in general position, consider the vector*

$$(b_0, b_1, \ldots, b_{n-2})$$

defined by the relations

$$e_k = \binom{n}{k} - \sum_{i=0}^{n-2} \binom{i}{k-2} b_i = \sum_{i=0}^{n-2} \binom{i}{k-2} (n-i-1-b_i) \ ,$$

for $2 \leqslant k \leqslant n$. Give a combinatorial interpretation of these numbers b_i, $0 \leqslant i \leqslant n-2$.

Finally, let us investigate how the vectors x, y, and z from relations (9.2), (9.3), and (9.4) connect to each other. Clearly, e_1 and e_2 given, they determine each other. But how? This will allow us to relate the vectors h and l.

Exercise 9.35. *Consider the relations defined in the beginning of this section on $x_{0..n-3}, y_{0..n-2}$, $z_{0..n-1}$, and $e_{1..n}$ (using $e_1 = e_2 = 0$). Then the y_i's are the forward differences of the x_i, and the z_i's are the forward differences of the y_i's. Prove this. More concretely, show that*

$$y_i = \begin{cases} -x_0 & i = 0 \\ x_{i-1} - x_i & 1 \leqslant i \leqslant n-3 \\ x_{n-3} & i = n-2 \end{cases}$$

or, equivalently,

$$y_i = x_{i-1} - x_i \quad \textit{for all } 0 \leqslant i \leqslant n-2, \textit{ with } x_{-1} := x_{n-2} := 0.$$

Show that this entails as well

$$x_i = -\sum_{j=0}^{i} y_j \quad , \textit{for } 0 \leqslant i \leqslant n-3.$$

Exercise 9.36. *Prove for vectors $a_{0..m}$ and $b_{0..m}$*

$$\begin{aligned} & \forall k, 0 \leqslant k \leqslant m: \quad a_k = \sum_{i=0}^{m} \binom{i}{k} b_i \\ \iff & \forall i, 0 \leqslant i \leqslant m: \quad b_i = \sum_{k=0}^{m} (-1)^{i+k} \binom{k}{i} a_k \ . \end{aligned}$$

Exercise 9.37. *Employing the previous exercise, what does* $h_0 = 1$ *say about* $e_{3..n}$.

The following facts can now be readily derived.

Theorem 9.38.

$$h_i = \binom{i+2}{2} - \sum_{j=0}^{i} (i+1-j)\ell_j$$

Exercise 9.39. *Prove Theorem 9.38.*

Note that this implies the upper bounds we proved for the h_i's in Theorem 9.18, since $\sum_{j=0}^{i}(i+1-j)\ell_j$ is always nonnegative. Moreover, a combinatorial interpretation of the slack becomes evident.

Theorem 9.40.

$$e_k = \sum_{i=0}^{n} \binom{i}{k} (\ell_i - \ell_{i-1}) \quad \textit{with } \ell_{-1} = \ell_n = 1$$

Exercise 9.41. *Prove Theorem 9.40.*

Let us point out other counting problems which can be solved efficiently with the insights developed.

Exercise 9.42. *Given a ray* r *(emanating from point* q*) and* n *points* P *in the plane, design an efficient algorithm that counts the number of points connecting segments intersecting* r. *You may assume that* $P \cup \{q\}$ *is in general position and that* r *is disjoint from* P.

Exercise 9.43. *Let* w *be a line minus an interval on it (an infinite wall with a window). Given* n *points* P *in the plane, design an efficient algorithm that counts the number of point connecting segments disjoint from* w *(i.e. the number of pairs of points that see each other, either because they are both on the same side of* w *or because they see each other through the window in* w. *You may assume general position.*

Exercise 9.44. *Recall that a point* c *is a centerpoint of* P *if every halfplane containing* c *contains at least* $|P|/3$ *points in* P.

Identify the properties of e, h *and* l *that show that* 0 *is a centerpoint of* P.

Exercise 9.45. *Show that* $y_i = -y_{n-2-i}$ *and* $y_i \leqslant 0$ *for all* $0 \leqslant i \leqslant \lfloor \frac{n-2}{2} \rfloor$. *(We refer here to the* y_i*'s as defined by relations (9.3) at the beginning of this section. Hint: You may wish to recall Homework 9.35 and Exercise 9.33.)*

Exercise 9.46. *Show that* $h_i \geqslant h_{i-1}$ *for all* $0 \leqslant \lfloor \frac{n-3}{2} \rfloor$.

Questions

49. Explain how the h-vector of a planar point set is defined via a lifting. Give the relation between the e-vector (number of embracing k-sets) and the h-vector.

All of the following three questions also include Question 49.

50. Argue, why the h-vector is independent of the lifting.
51. Show how the ℓ-vector can be computed in $O(n \log n)$ time.
52. Argue why the ℓ-vector is symmetric ($\ell_i = \ell_{n-1-i}$ for all i, $0 \leqslant i \leqslant n-1$).

Chapter 10

Crossings

So far within this course we have mostly tried to avoid edge crossings and studied classes of planar graphs that allow us to avoid crossings altogether. However, without doubt there are many interesting graphs that are not planar, and still we would like to draw them in a reasonable fashion. An obvious quantitative approach is to still avoid crossings as much as possible, even if they cannot be avoided completely.

For a graph $G = (V, E)$, the *crossing number* $\mathrm{cr}(G)$ is defined as the minimum number of edge crossings over all possible drawings of G. In an analogous fashion, the *rectilinear crossing number* $\overline{\mathrm{cr}}(G)$ is defined as the minimum number of edge crossings over all possible straight-line drawings of G.

In order to see that thes notions are well-defined, let us first argue that the number of crossings in a minimum-crossing drawing is finite and, in fact, upper bounded by $\binom{|E|}{2}$.

Lemma 10.1. *In a drawing of a graph* G *with* $\mathrm{cr}(G)$ *crossings, every two distinct edges share at most one point.*

Proof. By a rerouting argument... □

In particular, by Lemma 10.1 no two adjacent edges (that is, edges that have a common endpoint) cross. A drawing that satisfies the statement of Lemma 10.1 is called a *simple topological drawing*. So, using this term, Lemma 10.1 could also be stated as "Every minimum-crossing drawing is a simple topological drawing."

It is quite easy to give an upper bound on the crossing number of a particular graph, simply by describing a drawing and counting the number of crossings in that drawing. Conversely, it is much harder to give a lower bound on the crossing number of a graph because such a bound corresponds to a statement about *all* possible drawings of that graph. But the following simple lower bound can be obtained by counting edges.

Lemma 10.2. *For a graph* G *with* $n \geqslant 3$ *vertices and* e *edges, we have* $\mathrm{cr}(G) \geqslant e - (3n - 6)$.

Proof. Consider a drawing of $G = (V, E)$ with $\mathrm{cr}(G)$ crossings. For each crossing, pick one of the two involved edges arbitrarily. Obtain a new graph $G' = (V, E')$ from G by removing all picked edges. By construction G' is plane and, therefore, $|E'| \leqslant 3n - 6$ by Corollary 2.5. As at most $\mathrm{cr}(G)$ edges were picked (some edge could be picked for several crossings), we have $|E'| \geqslant |E| - \mathrm{cr}(G)$. Combining both bounds completes the proof. □

The bound in Lemma 10.2 is quite good if the number of edges is close to $3n$ but not so good for dense graphs. For instance, for the complete graph K_n the lemma guarantees a quadratic number of crossings, whereas according to the Harary-Hill Conjecture [3]

$$\mathrm{cr}(K_n) = \frac{1}{4}\left\lfloor\frac{n}{2}\right\rfloor\left\lfloor\frac{n-1}{2}\right\rfloor\left\lfloor\frac{n-2}{2}\right\rfloor\left\lfloor\frac{n-3}{2}\right\rfloor \in \Theta(n^4).$$

So for a dense graph G we should try a different approach. Given that the bound in Lemma 10.2 is not so bad for sparse graphs, why not apply it to some sparse subgraph of G? This astonishingly simple idea turns out to work really well, as the following theorem demonstrates.

Theorem 10.3 (Crossing Lemma [2]). *For a graph G with n vertices and $e \geqslant 4n$ edges, we have $\mathrm{cr}(G) \geqslant e^3/(64n^2)$.*

Proof. Consider a drawing of G with $\mathrm{cr}(G)$ crossings. Take a random induced subgraph of G by selecting each vertex independently with probability p (a suitable value for p will be determined later). By this process we obtain a random subset $U \subseteq V$ and the corresponding induced subgraph $G[U]$, along with an induced drawing for $G[U]$. Consider the following three random variables:

- N, the number of vertices selected, with $E[N] = pn$;
- M, the number of edges induced by the selected vertices, with $E[M] = p^2 e$; and
- C, the number of crossings induced by the selected vertices and edges, with $E[C] = p^4\mathrm{cr}(G)$. (Here we use that adjacent edges do not cross in a minimum-crossing drawing by Lemma 10.1.)

According to Lemma 10.2, these quantities satisfy $C \geqslant \mathrm{cr}(G[U]) \geqslant M - 3N$. Taking expectations on both sides and using linearity of expectation yields $E[C] \geqslant E[M] - 3E[N]$ and so $p^4\mathrm{cr}(G) \geqslant p^2 e - 3pn$. Setting $p = 4n/e$ (which is $\leqslant 1$ due to the assumption $e \geqslant 4n$) gives

$$\mathrm{cr}(G) \geqslant \frac{e}{p^2} - 3\frac{n}{p^3} = \frac{e^3}{16n^2} - 3\frac{e^3}{64n^2} = \frac{e^3}{64n^2}$$

□

The constant $1/64$ in the statement of Theorem 10.3 is not the best possible. On one hand, Ackerman [1] showed that $1/64$ can be replaced by $1/29$, at the cost of requiring $e \geqslant 7n$. On the other hand, Pach and Tóth [4] describe graphs with $n \ll e \ll n^2$ that have crossing number at most

$$\frac{16}{27\pi^2}\frac{e^3}{n^2} < \frac{1}{16.65}\frac{e^3}{n^2}.$$

Hence it is not possible to replace $1/64$ by $1/16.65$ in the statement of Theorem 10.3.

In the remainder of this chapter, we will discuss several nontrivial bounds on the size of combinatorial structures that can be obtained by a more-or-less straightforward application of the Crossing Lemma. These beautiful connections were observed by Székely [5], the original proofs were different and more involved.

Theorem 10.4 (Szemerédi-Trotter [6]). *The maximum number of incidences between* n *points and* m *lines in* $\mathbb{R}^2$ *is at most* $\sqrt[3]{32} \cdot n^{2/3} m^{2/3} + 4n + m$.

Proof. Let P denote the given set of n points, and let L denote the given set of m lines. We may suppose that every line from L contains at least one point from P. (Discard all lines that do not; they do not contribute any incidence.) Denote by I the number of incidences between P and L. Consider the graph $G = (P, E)$ whose vertex set is P, and where a pair p, q of points is connected by an edge if p and q appear consecutively along some line $\ell \in L$ (that is, both p and q are incident to ℓ and no other point from P lies on the line segment $\overline{pq}$). Using the straight-line drawing induced by the arrangement of L we may regard G as a geometric graph with at most $\binom{m}{2}$ crossings.

Every line from L contains $k \geqslant 1$ point(s) from P and contributes $k-1$ edges to G. Hence $|E| = I - m$. If $|E| \leqslant 4n$, then $I \leqslant 4n + m$ and the theorem holds. Otherwise, we can apply the Crossing Lemma to obtain

$$\binom{m}{2} \geqslant \mathrm{cr}(G) \geqslant \frac{|E|^3}{64n^2}$$

and so $I \leqslant \sqrt[3]{32}\, n^{2/3} m^{2/3} + m$. □

Theorem 10.5. *The maximum number of unit distances determined by* n *points in* $\mathbb{R}^2$ *is at most* $5n^{4/3}$.

Proof. Let P denote the given set of n points, and consider the set C of n unit circles centered at the points in P. Then the number I of incidences between P and C is exactly twice the number of unit distances between points from P.

Define a graph $G = (P, E)$ on P where two vertices p and q are connected by an edge if they appear consecutively along some circle $c \in C$ (that is, $p, q \in c$ and at least one of the circular arcs of c between p and q does not contain any other point from P). Clearly $|E| = I$.

Obtain a new graph $G' = (P, E')$ from G by removing all edges along circles from C that contain at most two points from P. Note that $|C| = n$ and that every circle whose edges are removed contributes at most two edges to G. Therefore $|E'| \geqslant |E| - 2n$. In G' there are no loops and no two vertices are connected by two edges along the same circle. Therefore, any two vertices are connected by at most two edges because there are exactly two distinct unit circles passing through any two distinct points in $\mathbb{R}^2$.

Obtain a new graph $G'' = (P, E'')$ from G' by removing one copy of every double edge. Clearly G'' is a simple graph with $|E''| \geqslant |E'|/2 \geqslant (|E|/2) - n$. As every pair of circles intersects in at most two points, we have $\mathrm{cr}(G'') \leqslant 2\binom{n}{2} \leqslant n^2$.

If $|E''| \leqslant 4n$, then $(|E|/2) - n \leqslant 4n$ and so $I = |E| \leqslant 10n < 10n^{4/3}$ and the theorem holds. Otherwise, by the Crossing Lemma we have

$$n^2 \geqslant \mathrm{cr}(G'') \geqslant \frac{|E''|^3}{64n^2}$$

and so $|E''| \leqslant 4n^{4/3}$. It follows that $I = |E| \leqslant 8n^{4/3} + 2n \leqslant 10n^{4/3}$. □

Theorem 10.6. *For $A \subset \mathbb{R}$ with $|A| = n \geqslant 3$ we have $\max\{|A + A|, |A \cdot A|\} \geqslant \frac{1}{4}n^{5/4}$.*

Proof. Let $A = \{a_1, \ldots, a_n\}$. Set $X = A + A$ and $Y = A \cdot A$. We will show that $|X||Y| \geqslant \frac{1}{16}n^{5/2}$, which proves the theorem. Let $P = X \times Y \subset \mathbb{R}^2$ be the set of points whose x-coordinate is in X and whose y-coordinate is in Y. Clearly $|P| = |X||Y|$. Next define a set L of lines by $\ell_{ij} = \{(x, y) \in \mathbb{R}^2 : y = a_i(x - a_j)\}$, for $i, j \in \{1, \ldots, n\}$. Clearly $|L| = n^2$.

On the one hand, every line ℓ_{ij} contains at least n points from P because for $x_k = a_j + a_k \in X$ and $y_k = a_i(x_k - a_j) = a_i a_k \in Y$ we have $(x_k, y_k) \in P \cap \ell_{ij}$, for $k \in \{1, \ldots, n\}$. Therefore the number I of incidences betwen P and L is at least n^3.

On the other hand, by the Szemerédi-Trotter Theorem we have $I \leqslant \sqrt[3]{32}|P|^{2/3}n^{4/3} + 4|P| + n^2$. Combining both bounds we obtain

$$n^3 \leqslant \sqrt[3]{32}|P|^{2/3}n^{4/3} + 4|P| + n^2.$$

Hence either $4|P| + n^2 \geqslant \frac{n^3}{2}$, which implies $|P| \geqslant \frac{1}{16}n^{5/2}$, for $n \geqslant 3$; or $\sqrt[3]{32}\,|P|^{2/3}n^{4/3} \geqslant \frac{n^3}{2}$ and thus

$$|P|^{2/3} \geqslant \frac{n^3}{2\sqrt[3]{32}\,n^{4/3}} = \left(\frac{n^5}{256}\right)^{1/3} \Longrightarrow |P| \geqslant \frac{n^{5/2}}{16}.$$

□

Exercise 10.7. *Consider two edges e and f in a topological plane drawing so that e and f cross at least twice. Prove or disprove: There exist always two distinct crossings x and y of e and f so that the portion of e between x and y is not crossed by f and the portion of f between x and y is not crossed by e.*

Exercise 10.8. *Let G be a graph with $n \geqslant 3$ vertices, e edges, and $\mathrm{cr}(G) = e - (3n - 6)$. Show that in every drawing of G with $\mathrm{cr}(G)$ crossings, every edge is crossed at most once.*

Exercise 10.9. *Consider the abstract graph G that is obtained as follows: Start from a plane embedding of the 3-dimensional (hyper-)cube, and add in every face a pair of (crossing) diagonals. Show that $\mathrm{cr}(G) = 6 < \overline{\mathrm{cr}}(G)$.*

Exercise 10.10. *A graph is* 1-planar *if it can be drawn in the plane so that every edge is crossed at most once. Show that a 1-planar graph on $n \geqslant 3$ vertices has at most $4n - 8$ edges.*

Exercise 10.11. *Show that the bound from the Crossing Lemma is asymptotically tight: There exists a constant* c *so that for every* $n, e \in \mathbb{N}$ *with* $e \leqslant \binom{n}{2}$ *there is a graph with* n *vertices and* e *edges that admits a plane drawing with at most* ce^3/n^2 *crossings.*

Exercise 10.12. *Show that the maximum number of unit distances determined by* n *points in* $\mathbb{R}^2$ *is* $\Omega(n \log n)$. *Hint: Consider the hypercube.*

Questions

54. *What is the crossing number of a graph? What is the rectilinear crossing number?* Give the definitions and examples. Explain the difference.
55. *For a nonplanar graph, the more edges it has, the more crossings we would expect. Can you quantify such a correspondence more precisely?* State and prove Lemma 10.2 and Theorem 10.3 (The Crossing Lemma).
56. *Why is it called "Crossing Lemma" rather than "Crossing Theorem"?* Explain at least two applications of the Crossing Lemma, for instance, your pick out of the theorems 10.4, 10.5, and 10.6.

References

[1] Eyal Ackerman, On topological graphs with at most four crossings per edge. *CoRR*, **abs/1509.01932**.

[2] Miklós Ajtai, Václav Chvátal, Monroe M. Newborn, and Endre Szemerédi, Crossing-free subgraphs. *Ann. Discrete Math.*, **12**, (1982), 9–12.

[3] Frank Harary and Anthony Hill, On the number of crossings in a complete graph. *Proc. Edinburgh Math. Soc.*, **13**, 4, (1963), 333–338.

[4] János Pach and Géza Tóth, Graphs drawn with few crossings per edge. *Combinatorica*, **17**, 3, (1997), 427–439.

[5] László A. Székely, Crossing numbers and hard Erdős problems in discrete geometry. *Combinatorics, Probability and Computing*, **6**, 3, (1997), 353–358.

[6] Endre Szemerédi and William T. Trotter, Jr., Extremal problems in discrete geometry. *Combinatorica*, **3**, 3–4, (1983), 381–392.

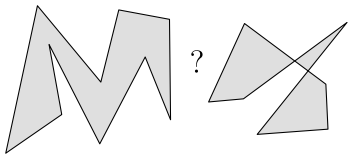
Appendix A

Line Sweep

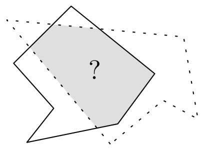
In this chapter we will discuss a simple and widely applicable paradigm to design geometric algorithms: the so-called *Line-Sweep* (or Plane-Sweep) technique. It can be used to solve a variety of different problems, some examples are listed below. The first part may come as a reminder to many of you, because you should have heard something about line-sweep in one of the basic CS courses already. However, we will soon proceed and encounter a couple of additional twists that were most likely not covered there.

Consider the following geometric problems.

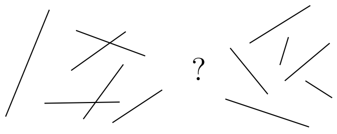
Problem A.1 (Simple Polygon Test). Given a sequence $P = (p_1, \ldots, p_n)$ of points in $\mathbb{R}^2$, does P describe the boundary of a simple polygon?



Problem A.2 (Polygon Intersection). Given two simple polygons P and Q in $\mathbb{R}^2$ as a (counterclockwise) sequence of their vertices, is $P \cap Q = \emptyset$?

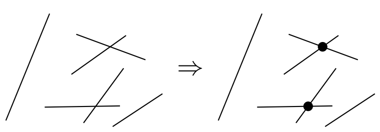


Problem A.3 (Segment Intersection Test). Given a set S of n closed line segments in $\mathbb{R}^2$, do any two of them intersect?

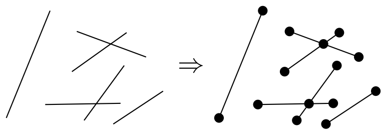


Remark: In principle it is clear what is meant by "two segments intersect". But there are a few special cases that one may have to consider carefully. For instance, does it count if an endpoint lies on another segment? What if two segments share an endpoint? What about overlapping segments and segments of length zero? In general, let us count all these as intersections. However, sometimes we may want to exclude some of these cases. For instance, in a simple polygon test, we do not want to consider the shared endpoint between two consecutive edges of the boundary as an intersection.

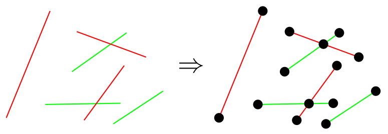
Problem A.4 (Segment Intersections). Given a set S of n closed line segments in $\mathbb{R}^2$, compute all pairs of segments that intersect.



Problem A.5 (Segment Arrangement). Given a set S of n closed line segments in $\mathbb{R}^2$, construct the arrangement induced by S, that is, the subdivision of $\mathbb{R}^2$ induced by S.



Problem A.6 (Map Overlay). Given two sets S and T of n and m, respectively, pairwise interior disjoint line segments in $\mathbb{R}^2$, construct the arrangement induced by $S \cup T$.



In the following we will use Problem A.4 as our flagship example.

Trivial Algorithm. Test all the $\binom{n}{2}$ pairs of segments from S in $O(n^2)$ time and $O(n)$ space. For Problem A.4 this is worst-case optimal because there may by $\Omega(n^2)$ intersecting pairs.

But in case that the number of intersecting pairs is, say, linear in n there is still hope to obtain a subquadratic algorithm. Given that there is a lower bound of $\Omega(n \log n)$ for *Element Uniqueness* (Given $x_1, \ldots, x_n \in \mathbb{R}$, is there an $i \neq j$ such that $x_i = x_j$?) in the algebraic computation tree model, all we can hope for is an output-sensitive runtime of the form $O(n \log n + k)$, where k denotes the number of intersecting pairs (output size).

A.1 Interval Intersections

As a warmup let us consider the corresponding problem in $\mathbb{R}^1$.

Problem A.7. Given a set I of n intervals $[\ell_i, r_i] \subset \mathbb{R}$, $1 \leqslant i \leqslant n$. Compute all pairs of intervals from I that intersect.

Theorem A.8. *Problem A.7 can be solved in* $O(n \log n + k)$ *time and* $O(n)$ *space, where* k *is the number of intersecting pairs from* $\binom{I}{2}$.

Proof. First observe that two real intervals intersect if and only if one contains the right endpoint of the other.

Sort the set $\{(\ell_i, 0) \mid 1 \leqslant i \leqslant n\} \cup \{(r_i, 1) \mid 1 \leqslant i \leqslant n\}$ in increasing lexicographic order and denote the resulting sequence by P. Store along with each point from P its origin (i). Walk through P from start to end while maintaining a list L of intervals that contain the current point $p \in P$.

Whenever $p = (\ell_i, 0)$, $1 \leqslant i \leqslant n$, insert i into L. Whenever $p = (r_i, 1)$, $1 \leqslant i \leqslant n$, remove i from L and then report for all $j \in L$ the pair $\{i, j\}$ as intersecting. □

A.2 Segment Intersections

How can we transfer the (optimal) algorithm for the corresponding problem in $\mathbb{R}^1$ to the plane? In $\mathbb{R}^1$ we moved a point from left to right and at any point resolved the situation locally around this point. More precisely, at any point during the algorithm, we knew all intersections that are to the left of the current (moving) point. A point can be regarded a hyperplane in $\mathbb{R}^1$, and the corresponding object in $\mathbb{R}^2$ is a line.

General idea. Move a line ℓ (so-called *sweep line*) from left to right over the plane, such that at any point during this process all intersections to the left of ℓ have been reported.

Sweep line status. The list of intervals containing the current point corresponds to a list L of segments (sorted by y-coordinate) that intersect the current sweep line ℓ. This list L is called *sweep line status* (SLS). Considering the situation locally around L, it is obvious that only segments that are adjacent in L can intersect each other. This observation allows to reduce the overall number of intersection tests, as we will see. In order to allow for efficient insertion and removal of segments, the SLS is usually implemented as a balanced binary search tree.

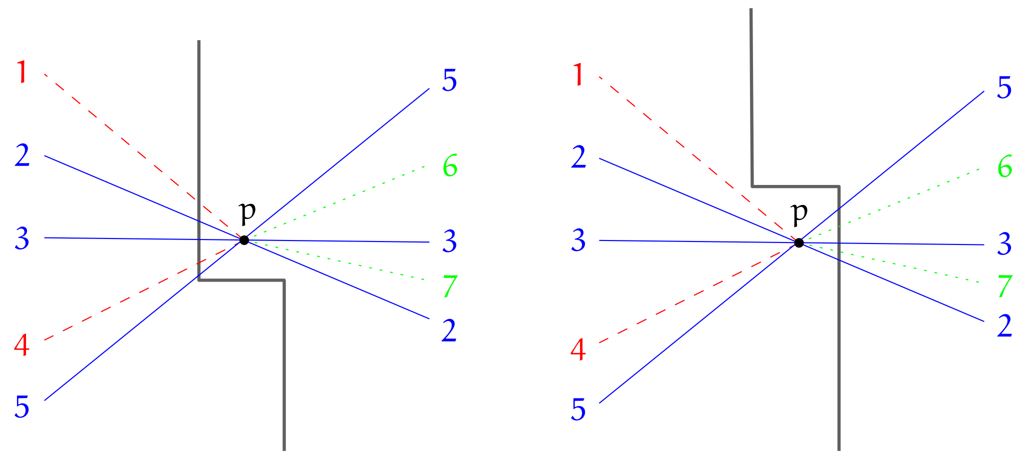
Event points. The order of segments in SLS can change at certain points only: whenever the sweep line moves over a segment endpoint or a point of intersection of two segments from S. Such a point is referred to as an *event point* (EP) of the sweep. Therefore we can reduce the conceptually continuous process of moving the sweep line over the plane to a discrete process that moves the line from EP to EP. This discretization allows for an efficient computation.

At any EP several events can happen simultaneously: several segments can start and/or end and at the same point a couple of other segments can intersect. In fact the sweep line does not even make a difference between any two event points that have the same x-coordinate. To properly resolve the order of processing, EPs are considered in lexicographic order and wherever several events happen at a single point, these are considered simultaneously as a single EP. In this light, the sweep line is actually not a line but an infinitesimal step function (see Figure A.1).

Event point schedule. In contrast to the one-dimensional situation, in the plane not all EP are known in advance because the points of intersection are discovered during the algorithm only. In order to be able to determine the next EP at any time, we use a priority queue data structure, the so-called *event point schedule* (EPS).

Along with every EP p store a list $\text{end}(p)$ of all segments that end at p, a list $\text{begin}(p)$ of all segments that begin at p, and a list $\text{int}(p)$ of all segments in SLS that intersect at p a segment that is adjacent to it in SLS.

Along with every segment we store pointers to all its appearances in an $\text{int}(\cdot)$ list of some EP. As a segment appears in such a list only if it intersects one of its neighbors there, every segment needs to store at most two such pointers.



(a) Before. (b) After.

Figure A.1: *Handling an event point* p*. Ending segments are shown red (dashed), starting segments green (dotted), and passing segments blue (solid).*

Invariants. The following conditions can be shown to hold before and after any event point is handled. (We will not formally prove this here.) In particular, the last condition at the end of the sweep implies the correctness of the algorithm.

1. L is the sequence of segments from S that intersect ℓ, ordered by y-coordinate of their point of intersection.
2. E contains all endpoints of segments from S and all points where two segments that are adjacent in L intersect to the right of ℓ.
3. All pairs from $\binom{S}{2}$ that intersect to the left of ℓ have been reported.

Event point handling. An EP p is processed as follows.

1. If $\text{end}(p) \cup \text{int}(p) = \emptyset$, localize p in L.
2. Report all pairs of segments from $\text{end}(p) \cup \text{begin}(p) \cup \text{int}(p)$ as intersecting.
3. Remove all segments in $\text{end}(p)$ from L.
4. Reverse the subsequence in L that is formed by the segments from $\text{int}(p)$.
5. Insert segments from $\text{begin}(p)$ into L, sorted by slope.
6. Test the topmost and bottommost segment in L from $\text{begin}(p) \cup \text{int}(p)$ for intersection with its successor and predecessor, respectively, and update EP if necessary.

Updating EPS. Insert an EP p corresponding to an intersection of two segments s and t. Without loss of generality let s be above t at the current position of ℓ.

1. If p does not yet appear in E, insert it.

2. If s is contained in an $\mathrm{int}(\cdot)$ list of some other EP q, where it intersects a segment t' from above: Remove both s and t' from the $\mathrm{int}(\cdot)$ list of q and possibly remove q from E (if $\mathrm{end}(q) \cup \mathrm{begin}(q) \cup \mathrm{int}(q) = \emptyset$). Proceed analogously in case that t is contained in an $\mathrm{int}(\cdot)$ list of some other EP, where it intersects a segment from below.

3. Insert s and t into $\mathrm{int}(p)$.

Sweep.

1. Insert all segment endpoints into $\mathrm{begin}(\cdot)/\mathrm{end}(\cdot)$ list of a corresponding EP in E.

2. As long as $E \neq \emptyset$, handle the first EP and then remove it from E.

Runtime analysis. Initialization: $O(n \log n)$. Processing of an EP p:

$$O(\#\text{intersecting pairs} + |\mathrm{end}(p)| \log n + |\mathrm{int}(p)| + |\mathrm{begin}(p)| \log n + \log n).$$

In total: $O(k+n \log n+k \log n) = O((n+k) \log n)$, where k is the number of intersecting pairs in S.

Space analysis. Clearly $|S| \leqslant n$. At begin we have $|E| \leqslant 2n$ and $|S| = 0$. Furthermore the number of additional EPs corresponding to points of intersection is always bounded by $2|S|$. Thus the space needed is $O(n)$.

Theorem A.9. *Problem A.4 and Problem A.5 can be solved in $O((n + k) \log n)$ time and $O(n)$ space.* □

Theorem A.10. *Problem A.1, Problem A.2 and Problem A.3 can be solved in $O(n \log n)$ time and $O(n)$ space.* □

Exercise A.11. *Flesh out the details of the sweep line algorithm for Problem A.2 that is referred to in Theorem A.10. What if you have to construct the intersection rather than just to decide whether or not it is empty?*

Exercise A.12. *You are given n axis–parallel rectangles in $\mathbb{R}^2$ with their bottom sides lying on the x–axis. Construct their union in $O(n \log n)$ time.*

A.3 Improvements

The basic ingredients of the line sweep algorithm go back to work by Bentley and Ottmann [2] from 1979. The particular formulation discussed here, which takes all possible degeneracies into account, is due to Mehlhorn and Näher [9].

Theorem A.10 is obviously optimal for Problem A.3, because this is just the 2-dimensional generalization of Element Uniqueness (see Section 1.1). One might suspect there is also a similar lower bound of $\Omega(n \log n)$ for Problem A.1 and Problem A.2. However, this is not the case: Both can be solved in $O(n)$ time, albeit using a very complicated algorithm of Chazelle [4] (the famous triangulation algorithm).

Similarly, it is not clear why $O(n \log n + k)$ time should not be possible in Theorem A.9. Indeed this was a prominent open problem in the 1980's that has been solved in several steps by Chazelle and Edelsbrunner in 1988. The journal version of their paper [5] consists of 54 pages and the space usage is suboptimal $O(n + k)$.

Clarkson and Shor [6] and independently Mulmuley [10, 11] described randomized algorithms with expected runtime $O(n \log n + k)$ using $O(n)$ and $O(n + k)$, respectively, space.

An optimal deterministic algorithm, with runtime $O(n \log n + k)$ and using $O(n)$ space, is known since 1995 only due to Balaban [1].

A.4 Algebraic degree of geometric primitives

We already have encountered different notions of complexity during this course: We can analyze the time complexity and the space complexity of algorithms and both usually depend on the size of the input. In addition we have seen examples of output-sensitive algorithms, whose complexity also depends on the size of the output. In this section, we will discuss a different kind of complexity that is the algebraic complexity of the underlying geometric primitives. In this way, we try to shed a bit of light into the bottom layer of geometric algorithms that is often swept under the rug because it consists of constant time operations only. Nevertheless, it would be a mistake to disregard this layer completely, because in most—if not every—real world application it plays a crucial role, by affecting efficiency as well as correctness. Regarding efficiency, the value of the constants involved can make a big difference. The possible effects on correctness will hopefully become clear in the course of this section.

In all geometric algorithms there are some fundamental geometric *predicates* and/or *constructions* on the bottom level. Both are operations on geometric objects, the difference is only in the result: The result of a construction is a geometric object (for instance, the point common to two non-parallel lines), whereas the result of a predicate is Boolean (*true* or *false*).

Geometric predicates and constructions in turn are based on fundamental arithmetic operations. For instance, we formulated planar convex hull algorithms in terms of an orientation predicate—given three points $p, q, r \in \mathbb{R}^2$, is r strictly to the right of the oriented line pr— which can be implemented using multiplication and addition/subtraction of coordinates/numbers. When using limited precision arithmetic, it is important to keep an eye on the size of the expressions that occur during an evaluation of such a predicate.

In Exercise 4.27 we have seen that the orientation predicate can be computed by evaluating a polynomial of degree two in the input coordinates and, therefore, we say that

Proposition A.13. *The rightturn/orientation predicate for three points in $\mathbb{R}^2$ has algebraic degree two.*

The degree of a predicate depends on the algebraic expression used to evaluate it. Any such expression is intimately tied to the representation of the geometric objects used. Where not stated otherwise, we assume that geometric objects are represented as described in Section 1.2. So in the proposition above we assume that points are represented using Cartesian coordinates. The situation might be different, if, say, a polar representation is used instead.

But even once a representation is agreed upon, there is no obvious correspondence to an algebraic expression that describes a given predicate. In fact, it is not even clear why such an expression should exist in general. But if it does—as, for instance, in case of the orientation predicate—we prefer to work with a polynomial of smallest possible degree. Therefore we define the *(algebraic) degree* of a geometric predicate as the minimum degree of a polynomial that defines it (predicate true $\iff$ polynomial positive). So there still is something to be shown in order to complete Proposition A.13.

Exercise A.14. *Show that there is no polynomial of degree at most one that describes the orientation test in $\mathbb{R}^2$.*

Hint: Let $p = (0, 0)$, $q = (x, 0)$, and $r = (0, y)$ and show that there is no linear function in x and y that distinguishes the region where pqr form a rightturn from its complement.

The degree of a predicate corresponds to the size of numbers that arise during its evaluation. If all input coordinates are k-bit integers then the numbers that occur during an evaluation of a degree d predicate on these coordinates are of size about[1] dk. If the number type used for the computation can represent all such numbers, the predicate can be evaluated exactly and thus always correctly. For instance, when using a standard IEEE double precision floating point implementation which has a mantissa length of 53 bit then the above orientation predicate can be evaluated exactly if the input coordinates are integers between 0 and 2^{25}, say.

Let us now get back to the line segment intersection problem. It needs a few new geometric primitives: most prominently, constructing the intersection point of two line segments.

[1]It is only *about* dk because not only multiplications play a role but also additions. As a rule of thumb, a multiplication may double the bitsize, while an addition may increase it by one.

Two segments. Given two line segments $s = \lambda a + (1-\lambda)b, \lambda \in [0,1]$ and $t = \mu c + (1-\mu)d, \mu \in [0,1]$, it is a simple exercise in linear algebra to compute $s \cap t$. Note that $s \cap t$ is either empty or a single point or a line segment.

For $a = (a_x, a_y)$, $b = (b_x, b_y)$, $c = (c_x, c_y)$, and $d = (d_x, d_y)$ we obtain two linear equations in two variables λ and μ.

$$\begin{aligned} \lambda a_x + (1-\lambda)b_x &= \mu c_x + (1-\mu)d_x \\ \lambda a_y + (1-\lambda)b_y &= \mu c_y + (1-\mu)d_y \end{aligned}$$

Rearranging terms yields

$$\begin{aligned} \lambda(a_x - b_x) + \mu(d_x - c_x) &= d_x - b_x \\ \lambda(a_y - b_y) + \mu(d_y - c_y) &= d_y - b_y \end{aligned}$$

Assuming that the lines underlying s and t have distinct slopes (that is, they are neither identical nor parallel) we have

$$D = \begin{vmatrix} a_x - b_x & d_x - c_x \\ a_y - b_y & d_y - c_y \end{vmatrix} = \begin{vmatrix} a_x & a_y & 1 & 0 \\ b_x & b_y & 1 & 0 \\ c_x & c_y & 0 & 1 \\ d_x & d_y & 0 & 1 \end{vmatrix} \neq 0$$

and using Cramer's rule

$$\lambda = \frac{1}{D}\begin{vmatrix} d_x - b_x & d_x - c_x \\ d_y - b_y & d_y - c_y \end{vmatrix} \text{ and } \mu = \frac{1}{D}\begin{vmatrix} a_x - b_x & d_x - b_x \\ a_y - b_y & d_y - b_y \end{vmatrix}.$$

To test if s and t intersect, we can—after having sorted out the degenerate case in which both segments have the same slope—compute λ and μ and then check whether $\lambda, \mu \in [0,1]$.

Observe that both λ and D result from multiplying two differences of input coordinates. Computing the x-coordinate of the point of intersection via $b_x + \lambda(a_x - b_x)$ uses another multiplication. Overall this computation yields a fraction whose numerator is a polynomial of degree three in the input coordinates and whose denominator is a polynomial of degree two in the input coordinates.

In order to maintain the sorted order of event points in the EPS, we need to compare event points lexicographically. In case that both are intersection points, this amounts to comparing two fractions of the type discussed above. In order to formulate such a comparison as a polynomial, we have to cross-multiply the denominators, and so obtain a polynomial of degree $3 + 2 = 5$. It can be shown (but we will not do it here) that this bound is tight and so we conclude that

Proposition A.15. *The algebraic degree of the predicate that compares two intersection points of line segments lexicographically is five.*

Therefore the coordinate range in which this predicate can be evaluated exactly using IEEE double precision numbers shrinks down to integers between 0 and about $2^{10} = 1'024$.

Exercise A.16. *What is the algebraic degree of the predicate checking whether two line segments intersect? (Above we were interested in the actual intersection point, but now we consider the predicate that merely answers the question whether two segments intersect or not by yes or no).*

A.5 Red-Blue Intersections

Although the Bentley-Ottmann sweep appears to be rather simple, its implementation is not straightforward. For once, the original formulation did not take care of possible degeneracies—as we did in the preceding section. But also the algebraic degree of the predicates used is comparatively high. In particular, comparing two points of intersection lexicographically is a predicate of degree five, that is, in order to compute it, we need to evaluate a polynomial of degree five. When evaluating such a predicate with plain floating point arithmetic, one easily gets incorrect results due to limited precision roundoff errors. Such failures frequently render the whole computation useless. The line sweep algorithm is problematic in this respect, as a failure to detect one single point of intersection often implies that no other intersection of the involved segments to the right is found.

In general, predicates of degree four are needed to construct the arrangement of line segments because one needs to determine the orientation of a triangle formed by three segments. This is basically an orientation test where two points are segment endpoints and the third point is an intersection point of segments. Given that the coordinates of the latter are fractions, whose numerator is a degree three polynomial and the common denominator is a degree two polynomial, we obtain a degree four polynomial overall.

Motivated by the Map overlay application we consider here a restricted case. In the *red-blue intersection* problem, the input consists of two sets R (red) and B (blue) of segments such that the segments in each set are *interior-disjoint*, that is, for any pair of distinct segments their relative interior is disjoint. In this case there are no triangles of intersecting segments, and it turns out that predicates of degree two suffice to construct the arrangement. This is optimal because already the intersection test for two segments is a predicate of degree two.

Predicates of degree two. Restricting to degree two predicates has certain consequences. While it is possible to determine the position of a segment endpoint relative to a(nother) segment using an orientation test, one cannot, for example, compare a segment endpoint with a point of intersection lexicographically. Even computing the coordinates for a point of intersection is not possible. Therefore the output of intersection points is done implicitly, as "intersection of s and t".

Graphically speaking we can deform any segment—keeping its endpoints fixed—as long as it remains monotone and it does not reach or cross any segment endpoint (it did not touch before). With help of degree two predicates there is no way to tell the difference.

Witnesses. Using such transformations the processing of intersection points is deferred as long as possible (lazy computation). The last possible point $w(s,t)$ where an intersection between two segments s and t has to be processed we call the *witness* of the intersection. The witness $w(s,t)$ is the lexicographically smallest segment endpoint that is located within the closed wedge formed by the two intersecting segments s and t to the right of the point of intersection (Figure A.2). Note that each such wedge contains at least two segment endpoints, namely the right endpoints of the two intersecting segments. Therefore for any pair of intersecting segments its witness is well-defined.

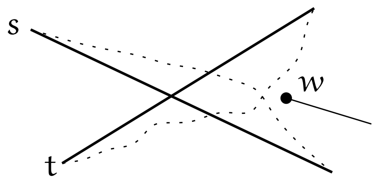


Figure A.2: *The witness* $w(s,t)$ *of an intersection* $s \cap t$.

As a consequence, only the segment endpoints are EPs and the EPS can be determined by lexicographic sorting during initialization. On the other hand, the SLS structure gets more complicated because its order of segments does not necessarily reflect the order in which the segments intersect the sweep line.

Invariants. The invariants of the algorithm have to take the relaxed notion of order into account. Denote the sweep line by ℓ.

1. L is the sequence of segments from $S = R \cup B$ intersecting ℓ; s appears before t in $L \implies s$ intersects ℓ above t or s intersects t and the witness of this intersection is to the right of ℓ.

2. All intersections of segments from S whose witness is to the left of ℓ have been reported.

SLS Data Structure. The SLS structure consist of three levels. We use the fact that segments of the same color do not interact, except by possibly sharing endpoints.

1. Collect adjacent segments of the same color in *bundles*, stored as balanced search trees. For each bundle store pointers to the topmost and bottommost segment. (As the segments within one bundle are interior-disjoint, their order remains static and thus correct under possible deformations due to lazy computation.)

2. All bundles are stored in a doubly linked list, sorted by y-coordinate.

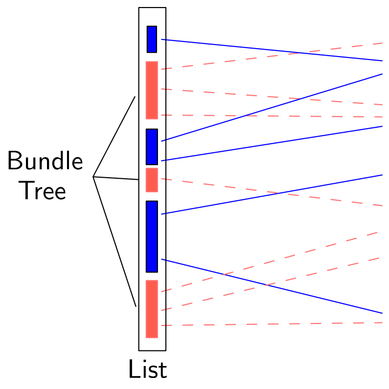


Figure A.3: *Graphical representation of the SLS data structure.*

3. All red bundles are stored in a balanced search tree (*bundle tree*).

The search tree structure should support insert, delete, split and merge in (amortized) logarithmic time each. For instance, splay trees [12] meet these requirements. (If you have never heard about splay trees so far, you do not need to know for our purposes here. Just treat them as a black box. However, you should take a note and read about splay trees still. They are a fascinating data structure.)

EP handling. An EP p is processed as follows.

1. We first want to classify all bundles with respect to their position relative to p: as either lying *above* or *below* p or as *ending* at p. There are $\leqslant 2$ bundles that have no clear such characterization.

   (a) Localize p in bundle tree $\rightarrow$ Find $\leqslant 2$ bundles without clear characterization. For the localization we use the pointers to the topmost and bottommost segment within a bundle.

   (b) Localize p in $\leqslant 2$ red bundles found and split them at p. (If p is above the topmost or below the bottommost segment of the bundle, there is no split.) All red bundles are now either above, ending, or below with respect to p.

   (c) Localize p within the blue bundles by linear search.

   (d) Localize p in the $\leqslant 2$ blue bundles found and split them at p. (If p is above the topmost or below the bottommost segment of the bundle, there is no split.) All bundles are now either above, ending, or below with respect to p.

2. Run through the list of bundles around p. More precisely, start from one of the bundles containing p found in Step 1 and walk up in the doubly linked list of bundles until two successive bundles (hence of opposite color) are both above p. Similarly, walk down in the doubly linked list of bundles, until two successive bundles are both below p. Handle all adjacent pairs of bundles (A, B) that are in

wrong order and report all pairs of segments as intersecting. (Exchange A and B in the bundle list and merge them with their new neighbors.)

3. Report all two-colored pairs from $\text{begin}(p) \times \text{end}(p)$ as intersecting.

4. Remove ending bundles and insert starting segments, sorted by slope and bundled by color and possibly merge with the closest bundle above or below.

Remark: As both the red and the blue segments are interior-disjoint, at every EP there can be at most one segment that passes through the EP. Should this happen, for the purpose of EP processing split this segment into two, one ending and one starting at this EP. But make sure to not report an intersection between the two parts!

Analysis. Sorting the EPS: $O(n \log n)$ time. Every EP generates a constant number of tree searches and splits of $O(\log n)$ each. Every exchange in Step 2 generates at least one intersection. New bundles are created only by inserting a new segment or by one of the constant number of splits at an EP. Therefore $O(n)$ bundles are created in total. The total number of merge operations is $O(n)$, as every merge kills one bundle and $O(n)$ bundles are created overall. The linear search in steps 1 and 2 can be charged either to the ending bundle or—for continuing bundles—to the subsequent merge operation. In summary, we have a runtime of $O(n \log n + k)$ and space usage is linear obviously.

Theorem A.17. *For two sets R and B, each consisting of interior-disjoint line segments in $\mathbb{R}^2$, one can find all intersecting pairs of segments in $O(n \log n + k)$ time and linear space, using predicates of maximum degree two. Here $n = |R| + |B|$ and k is the number of intersecting pairs.*

Remarks. The first optimal algorithm for the red-blue intersection problem was published in 1988 by Harry Mairson and Jorge Stolfi [7]. In 1994 Timothy Chan [3] described a trapezoidal-sweep algorithm that uses predicates of degree three only. The approach discussed above is due to Andrea Mantler and Jack Snoeyink [8] from 2000.

Exercise A.18. *Let S be a set of n segments each of which is either horizontal or vertical. Describe an $O(n \log n)$ time and $O(n)$ space algorithm that counts the number of pairs in $\binom{S}{2}$ that intersect.*

Questions

57. *How can one test whether a polygon on n vertices is simple?* Describe an $O(n \log n)$ time algorithm.

58. *How can one test whether two simple polygons on altogether n vertices intersect?* Describe an $O(n \log n)$ time algorithm.

59. *How does the line sweep algorithm work that finds all k intersecting pairs among n line segments in $\mathbb{R}^2$?* Describe the algorithm, using $O((n+k)\log n)$ time and $O(n)$ space. In particular, explain the data structures used, how event points are handled, and how to cope with degeneracies.

60. *Given two line segments s and t in $\mathbb{R}^2$ whose endpoints have integer coordinates in $[0, 2^b)$; suppose s and t intersect in a single point q, what can you say about the coordinates of q?* Give a good (tight up to small additive number of bits) upper bound on the size of these coordinates. (No need to prove tightness, that is, give an example which achieves the bound.)

61. *What is the degree of a predicate and why is it an important parameter? What is the degree of the orientation test and the incircle test in $\mathbb{R}^2$?* Explain the term and give two reasons for its relevance. Provide tight upper bounds for the two predicates mentioned. (No need to prove tightness.)

62. *What is the map overlay problem and how can it be solved optimally using degree two predicates only?* Give the problem definition and explain the term "interior-disjoint". Explain the bundle-sweep algorithm, in particular, the data structures used, how an event point is processed, and the concept of witnesses.

References

[1] Ivan J. Balaban, An optimal algorithm for finding segment intersections. In *Proc. 11th Annu. ACM Sympos. Comput. Geom.*, pp. 211–219, 1995.

[2] Jon L. Bentley and Thomas A. Ottmann, Algorithms for reporting and counting geometric intersections. *IEEE Trans. Comput.*, **C-28**, 9, (1979), 643–647.

[3] Timothy M. Chan, A simple trapezoid sweep algorithm for reporting red/blue segment intersections. In *Proc. 6th Canad. Conf. Comput. Geom.*, pp. 263–268, 1994.

[4] Bernard Chazelle, Triangulating a simple polygon in linear time. *Discrete Comput. Geom.*, **6**, 5, (1991), 485–524.

[5] Bernard Chazelle and Herbert Edelsbrunner, An optimal algorithm for intersecting line segments in the plane. *J. ACM*, **39**, 1, (1992), 1–54.

[6] Kenneth L. Clarkson and Peter W. Shor, Applications of random sampling in computational geometry, II. *Discrete Comput. Geom.*, **4**, (1989), 387–421.

[7] Harry G. Mairson and Jorge Stolfi, Reporting and counting intersections between two sets of line segments. In R. A. Earnshaw, ed., *Theoretical Foundations of Computer Graphics and CAD*, vol. 40 of *NATO ASI Series F*, pp. 307–325, Springer-Verlag, Berlin, Germany, 1988.

[8] Andrea Mantler and Jack Snoeyink, Intersecting red and blue line segments in optimal time and precision. In J. Akiyama, M. Kano, and M. Urabe, eds., *Proc. Japan Conf. Discrete Comput. Geom.*, vol. 2098 of *Lecture Notes Comput. Sci.*, pp. 244–251, Springer Verlag, 2001.

[9] Kurt Mehlhorn and Stefan Näher, Implementation of a sweep line algorithm for the straight line segment intersection problem. Report MPI-I-94-160, Max-Planck-Institut Inform., Saarbrücken, Germany, 1994.

[10] Ketan Mulmuley, A fast planar partition algorithm, I. *J. Symbolic Comput.*, **10**, 3-4, (1990), 253–280.

[11] Ketan Mulmuley, A fast planar partition algorithm, II. *J. ACM*, **38**, (1991), 74–103.

[12] Daniel D. Sleator and Robert E. Tarjan, Self-adjusting binary search trees. *J. ACM*, **32**, 3, (1985), 652–686.

Appendix B

The Configuration Space Framework

In Section 6.1, we have discussed the incremental construction of the Delaunay triangulation of a finite point set. In this lecture, we want to analyze the runtime of this algorithm if the insertion order is chosen *uniformly at random* among all insertion orders. We will do the analysis not directly for the problem of constructing the Delaunay triangulation but in a somewhat more abstract framework, with the goal of reusing the analysis for other problems.

Throughout this lecture, we again assume general position: no three points on a line, no four on a circle.

B.1 The Delaunay triangulation — an abstract view

The incremental construction constructs and destroys triangles. In this section, we want to take a closer look at these triangles, and we want to understand exactly when a triangle is "there".

Lemma B.1. *Given three points* $p, q, r \in R$*, the triangle* $\Delta(p, q, r)$ *with vertices* p, q, r *is a triangle of* $\mathcal{DT}(R)$ *if and only if the circumcircle of* $\Delta(p, q, r)$ *is empty of points from* R*.*

Proof. The "only if" direction follows from the definition of a Delaunay triangulation (Definition 5.8). The "if" direction is a consequence of general position and Lemma 5.18: if the circumcircle C of $\Delta(p, q, r)$ is empty of points from R, then all the three edges $\overline{pq}, \overline{qr}, \overline{pr}$ are easily seen to be in the Delaunay graph of R. C being empty also implies that the triangle $\Delta(p, q, r)$ is empty, and hence it forms a triangle of $\mathcal{DT}(R)$. □

Next we develop a somewhat more abstract view of $\mathcal{DT}(R)$.

Definition B.2.

(i) For all $p, q, r \in P$*, the triangle* $\Delta = \Delta(p, q, r)$ *is called a* configuration*. The points* p, q *and* r *are called the* defining elements *of* Δ*.*

(ii) A configuration Δ is in conflict *with a point $s \in P$ if s is strictly inside the circumcircle of Δ. In this case, the pair (Δ, s) is called a* conflict.

(iii) A configuration Δ is called active *w.r.t. $R \subseteq P$ if (a) the defining elements of Δ are in R, and (b) if Δ is not in conflict with any element of R.*

According to this definition and Lemma B.1, $\mathcal{DT}(R)$ consists of exactly the configurations that are active w.r.t. R. Moreover, if we consider $\mathcal{DT}(R)$ and $\mathcal{DT}(R \cup \{s\})$ as sets of configurations, we can exactly say how these two sets differ.

There are the configurations in $\mathcal{DT}(R)$ that are not in conflict with s. These configurations are still in $\mathcal{DT}(R \cup \{s\})$. The configurations of $\mathcal{DT}(R)$ that *are* in conflict with s will be removed when going from R to $R \cup \{s\}$. Finally, $\mathcal{DT}(R \cup \{s\})$ contains some new configurations, all of which must have s in their defining set. According to Lemma B.1, it cannot happen that we get a new configuration without s in its defining set, as such a configuration would have been present in $\mathcal{DT}(R)$ already.

B.2 Configuration Spaces

Here is the abstract framework that generalizes the previous configuration view of the Delaunay triangulation.

Definition B.3. *Let X (the ground set) and Π (the set of configurations) be finite sets. Furthermore, let*

$$D : \Pi \to 2^X$$

be a function that assigns to every configuration Δ a set of defining elements $D(\Delta)$. *We assume that only a constant number of configurations have the same defining elements. Let*

$$K : \Pi \to 2^X$$

be a function that assigns to every configuration Δ a set of elements in conflict *with Δ (the "killer" elements). We stipulate that $D(\Delta) \cap K(\Delta) = \emptyset$ for all $\Delta \in \Pi$.*

Then the quadruple $\mathcal{S} = (X, \Pi, D, K)$ is called a configuration space. *The number*

$$d = d(\mathcal{S}) := \max_{\Delta \in \Pi} |D(\Delta)|$$

is called the dimension *of $\mathcal{S}$.*

Given $R \subseteq X$, a configuration Δ is called active *w.r.t. R if*

$$D(\Delta) \subseteq R \quad \textit{and} \quad K(\Delta) \cap R = \emptyset,$$

i.e. if all defining elements are in R but no element of R is in conflict with Δ. The set of active configurations w.r.t. R is denoted by $\mathcal{T}_{\mathcal{S}}(R)$, where we drop the subscript if the configuration space is clear from the context.

In case of the Delaunay triangulation, we set $X = P$ (the input point set). Π consists of all triangles $\Delta = \Delta(p, q, r)$ spanned by three points $p, q, r \in X \cup \{a, b, c\}$, where a, b, c are the three artificial far-away points. We set $D(\Delta) := \{p, q, r\} \cap X$. The set $K(\Delta)$ consists of all points strictly inside the circumcircle of Δ. The resulting configuration space has dimension 3, and the technical condition that only a constant number of configurations share the defining set is satisfied as well. In fact, every set of three points defines a *unique* configuration (triangle) in this case. A set of two points or one point defines three triangles (we have to add one or two artificial points which can be done in three ways). The empty set defines one triangle, the initial triangle consisting of just the three artificial points.

Furthermore, in the setting of the Delaunay triangulation, a configuration is active w.r.t. R if it is in $\mathcal{DT}(R \cup \{a, b, c\})$, i.e. we have $\mathcal{T}(R) = \mathcal{DT}(R \cup \{a, b, c\})$.

B.3 Expected structural change

Let us fix a configuration space $\mathcal{S} = (X, \Pi, D, K)$ for the remainder of this lecture. We can also interpret the incremental construction in $\mathcal{S}$. Given $R \subseteq X$ and $s \in X \setminus R$, we want to update $\mathcal{T}(R)$ to $\mathcal{T}(R \cup \{s\})$. What is the number of new configurations that arise during this step? For the case of Delaunay triangulations, this is the relevant question when we want to bound the number of Lawson flips during one update step, since this number is exactly the number of new configurations minus three.

Here is the general picture.

Definition B.4. *For $Q \subseteq X$ and $s \in Q$, $\deg(s, Q)$ is defined as the number of configurations of $\mathcal{T}(Q)$ that have s in their defining set.*

With this, we can say that the number of new configurations in going from $\mathcal{T}(R)$ to $\mathcal{T}(R \cup \{s\})$ is precisely $\deg(s, R \cup \{s\})$, since the new configurations are by definition exactly the ones that have s in their defining set.

Now the random insertion order comes in for the first time: what is

$$E(\deg(s, R \cup \{s\})),$$

averaged over all insertion orders? In such a random insertion order, R is a random r-element subset of X (when we are about to insert the $(r+1)$-st element), and s is a random element of $X \setminus R$. Let $\mathcal{T}_r$ be the "random variable" for the set of active configurations after r insertion steps.

It seems hard to average over all R, but there is a trick: we make a movie of the randomized incremental construction, and then we watch the movie backwards. What we see is elements of X being deleted one after another, again in random order. This is due to the fact that the reverse of a random order is also random. At the point where the $(r+1)$-st element is being deleted, it is going to be a *random* element s of the currently present $(r+1)$-element subset Q. For fixed Q, the expected degree of s is simply the average degree of an element in Q which is

$$\frac{1}{r+1}\sum_{s\in Q}\deg(s,Q) \leqslant \frac{d}{r+1}|\mathcal{T}(Q)|,$$

since the sum counts every configuration of $\mathcal{T}(Q)$ at most d times. Since Q is a random $(r+1)$-element subset, we get

$$E(\deg(s,R\cup\{s\})) \leqslant \frac{d}{r+1}t_{r+1},$$

where t_{r+1} is defined as the expected number of active configurations w.r.t. a random $(r+1)$-element set.

Here is a more formal derivation that does not use the backwards movie view. It exploits the bijection

$$(R,s)\mapsto(\underbrace{R\cup\{s\}}_{Q},s)$$

between pairs (R,s) with $|R|=r$ and $s\notin R$ and pairs (Q,s) with $|Q|=r+1$ and $s\in Q$. Let $n=|X|$.

$$\begin{aligned}
E(\deg(s,R\cup\{s\})) &= \frac{1}{\binom{n}{r}}\sum_{R\subseteq X,|R|=r}\frac{1}{n-r}\sum_{s\in X\setminus R}\deg(s,R\cup\{s\})\\
&= \frac{1}{\binom{n}{r}}\sum_{Q\subseteq X,|Q|=r+1}\frac{1}{n-r}\sum_{s\in Q}\deg(s,Q)\\
&= \frac{1}{\binom{n}{r+1}}\sum_{Q\subseteq X,|Q|=r+1}\frac{\binom{n}{r+1}}{\binom{n}{r}}\frac{1}{n-r}\sum_{s\in Q}\deg(s,Q)\\
&= \frac{1}{\binom{n}{r+1}}\sum_{Q\subseteq X,|Q|=r+1}\frac{1}{r+1}\sum_{s\in Q}\deg(s,Q)\\
&\leqslant \frac{1}{\binom{n}{r+1}}\sum_{Q\subseteq X,|Q|=r+1}\frac{d}{r+1}|\mathcal{T}(Q)|\\
&= \frac{d}{r+1}t_{r+1}.
\end{aligned}$$

Thus, the expected number of new active configurations in going from $\mathcal{T}_r$ to $\mathcal{T}_{r+1}$ is bounded by

$$\frac{d}{r+1}t_{r+1},$$

where t_{r+1} is the expected size of $\mathcal{T}_{r+1}$.

What do we get for Delaunay triangulations? We have $d = 3$ and $t_{r+1} \leqslant 2(r+4) - 4$ (the maximum number of triangles in a triangulation of $r + 4$ points). Hence,

$$E(\deg(s, R \cup \{s\})) \leqslant \frac{6r + 12}{r + 1} \approx 6.$$

This means that on average, ≈ 3 Lawson flips are done to update $\mathcal{DT}_r$ (the Delaunay triangulation after r insertion steps) to $\mathcal{DT}_{r+1}$. Over the whole algorithm, the expected update cost is thus $O(n)$.

B.4 Bounding location costs by conflict counting

Before we can even update $\mathcal{DT}_r$ to $\mathcal{DT}_{r+1}$ during the incremental construction of the Delaunay triangulation, we need to locate the new point s in $\mathcal{DT}_r$, meaning that we need to find the triangle that contains s. We have done this with the history graph: During the insertion of s we "visit" a sequence of triangles from the history graph, each of which contains s and was created at some previous iteration $k < r$.

However, some of these visited triangles are "ephemeral" triangles (recall the discussion at the end of Section 6.2), and they present a problem to the generic analysis we want to perform. Therefore, we will do a charging scheme, so that all triangles charged are valid Delaunay triangles.

The charging scheme is as follows: If the visited triangle Δ is a valid Delaunay triangle (from some previous iteration), then we simply charge the visit of Δ during the insertion of s to the triangle-point pair (Δ, s).

If, on the other hand, Δ is an "ephemeral" triangle, then Δ was destroyed, together with some neighbor Δ', by a Lawson flip into another pair Δ'', Δ'''. Note that this neighbor Δ' was a valid triangle. Thus, in this case we charge the visit of Δ during the insertion of s to the pair (Δ', s). Observe that s is contained in the circumcircle of Δ', so s is in conflict with Δ'.

This way, we have charged each visit to a triangle in the history graph to a triangle-point pair of the form (Δ, s), such that Δ is in conflict with s. Furthermore, it is easy to see that no such pair gets charged more than once.

We define the notion of a *conflict* in general:

Definition B.5. *A* conflict *is a configuration-element pair* (Δ, s) *where* $\Delta \in \mathcal{T}_r$ *for some* r *and* $s \in K(\Delta)$.

Thus, the running time of the Delaunay algorithm is proportional to the number of conflicts. We now proceed to derive a bound on the expected number of conflicts in the generic configuration-space framework.

B.5 Expected number of conflicts

Since every configuration involved in a conflict has been created in some step r (we include step 0), the total number of conflicts is

$$\sum_{r=0}^{n} \sum_{\Delta \in \mathcal{T}_r \setminus \mathcal{T}_{r-1}} |K(\Delta)|,$$

where $\mathcal{T}_{-1} := \emptyset$. $\mathcal{T}_0$ consists of constantly many configurations only (namely those where the set of defining elements is the empty set), each of which is in conflict with at most all elements; moreover, no conflict is created in step n. Hence,

$$\sum_{r=0}^{n} \sum_{\Delta \in \mathcal{T}_r \setminus \mathcal{T}_{r-1}} |K(\Delta)| = O(n) + \sum_{r=1}^{n-1} \sum_{\Delta \in \mathcal{T}_r \setminus \mathcal{T}_{r-1}} |K(\Delta)|,$$

and we will bound the latter quantity. Let

$$K(r) := \sum_{\Delta \in \mathcal{T}_r \setminus \mathcal{T}_{r-1}} |K(\Delta)|, \quad r = 1, \ldots, n-1.$$

and $k(r) := E(K(r))$ the expected number of conflicts created in step r.

Bounding $k(r)$. We know that $\mathcal{T}_r$ arises from a random r-element set R. Fixing R, the backwards movie view tells us that $\mathcal{T}_{r-1}$ arises from $\mathcal{T}_r$ by deleting a random element s of R. Thus,

$$\begin{aligned}
k(r) &= \frac{1}{\binom{n}{r}} \sum_{R \subseteq X, |R|=r} \frac{1}{r} \sum_{s \in R} \sum_{\Delta \in \mathcal{T}(R) \setminus \mathcal{T}(R \setminus \{s\})} |K(\Delta)| \\
&= \frac{1}{\binom{n}{r}} \sum_{R \subseteq X, |R|=r} \frac{1}{r} \sum_{s \in R} \sum_{\Delta \in \mathcal{T}(R), s \in D(\Delta)} |K(\Delta)| \\
&\leqslant \frac{1}{\binom{n}{r}} \sum_{R \subseteq X, |R|=r} \frac{d}{r} \sum_{\Delta \in \mathcal{T}(R)} |K(\Delta)|,
\end{aligned}$$

since in the sum over $s \in R$, every configuration is counted at most d times. Since we can rewrite

$$\sum_{\Delta \in \mathcal{T}(R)} |K(\Delta)| = \sum_{y \in X \setminus R} |\{\Delta \in \mathcal{T}(R) : y \in K(\Delta)\}|,$$

we thus have

$$k(r) \leqslant \frac{1}{\binom{n}{r}} \sum_{R \subseteq X, |R|=r} \frac{d}{r} \sum_{y \in X \setminus R} |\{\Delta \in \mathcal{T}(R) : y \in K(\Delta)\}|.$$

To estimate this further, here is a simple but crucial

Lemma B.6. *The configurations in* $\mathcal{T}(R)$ *that are not in conflict with* $y \in X \setminus R$ *are the configurations in* $\mathcal{T}(R \cup \{y\})$ *that do not have* y *in their defining set; in formulas:*

$$|\mathcal{T}(R)| - |\{\Delta \in \mathcal{T}(R) : y \in K(\Delta)\}| = |\mathcal{T}(R \cup \{y\})| - \deg(y, R \cup \{y\}).$$

The proof is a direct consequence of the definitions: every configuration in $\mathcal{T}(R)$ not in conflict with y is by definition still present in $\mathcal{T}(R \cup \{y\})$ and still does not have y in its defining set. And a configuration in $\mathcal{T}(R \cup \{y\})$ with y not in its defining set is by definition already present in $\mathcal{T}(R)$ and already there not in conflict with y.

The lemma implies that

$$k(r) \leqslant k_1(r) - k_2(r) + k_3(r),$$

where

$$k_1(r) = \frac{1}{\binom{n}{r}} \sum_{R \subseteq X, |R|=r} \frac{d}{r} \sum_{y \in X \setminus R} |\mathcal{T}(R)|,$$

$$k_2(r) = \frac{1}{\binom{n}{r}} \sum_{R \subseteq X, |R|=r} \frac{d}{r} \sum_{y \in X \setminus R} |\mathcal{T}(R \cup \{y\})|,$$

$$k_3(r) = \frac{1}{\binom{n}{r}} \sum_{R \subseteq X, |R|=r} \frac{d}{r} \sum_{y \in X \setminus R} \deg(y, R \cup \{y\}).$$

Estimating $k_1(r)$. This is really simple.

$$\begin{aligned} k_1(r) &= \frac{1}{\binom{n}{r}} \sum_{R \subseteq X, |R|=r} \frac{d}{r} \sum_{y \in X \setminus R} |\mathcal{T}(R)| \\ &= \frac{1}{\binom{n}{r}} \sum_{R \subseteq X, |R|=r} \frac{d}{r}(n-r)|\mathcal{T}(R)| \\ &= \frac{d}{r}(n-r)t_r. \end{aligned}$$

Estimating $k_2(r)$. For this, we need to employ our earlier $(R, y) \mapsto (R \cup \{y\}, y)$ bijection again.

$$\begin{aligned}
k_2(r) &= \frac{1}{\binom{n}{r}} \sum_{R\subseteq X, |R|=r} \frac{d}{r} \sum_{y\in X\setminus R} |\mathcal{T}(R\cup\{y\})| \\
&= \frac{1}{\binom{n}{r}} \sum_{Q\subseteq X, |Q|=r+1} \frac{d}{r} \sum_{y\in Q} |\mathcal{T}(Q)| \\
&= \frac{1}{\binom{n}{r+1}} \sum_{Q\subseteq X, |Q|=r+1} \frac{\binom{n}{r+1}}{\binom{n}{r}} \frac{d}{r}(r+1)|\mathcal{T}(Q)| \\
&= \frac{1}{\binom{n}{r+1}} \sum_{Q\subseteq X, |Q|=r+1} \frac{d}{r}(n-r)|\mathcal{T}(Q)| \\
&= \frac{d}{r}(n-r)t_{r+1} \\
&= \frac{d}{r+1}(n-(r+1))t_{r+1} + \frac{dn}{r(r+1)}t_{r+1} \\
&= k_1(r+1) + \frac{dn}{r(r+1)}t_{r+1}.
\end{aligned}$$

Estimating $k_3(r)$. This is similar to $k_2(r)$ and in addition uses a fact that we have employed before: $\sum_{y\in Q} \deg(y, Q) \leqslant d|\mathcal{T}(Q)|$.

$$\begin{aligned}
k_3(r) &= \frac{1}{\binom{n}{r}} \sum_{R\subseteq X, |R|=r} \frac{d}{r} \sum_{y\in X\setminus R} \deg(y, R\cup\{y\}) \\
&= \frac{1}{\binom{n}{r}} \sum_{Q\subseteq X, |Q|=r+1} \frac{d}{r} \sum_{y\in Q} \deg(y, Q) \\
&\leqslant \frac{1}{\binom{n}{r}} \sum_{Q\subseteq X, |Q|=r+1} \frac{d^2}{r}|\mathcal{T}(Q)| \\
&= \frac{1}{\binom{n}{r+1}} \sum_{Q\subseteq X, |Q|=r+1} \frac{\binom{n}{r+1}}{\binom{n}{r}} \frac{d^2}{r}|\mathcal{T}(Q)| \\
&= \frac{1}{\binom{n}{r+1}} \sum_{Q\subseteq X, |Q|=r+1} \frac{n-r}{r+1} \cdot \frac{d^2}{r}|\mathcal{T}(Q)| \\
&= \frac{d^2}{r(r+1)}(n-r)t_{r+1} \\
&= \frac{d^2 n}{r(r+1)}t_{r+1} - \frac{d^2}{r+1}t_{r+1}.
\end{aligned}$$

Summing up. Let us recapitulate: the overall expected number of conflicts is $O(n)$ plus

$$\sum_{r=1}^{n-1} k(r) \leqslant \sum_{r=1}^{n-1} (k_1(r) - k_2(r) + k_3(r)).$$

Using our previous estimates, $k_1(2), \ldots, k_1(n-1)$ are canceled by the first terms of $k_2(1), \ldots, k_2(n-2)$. The second term of $k_2(r)$ can be combined with the first term of $k_3(r)$, so that we get

$$\begin{aligned}\sum_{r=1}^{n-1} (k_1(r) - k_2(r) + k_3(r)) &\leqslant k_1(1) - \underbrace{k_1(n)}_{=0} + n\sum_{r=1}^{n-1} \frac{d(d-1)}{r(r+1)} t_{r+1} - \sum_{r=1}^{n-1} \frac{d^2}{r+1} t_{r+1} \\ &\leqslant d(n-1)t_1 + d(d-1)n \sum_{r=1}^{n-1} \frac{t_{r+1}}{r(r+1)} \\ &= O\left(d^2 n \sum_{r=1}^{n} \frac{t_r}{r^2}\right).\end{aligned}$$

The Delaunay case. We have argued that the expected number of conflicts asymptotically bounds the expected total location cost over all insertion steps. The previous equation tells us that this cost is proportional to $O(n)$ plus

$$O\left(9n \sum_{r=1}^{n} \frac{2(r+3)-4}{r^2}\right) = O\left(n \sum_{r=1}^{n} \frac{1}{r}\right) = O(n \log n).$$

Here,

$$\sum_{r=1}^{n} \frac{1}{r} =: H_n$$

is the n-th Harmonic Number which is known to be approximately $\ln n$.

By going through the abstract framework of configuration spaces, we have thus analyzed the randomized incremental construction of the Delaunay triangulation of n points. According to Section B.3, the expected update cost itself is only $O(n)$. The steps dominating the runtime are the location steps via the history graph. According to Section B.5, all history graph searches (whose number is proportional to the number of conflicts) can be performed in expected time $O(n \log n)$, and this then also bounds the space requirements of the algorithm.

Exercise B.7. *Design and analyze a sorting algorithm based on randomized incremental construction in configuration spaces. The input is a set S of numbers, and the output should be the sorted sequence (in increasing order).*

a) *Define an appropriate configuration space for the problem! In particular, the set of active configurations w.r.t.* S *should represent the desired sorted sequence.*

b) *Provide an efficient implementation of the incremental construction algorithm. "Efficient" means that the runtime of the algorithm is asymptotically dominated by the number of conflicts.*

c) *What is the expected number of conflicts (and thus the asymptotic runtime of your sorting algorithm) for a set* S *of* n *numbers?*

Questions

63. *What is a configuration space?* Give a precise definition! What is an active configuration?

64. *How do we get a configuration space from the problem of computing the Delaunay triangulation of a finite point set?*

65. *How many new active configurations do we get on average when inserting the* r*-th element?* Provide an answer for configuration spaces in general, and for the special case of the Delaunay triangulation.

66. *What is a conflict?* Provide an answer for configuration spaces in general, and for the special case of the Delaunay triangulation.

67. *Explain why counting the expected number of conflicts asymptotically bounds the cost for the history searches during the randomized incremental construction of the Delaunay triangulation!*

Appendix C

Trapezoidal Maps

In this section, we will see another application of randomized incremental construction in the abstract configuration space framework. At the same time, this will give us an efficient algorithm for solving the general problem of point location, as well as a faster algorithm for computing all intersections between a given set of line segments.

C.1 The Trapezoidal Map

To start with, let us introduce the concept of a *trapezoidal map*.

We are given a set $S = \{s_1, \ldots, s_n\}$ of line segments in the plane (not necessarily disjoint). We make several general position assumptions.

We assume that no two segment endpoints and intersection points have the same x-coordinate. As an exception, we do allow several segments to share an endpoint. We also assume that no line segment is vertical, that any two line segments intersect in at most one point (which is a common endpoint, or a proper crossing), and that no three line segments have a common point. Finally, we assume that $s_i \subseteq [0, 1]^2$ for all i (which can be achieved by scaling the coordinates of the segments accordingly).

Definition C.1. *The* trapezoidal map *of S is the partition of $[0, 1]^2$ into vertices, edges, and faces (called* trapezoids*), obtained as follows. Every segment endpoint and point of intersection between segments gets connected by two vertical extensions with the next feature below and above, where a feature is either another line segment or an edge of the bounding box $[0, 1]^2$.*

Figure C.1 gives an example.

(The general-position assumptions are made only for convenience and simplicity of the presentation. The various degeneracies can be handled without too much trouble, though we will not get into the details.)

The trapezoids of the trapezoidal map are "true" trapezoids (quadrangles with two parallel vertical sides), and triangles (which may be considered as degenerate trapezoids).

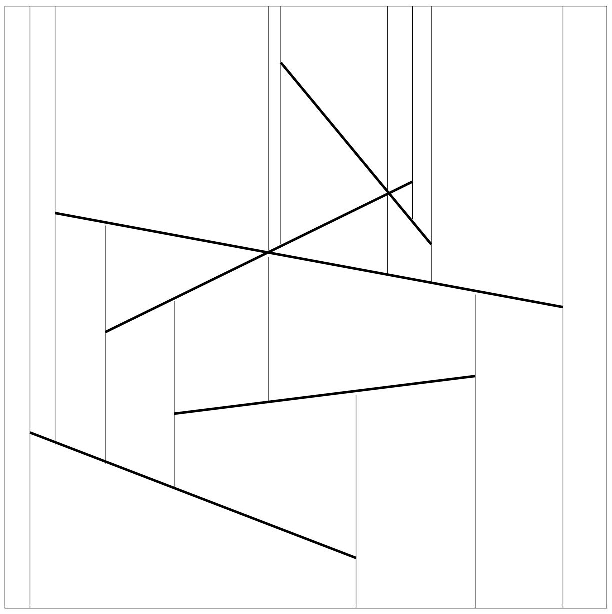


Figure C.1: *The trapezoidal map of five line segments (depicted in bold)*

C.2 Applications of trapezoidal maps

In this chapter we will see two applications of trapezoidal maps (there are others):

1. *Point location:* Given a set of n segments in the plane, we want to preprocess them in order to answer point-location queries: given a point p, return the cell (connected component of the complement of the segments) that contains p (see Figure C.2). This is a more powerful alternative to Kirkpatrick's algorithm that handles only triangulations, and which is treated in Section 7.5. The preprocessing constructs the trapezoidal map of the segments (Figure C.3) in expected time $O(n \log n + K)$, where K is the number of intersections between the input segments; the query time will be $O(\log n)$ in expectation.

2. *Line segment intersection:* Given a set of n segments, we will report all segment intersections in expected time $O(n \log n + K)$. This is a faster alternative to the classical line-sweep algorithm, which takes time $O((n + K) \log n)$.

C.3 Incremental Construction of the Trapezoidal Map

We can construct the trapezoidal map by inserting the segments one by one, in random order, always maintaining the trapezoidal map of the segments inserted so far. In order to perform manipulations efficiently, we can represent the current trapezoidal map as a doubly-connected edge list (see Section 2.2.1), for example.

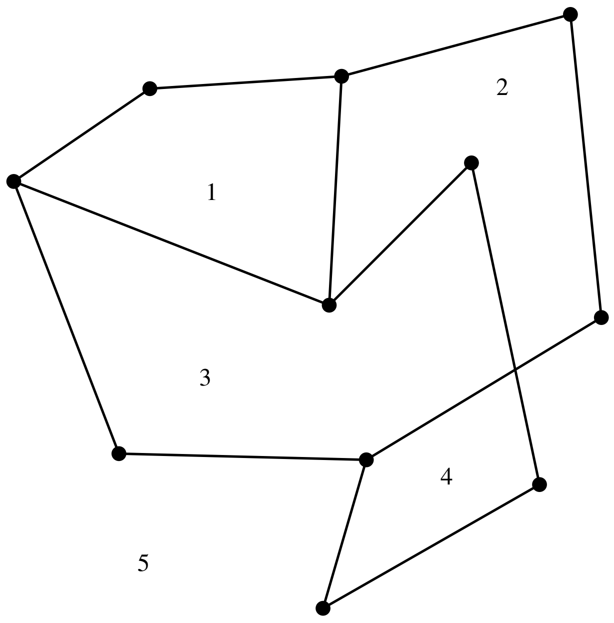


Figure C.2: *The general point location problem defined by a set of (possibly intersecting) line segments. In this example, the segments partition the plane into 5 cells.*

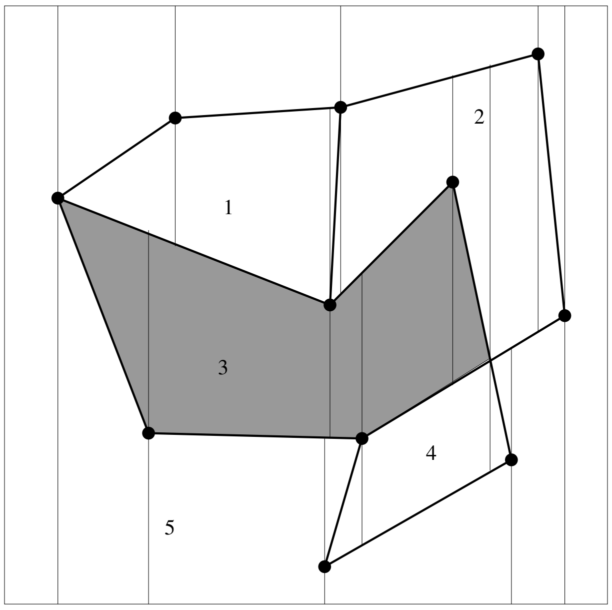


Figure C.3: *The trapezoidal map is a refinement of the partition of the plane into cells. For example, cell 3 is a union of five trapezoids.*

Suppose that we have already inserted segments $s_1, \ldots, s_{r-1}$, and that the resulting trapezoidal map $\mathcal{T}_{r-1}$ looks like in Figure C.1. Now we insert segment s_r (see Figure C.4).

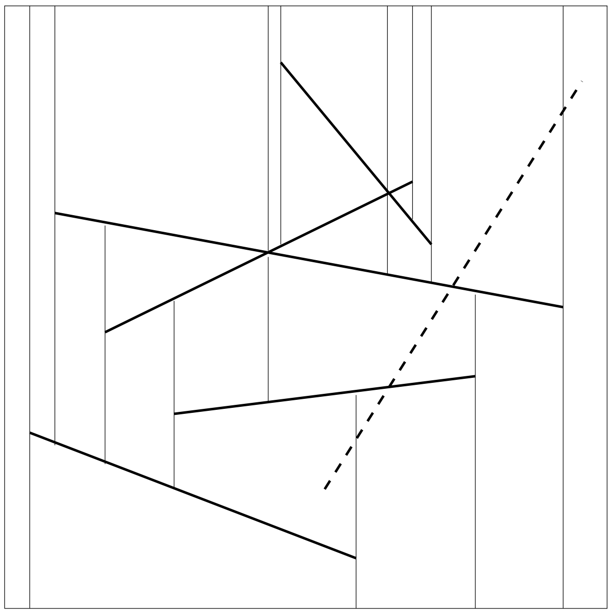


Figure C.4: *A new segment (dashed) is to be inserted*

Here are the four steps that we need to do in order to construct $\mathcal{T}_r$.

1. **Find** the trapezoid $\square_0$ of $\mathcal{T}_{r-1}$ that contains the left endpoint of s_r.
2. **Trace** s_r through $\mathcal{T}_{r-1}$ until the trapezoid containing the right endpoint of s_r is found. To get from the current trapezoid $\square$ to the next one, traverse the boundary of $\square$ until the edge is found through which s_r leaves $\square$.
3. **Split** the trapezoids intersected by s_r. A trapezoid $\square$ may get replaced by
   - two new trapezoids (if s_r intersects two vertical extensions of $\square$);
   - three new trapezoids (if s_r intersects one vertical extension of $\square$);
   - four new trapezoids (if s_r intersects no vertical extension of $\square$).
4. **Merge** trapezoids by removing parts of vertical extensions that do not belong to $\mathcal{T}_r$ anymore.

Figure C.5 illustrates the **Trace** and **Split** steps. s_6 intersects 5 trapezoids, and they are being split into $3, 3, 4, 3$, and 3 trapezoids, respectively.

The **Merge** step is shown in Figure C.6. In the example, there are two vertical edges that need to be removed (indicated with a cross) because they come from vertical extensions that are cut off by the new segment. In both cases, the two trapezoids to the left and right of the removed edge are being merged into one trapezoid (drawn shaded).

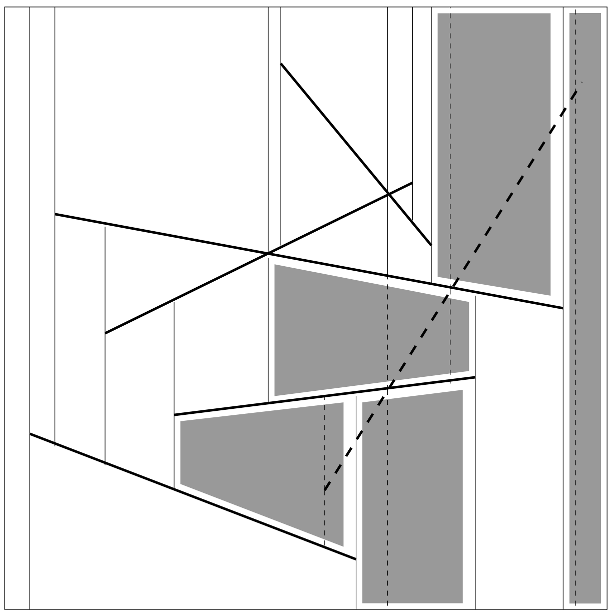


Figure C.5: *The* **Trace** *and* **Split** *steps*

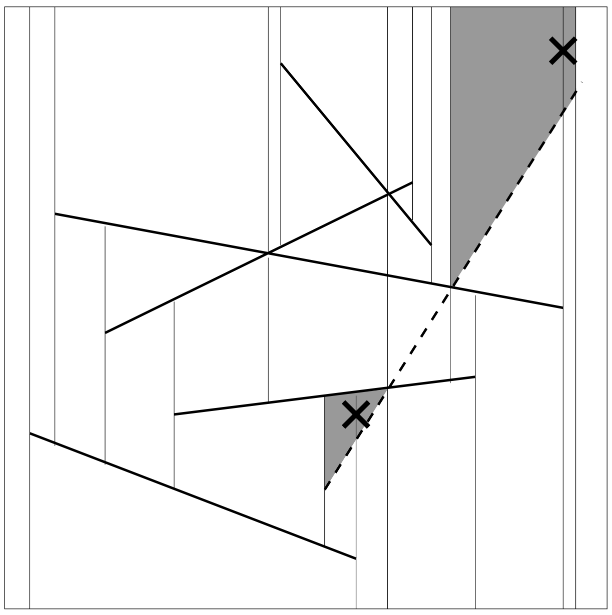


Figure C.6: *The* **Merge** *steps*

C.4 Using trapezoidal maps for point location

Recall that in the point location problem we want to preprocess a given set S of segments in order to answer subsequent point-location queries: S partitions the plane into connected *cells* and we want to know, given a query point q, to which cell q belongs, see Figure C.2.

Note that the trapezoidal map of S is a *refinement* of the partition of the plane into cells, in the sense that a cell might be partitioned into several trapezoids, but every trapezoid belongs to a single cell, see Figure C.3. Thus, once the trapezoidal map of S is constructed, we can easily “glue together" trapezoids that touch along their vertical sides, obtaining the original cells. Then we can answer point-location queries using the same routine that performs the **Find** step (whose implementation will be described below).

C.5 Analysis of the incremental construction

In order to analyze the runtime of the incremental construction, we insert the segments in random order, and we employ the configuration space framework. We also implement the **Find** step in such a way that the analysis boils down to conflict counting, just as for the Delaunay triangulation.

C.5.1 Defining The Right Configurations

Recall that a configuration space is a quadruple $\mathcal{S} = (X, \Pi, D, K)$, where X is the ground set, Π is the set of configurations, D is a mapping that assigns to each configuration its defining elements (“generators”), and K is a mapping that assigns to each configuration its conflict elements (“killers”).

It seems natural to choose $X = S$, the set of segments, and to define Π as the set of all possible trapezoids that could appear in the trapezoidal map of some subset of segments. Indeed, this satisfies one important property of configuration spaces: for each configuration, the number of generators is constant.

Lemma C.2. *For every trapezoid $\square$ in the trapezoidal map of $R \subseteq S$, there exists a set $D \subseteq R$ of at most four segments, such that $\square$ is in the trapezoidal map of D.*

Proof. By our general position assumption, each non-vertical side of $\square$ is a subset of a unique segment in R, and each vertical side of $\square$ is induced by a unique (left or right) endpoint, or by the intersection of two unique segments. In the latter case, one of these segments also contributes a non-vertical side, and in the former case, we attribute the endpoint to the “topmost” segment with that (left or right) endpoint. It follows that there is a set of at most four segments whose trapezoidal map already contains $\square$. □

But there is a problem with this definition of configurations. Recall that we can apply the general configuration space analysis only if

(i) the cost of updating $\mathcal{T}_{r-1}$ to $\mathcal{T}_r$ is proportional to the *structural change*, the number of configurations in $\mathcal{T}_r \setminus \mathcal{T}_{r-1}$; and

(ii) the expected cost of all **Find** operations during the randomized incremental construction is proportional to the expected number of conflicts. (This is the "conflict counting" part.)

Here we see that already (i) fails. During the **Trace** step, we traverse the boundary of each trapezoid intersected by s_r in order to find the next trapezoid. Even if s_r intersects only a small number of trapezoids (so that the structural change is small), the traversals may take very long. This is due to the fact that a trapezoid can be incident to a large number of edges. Consider the trapezoid labeled $\square$ in Figure C.7. It has many incident vertical extensions from above. Tracing a segment through such a trapezoid takes time that we cannot charge to the structural change.

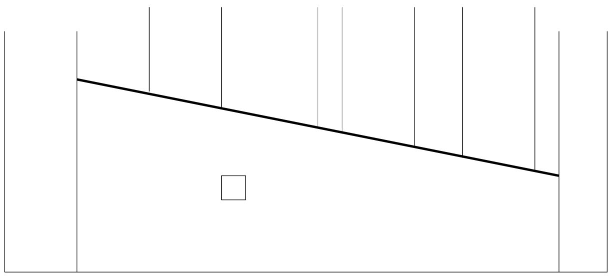


Figure C.7: *Trapezoids may have arbitrarily large complexity*

To deal with this, we slightly adapt our notion of configuration.

Definition C.3. *Let Π be the set of all trapezoids together with at most one incident vertical edge ("trapezoids with tail") that appear in the trapezoidal map of some subset of $X = S$, see Figure C.8. A trapezoid without any incident vertical edge is also considered a trapezoid with tail.*

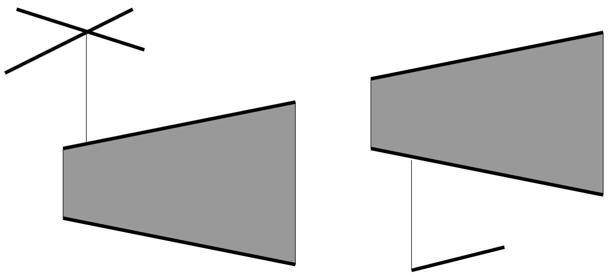


Figure C.8: *A trapezoid with tail is a trapezoid together with at most one vertical edge attached to its upper or its lower segment.*

As it turns out, we still have constantly many generators.

Lemma C.4. *For every trapezoid with tail* $\Box$ *in the trapezoidal map of* $R \subseteq S$*, there exists a set* $D \subseteq R$ *of at most six segments, such that* $\Box$ *is in the trapezoidal map of* D*.*

Proof. We already know from Lemma C.2 that the trapezoid without tail has at most 4 generators. And since the tail is induced by a unique segment endpoint or by the intersection of a unique pair of segments, the claim follows. $\Box$

Here is the complete specification of our configuration space $\mathcal{S} = (X, \Pi, D, K)$.

Definition C.5. *Let* $X = S$ *be the set of segments, and* Π *the set of all trapezoids with tail. For each trapezoid with tail* $\Box$*,* $D(\Box)$ *is the set of at most 6 generators.* $K(\Box)$ *is the set of all segments that intersect* $\Box$ *in the interior of the trapezoid, or cut off some part of the tail, or replace the topmost generator of the left or right side, see Figure C.9.*

Then $\mathcal{S} = (X, \Pi, D, K)$ is a configuration space of dimension at most 6, by Lemma C.4. The only additional property that we need to check is that $D(\Box) \cap K(\Box) = \emptyset$ for all trapezoids with tail, but this is clear since no generator of $\Box$ properly intersects the trapezoid of $\Box$ or cuts off part of its tail.

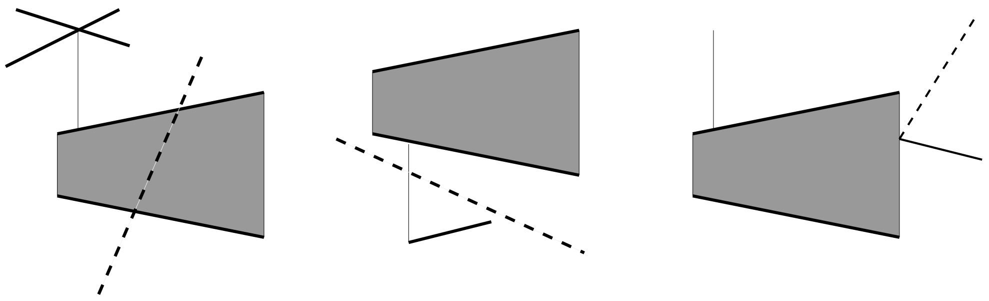


Figure C.9: *A trapezoid with tail* $\Box$ *is in conflict with a segment* s *(dashed) if* s *intersects* $\Box$ *in the interior of the trapezoid (left), or cuts off part of the tail (middle), or is a new topmost segment generating a vertical side.*

C.5.2 Update Cost

Now we can argue that the update cost can be bounded by the structural change. We employ the same trick as for Delaunay triangulations. We prove that the update cost is in each step $r-1 \to r$ proportional to the number of configurations that are being *destroyed*. Over the whole algorithm, we cannot destroy more configurations than we create, so the bound that we get is also a bound in terms of the overall structural change.

Lemma C.6. *In updating* $\mathcal{T}_{r-1}$ *to* $\mathcal{T}_r$*, the steps* Trace*,* Split*, and* Merge *can be performed in time proportional to the number of trapezoids with tail in* $\mathcal{T}_{r-1} \setminus \mathcal{T}_r$*.*

Proof. By definition, the complexity (number of edges) of a trapezoid is proportional to the number of trapezoids with tail that share this trapezoid. This means, the cost of traversing the trapezoid can be charged to the trapezoids with tail containing it, and all of them will be destroyed (this includes the trapezoids with tail that just change their left or right generator; in the configuration space, this is also a destruction). This takes care of the **Trace** step. The **Split** and **Merge** steps can be done within the same asymptotic time bounds since they can be performed by traversing the boundaries of all intersected trapezoids a constant number of times each. For efficiently doing the manipulations on the trapezoidal map, we can for example represent it using a doubly-connected edge list. □

We can now employ the general configuration space analysis to bound the *expected* structural change throughout the randomized incremental construction; as previously shown, this asymptotically bounds the expected cost of the **Trace**, **Split**, and **Merge** steps throughout the algorithm. Let us recall the general bound.

Theorem C.7. *Let* $\mathcal{S} = (X, \Pi, D, K)$ *be a configuration space of fixed dimension with* $|X| = n$. *The expected number of configurations that are created throughout the algorithm is bounded by*

$$O\left(\sum_{r=1}^{n} \frac{t_r}{r}\right),$$

where t_r *is the expected size of* $\mathcal{T}_r$, *the expected number of active configurations after* r *insertion steps.*

C.5.3 The History Graph

Here is how we realize the **Find** step (as well as the point-location queries for our point-location application). It is a straightforward history graph approach as for Delaunay triangulations. Every trapezoid that we ever create is a node in the history graph; whenever a trapezoid is destroyed, we add outgoing edges to its (at most four) successor trapezoids. Note that trapezoids are destroyed during the steps **Split** and **Merge**. In the latter step, every destroyed trapezoid has only one successor trapezoid, namely the one it is merged into. It follows that we can prune the nodes of the "ephemeral" trapezoids that exist only between the **Split** and **Merge** steps. What we get is a history graph of degree at most 4, such that every non-leaf node corresponds to a trapezoid in $\mathcal{T}_{r-1} \setminus \mathcal{T}_r$, for some r.

C.5.4 Cost of the Find step

We can use the history graph for point location during the **Find** step. Given a segment endpoint p, we start from the bounding box (the unique trapezoid with no generators) that is certain to contain p. Since for every trapezoid in the history graph, its area is covered by the at most four successor trapezoids, we can simply traverse the history graph along directed edges until we reach a leaf that contains p. This leaf corresponds to the trapezoid of the current trapezoidal map containing p. By the outdegree-4-property, the cost of the traversal is proportional to the length of the path that we traverse.

Here is the crucial observation that allows us to reduce the analysis of the **Find** step to "conflict counting". Note that this is is precisely what we also did for Delaunay triangulations, except that there, we had to deal explicitly with "ephemeral" triangles.

Recall Definition B.5, according to which a *conflict* is a pair $(\square, s)$ where $\square$ is a trapezoid with tail, contained in some intermediate trapezoidal map, and $s \in K(\square)$.

Lemma C.8. *During a run of the incremental construction algorithm for the trapezoidal map, the total number of history graph nodes traversed during all* **Find** *steps is bounded by the number of conflicts during the run.*

Proof. Whenever we traverse a node (during insertion of segment s_r, say), the node corresponds to a trapezoid $\square$ (which we also consider as a trapezoid with tail) in some set $\mathcal{T}_s, s < r$, such that $p \in \square$, where p is the left endpoint of the segment s_r. We can therefore uniquely identify this edge with the conflict $(\square, s_r)$. The statement follows. $\square$

Now we can use the configuration space analysis that precisely bounds the *expected* number of conflicts, and therefore the expected cost of the **Find** steps over the whole algorithm. Let us recapitulate the bound.

Theorem C.9. *Let $\mathcal{S} = (X, \Pi, D, K)$ be a configuration space of fixed dimension d with $|X| = n$. The expected number of conflicts during randomized incremental construction of $\mathcal{T}_n$ is bounded by*

$$O\left(n \sum_{r=1}^{n} \frac{t_r}{r^2}\right),$$

where t_r is as before the expected size of $\mathcal{T}_r$.

C.5.5 Applying the General Bounds

Let us now apply Theorem C.7 and Theorem C.9 to our concrete situation of trapezoidal maps. What we obviously need to determine for that is the quantity t_r, the expected number of active configurations after r insertion steps.

Recall that the configurations are the trapezoids with tail that exist at this point. The first step is easy.

Observation C.10. *In every trapezoidal map, the number of trapezoids with tail is proportional to the number vertices.*

Proof. Every trapezoid with tail that actually has a tail can be charged to the vertex of the trapezoidal map on the "trapezoid side" of the tail. No vertex can be charged twice in this way. The trapezoids with no tail are exactly the faces of the trapezoidal map, and since the trapezoidal map is a planar graph, their number is also proportional to the number vertices. □

Using this observation, we have therefore reduced the problem of computing t_r to the problem of computing the expected number of vertices in $\mathcal{T}_r$. To count the latter, we note that every segment endpoint and every segment intersection generates 3 vertices: one at the point itself, and two where the vertical extensions hit another feature. Here, we are sweeping the 4 bounding box vertices under the rug.

Observation C.11. *In every trapezoidal map of* r *segments, the number of vertices is*

$$6r + 3k,$$

where k *is the number of pairwise intersections between the* r *segments.*

So far, we have not used the fact that we have a random insertion order, but this comes next.

Lemma C.12. *Let* K *be the total number of pairwise intersections between segments in* S*, and let* k_r *be the random variable for the expected number of pairwise intersections between the first* r *segments inserted during randomized incremental construction. Then*

$$k_r = K\frac{\binom{n-2}{r-2}}{\binom{n}{r}} = K\frac{r(r-1)}{n(n-1)}.$$

Proof. Let us consider the intersection point of two fixed segments s and s'. This intersection point appears in $\mathcal{T}_r$ if and only both s and s' are among the first r segments. There are $\binom{n}{r}$ ways of choosing the set of r segments (and all choices have the same probability); since the number of r-element sets containing s and s' is $\binom{n-2}{r-2}$, the probability for the intersection point to appear is

$$\frac{\binom{n-2}{r-2}}{\binom{n}{r}} = \frac{r(r-1)}{n(n-1)}.$$

Summing this up over all K intersection points, and using linearity of expectation, the statement follows. □

From Observation C.10, Observation C.11 and Lemma C.12, we obtain the following

Corollary C.13. *The expected number* t_r *of active configurations after* r *insertion steps is*

$$t_r = O\left(r + K\frac{r(r-1)}{n^2}\right).$$

Plugging this into Theorem C.7 and Theorem C.9, we obtain the following final

Theorem C.14. *Let S be a set of n segments in the plane, with a total of K pairwise intersections. The randomized incremental construction computes the trapezoidal map of S in time*

$$O(n \log n + K).$$

Proof. We already know that the expected update cost (subsuming steps **Trace**, **Split**, and **Merge**) is proportional to the expected overall structural change, which by Theorem C.7 is

$$O\left(\sum_{r=1}^{n} \frac{t_r}{r}\right) = O(n) + O\left(\frac{K}{n^2}\sum_{r=1}^{n} r\right) = O(n+K).$$

We further know that the expected point location cost (subsuming step **Find**) is proportional to the overall expected number of conflicts which by Theorem C.9 is

$$O\left(n\sum_{r=1}^{n} \frac{t_r}{r^2}\right) = O(n\log n) + O\left(\frac{K}{n}\sum_{r=1}^{n} 1\right) = O(n\log n + K).$$

□

C.6 Analysis of the point location

Finally, we return to the application of trapezoidal maps for point location. We make precise what we mean by saying that "point-location queries are handled in $O(\log n)$ expected time", and we prove our claim.

Lemma C.15. *Let $S = \{s_1, \dots, s_n\}$ be any set of n segments. Then there exists a constant $c > 0$ such that, with high probability (meaning, with probability tending to 1 as $n \to \infty$), the history graph produced by the random incremental construction answers* every possible *point-location query in time at most $c \log n$.*

Note that our only randomness assumption is over the random permutation of S chosen at the beginning of the incremental construction. We do not make any randomness assumption on the given set of segments.

The proof of Lemma C.15 is by a typical application of Chernoff's bound followed by the union bound.

Recall (or please meet) Chernoff's bound:

Lemma C.16. *Let $X_1, X_2, \dots, X_n$ be independent 0/1 random variables, and let $X = X_1 + \cdots + X_n$. Let $p_i = \Pr[X_i = 1]$, and let $\mu = E[X] = p_1 + \cdots + p_n$. Then,*

$$\Pr[X < (1-\delta)\mu] < \left(\frac{e^{-\delta}}{(1-\delta)^{1-\delta}}\right)^{\mu} \qquad \text{for every } 0 < \delta < 1;$$

$$\Pr[X > (1+\delta)\mu] < \left(\frac{e^{\delta}}{(1+\delta)^{1+\delta}}\right)^{\mu} \qquad \text{for every } \delta > 0.$$

The important thing to note is that $e^{-\delta}/(1-\delta)^{1-\delta}$ as well as $e^{\delta}/(1+\delta)^{1+\delta}$ are strictly less than 1 for every fixed $\delta > 0$, and decrease with increasing δ.

Now back to the proof of Lemma C.15:

Proof. First note that, even though there are infinitely many possible query points, there is only a finite number of combinatorially distinct possible *queries*: If two query points lie together in every trapezoid (either both inside or both outside), among all possible trapezoids defined by segments of S, then there is no difference in querying one point versus querying the other, as far as the algorithm is concerned. Since there are $O(n^4)$ possible trapezoids (recall that each trapezoid is defined by at most four segments), there are only $O(n^4)$ queries we have to consider.

Fix a query point q. We will show that there exists a large enough constant $c > 0$, such that only with probability at most $O(n^{-5})$ does the query on q take more than $c \log n$ steps.

Let $s_1, s_2, \dots, s_n$ be the random order of the segments chosen by the algorithm, and for $1 \leqslant r \leqslant n$ let $\mathcal{T}_r$ be the trapezoidal map generated by the first r segments. Note that for every r, the point q belongs to exactly one trapezoid of $\mathcal{T}_r$. The question is how many times the trapezoid containing q changes during the insertion of the segments, since these are exactly the trapezoids of the history graph that will be visited when we do a point-location query on q.

For $1 \leqslant r \leqslant n$, let A_r be the event that the trapezoid containing q changes from $\mathcal{T}_{r-1}$ to $\mathcal{T}_r$. What is the probability of A_r? As in Section B.3, we "run the movie backwards": To obtain $\mathcal{T}_{r-1}$ from $\mathcal{T}_r$, we delete a random segment from among $s_1, \dots, s_r$; the probability that the trapezoid containing q in $\mathcal{T}_r$ is destroyed is at most $4/r$, since this trapezoid is defined by at most four segments. Thus, $\Pr[A_r] \leqslant 4/r$, independently of every other A_s, $s \neq r$.

For each r let X_r be a random variable equal to 1 if A_r occurs, and equal to 0 otherwise. We are interested in the quantity $X = X_1 + \cdots + X_r$. Then $\mu = E[X] = \sum_{r=1}^{n} \Pr[A_r] = 4 \ln n + O(1)$. Applying Chernoff's bound with $\delta = 2$ (it is just a matter of choosing δ large enough), we get

$$\Pr[X > (1+\delta)\mu] < 0.273^{\mu} = O(0.273^{4 \ln n}) = O(n^{-5.19}),$$

so we can take our c to be anything larger than $4(1+\delta) = 12$.

Thus, for every fixed query q, the probability of a "bad event" (a permutation that results in a long query time) is $O(n^{-5})$. Since there are only $O(n^4)$ possible choices for q, by the union bound the probability of *some* q having a bad event is $O(1/n)$, which tends to zero with n. □

C.7 The trapezoidal map of a simple polygon

An important special case of the trapezoidal map is obtained when the input segments form a simple polygon; seee Figure C.10. In this case, we are mostly interested in the part of the trapezoidal map inside the polygon, since that parts allows us to obtain a triangulation of the polygon in linear time.

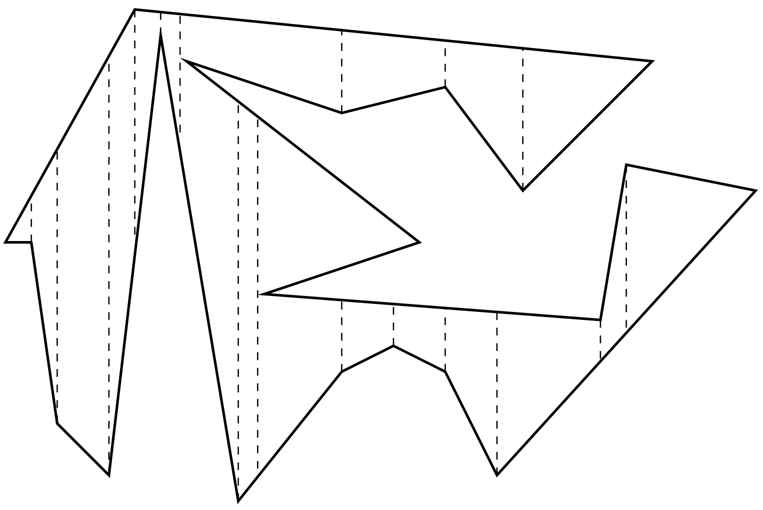


Figure C.10: *The trapezoidal map inside a simple polygon*

To get a triangulation, we first go through all trapezoids; we know that each trapezoid must have one polygon vertex on its left and one on its right boundary (due to general position, there is actually *exactly* one vertex on each of these two boundaries). Whenever the segment connecting these two vertices is not an edge of the polygon, we have a diagonal, and we insert this diagonal. Once we have done this for all trapezoids, it is easily seen that we have obtained a subdivision of the polygon into x-monotone polygons, each of which can be triangualated in linear time; see Exercise C.24. This immediately allows us to improve over the statement of Exercise 3.17.

Corollary C.17. *A simple polygon with* n *vertices can be triangulated in expected time* $O(n \log n)$*.*

Proof. By Theorem C.14, the trapezoidal decomposition induced by the segments of a simple polygon can be combuted in expected time $O(n \log n)$, since there are no intersections between the segments. Using the above linear-time triangulation algorithm from the trapezoidal map, the result follows. □

The goal of this section is to further improve this bound and show the following result.

Theorem C.18. *Let* $S = \{s_1, s_2, \ldots, s_n\}$ *be the set of edges of an* n*-vertex simple polygon, in counterclockwise order around the polygon. The trapezoidal map induced by* S *(and thus also a triangulation of* S*) can be computed in exptected time* $O(n \log^* n)$*.*

Informally speaking, the function $\log^* n$ is the number of times we have to iterate the operation of taking (binary) logarithms, before we get from n down to 1. Formally, we define

$$\log^{(h)}(n) = \begin{cases} n, & \text{if } h = 0 \\ \log^{(h-1)}(\log n), & \text{otherwise} \end{cases}$$

as the h-times iterated logarithm, and for $n \geqslant 1$ we set

$$\log^* n = \max\{h : \log^{(h)} n \geqslant 1\}.$$

For example, we have

$$\log^*(2^{65536}) = 5,$$

meaning that for all practical purposes, $\log^* n \leqslant 5$; a bound of $O(n \log^* n)$ is therefore very close to a linear bound.

History flattening. Recall that the bottleneck in the randomized incremental construction is the **Find** step. Using the special structure we have (the segments form a simple polygon in this order), we can speed up this step. Suppose that at some point during incremental construction, we have built the trapezoidal map of a subset of r segments, along with the history graph. We now flatten the history by removing all trapezoids that are not in $\mathcal{T}_r$, the current trapezoidal map. To allow for point location also in the future, we need an "entry point" into the flattened history, for every segment not inserted so far (the old entry point for all segments was the bounding unit square $[0, 1]^2$).

Lemma C.19. *Let S be the set of edges of an n-vertex simple polygon, in counterclockwise order around the polygon. For $R \subseteq S$, let $\mathcal{T}(R)$ be the trapezoidal map induced by R. In time proportional to n plus the number of conflicts between trapezoids in $\mathcal{T}(R)$ and segments in $S \setminus R$, we can find for all segment $s \in S$ a trapezoid of $\mathcal{T}(R)$ that contains an endpoint of s.*

Proof. In a trivial manner (and in time $O(n)$), we do this for the first segment s_1 and its first endpoint p_1. Knowing the trapezoid $\square_i$ containing p_i, the first endpoint of s_i, we trace the segment s_i through $\mathcal{T}(R)$ until p_{i+1}, its other endpoint and first endpoint of s_{i+1} is found, along with the trapezoid $\square_{i+1}$ containing it. This is precisely what we also did in the **Trace** Step 2 of the incremental construction in Section C.3.

By Lemma C.6 (pretending that we are about to insert s_i), the cost of tracing s_i through $\mathcal{T}(R)$ is proportional to the size of $\mathcal{T}(R) \setminus \mathcal{T}(R \cup \{s_i\})$, equivalently, the number of conflicts between trapezoids in $\mathcal{T}(R)$ and s_i. The statement follows by adding up the costs for all i. □

This is exactly where the special structure of our segments forming a polygon helps. After locating p_i, we can locate the next endpoint p_{i+1} in time proportional to the structural change that the insertion of s_i would cause. We completely avoid the traversal of old history trapezoids that would be necessary for an efficient location of p_{i+1} from scratch.

Next we show what Lemma C.19 gives us in expectation.

Lemma C.20. *Let S be the set of edges of an n-vertex simple polygon, in counterclockwise order around the polygon, and let $\mathcal{T}_r$ be the trapezoidal map obtained after inserting r segments in random order. In expected time $O(n)$, we can find for each segment s not inserted so far a trapezoid of $\mathcal{T}_r$ containing an endpoint of s.*

Proof. According to Lemma C.19, the expected time is bounded by $O(n + \ell(r))$, where

$$\ell(r) = \frac{1}{\binom{n}{r}} \sum_{R \subseteq S, |R|=r} \sum_{y \in S \setminus R} |\{\square \in \mathcal{T}(R) : y \in K(\square)\}|.$$

This looks very similar to the bound on the quantity $k(r)$ that we have derived in Section B.5 to count the expected number of conflicts in general configuration spaces:

$$k(r) \leqslant \frac{1}{\binom{n}{r}} \sum_{R \subseteq S, |R|=r} \frac{d}{r} \sum_{y \in S \setminus R} |\{\Delta \in \mathcal{T}(R) : y \in K(\Delta)\}|.$$

Indeed, the difference is only an additional factor of $\frac{d}{r}$ in the latter. For $k(r)$, we have then computed the bound

$$k(r) \leqslant k_1(r) - k_2(r) + k_3(r) \leqslant \frac{d}{r}(n-r)t_r - \frac{d}{r}(n-r)t_{r+1} + \frac{d^2}{r(r+1)}(n-r)t_{r+1}, \quad \text{(C.21)}$$

where t_r is the expected size of $\mathcal{T}_r$. Hence, to get a bound for $\ell(r)$, we simply have to cancel $\frac{d}{r}$ from all terms to obtain

$$\ell(r) \leqslant (n-r)t_r - (n-r)t_{r+1} + \frac{d}{r+1}(n-r)t_{r+1} = O(n),$$

since $t_r \leqslant t_{r+1} = O(r)$ in the case of nonintersecting line segments. □

The faster algorithm. We proceed as in the regular randomized incremental construction, except that we frequently and at well-chosen points flatten the history. Let us define

$$N(h) = \left\lceil \frac{n}{\log^{(h)} n} \right\rceil, \quad 0 \leqslant h \leqslant \log^* n.$$

We have $N(0) = 1, N(\log^* n) \leqslant n$ and $N(\log^* n + 1) > n$. We insert the segments in random order, but proceed in $\log^* n + 1$ *rounds*. In round $h = 1, \ldots, \log^* n + 1$, we do the following.

(i) Flatten the history graph by finding for each segment s not inserted so far a trapezoid of the current trapezoidal map containing an endpoint of s.

(ii) Insert the segments $N(h-1)$ up to $N(h)-1$ in the random order, as usual, but starting from the flat history established in (i).

In the last round, we have $N(h)-1 \geqslant n$, so we stop with segment n in the random order.

From Lemma C.20, we know that the expected cost of step (i) over all rounds is bounded by $O(n \log^* n)$ which is our desired overall bound. It remains to prove that the same bound also deals with step (ii). We do not have to worry about the overall expected cost of performing the structural changes in the trapezoidal map: this will be bounded by $O(n)$, using $t_r = O(r)$ and Theorem C.7. It remains to analyze the **Find** step, and this is where the history flattening leads to a speedup. Adapting Lemma C.8 and its proof accordingly, we obtain the following.

Lemma C.22. *During round h of the fast incremental construction algorithm for the trapezoidal map, the total number of history graph nodes traversed during all* **Find** *steps is bounded by $N(h)-N(h-1)$*[1]*, plus the number of conflicts between trapezoids that are created during round h, and segments inserted during round h.*

Proof. The history in round h only contains trapezoids that are active at some point in round h. On the one hand, we have the "root nodes" present immediately after flattening the history, on the other hand the trapezoids that are created during the round. The term $N(h)-N(h-1)$ accounts for the traversals of the root nodes during round h. As in the proof of Lemma C.8, the traversals of other history graph nodes can be charged to the number of conflicts between trapezoids that are created during round h and segments inserted during round h. □

To count the expected number of conflicts involving trapezoids created in round h, we go back to the general configuration space framework once more. With $k_h(r)$ being the expected number of conflicts created in step r, *restricted* to the ones involving segments inserted in round h, we need to bound

$$\kappa(h) = \sum_{r=N(h-1)}^{N(h)-1} k_h(r).$$

As in (C.21), we get

$$k_h(r) \leqslant \frac{d}{r}(N(h)-r)t_r - \frac{d}{r}(N(h)-r)t_{r+1} + \frac{d^2}{r(r+1)}(N(h)-r)t_{r+1},$$

[1]this amounts to $O(n)$ throughout the algorithm and can therefore be neglected

where we have replaced n with $N(h)$, due to the fact that we do not need to consider segments in later rounds. Then $t_r = O(r)$ yields

$$\begin{aligned}
\kappa(h) &\leqslant O\left(N(h)\sum_{r=N(h-1)}^{N(h)-1}\frac{1}{r}\right)\\
&= O\left(N(h)\log\frac{N(h)}{N(h-1)}\right)\\
&= O\left(N(h)\log\frac{\log^{(h-1)}n}{\log^{(h)}n}\right)\\
&= O\left(N(h)\log^{(h)}n\right) = O(n).
\end{aligned}$$

It follows that step (ii) of the fast algorithm also requires expected linear time per round, and Theorem C.18 follows.

Exercise C.23. *a) You are given a set of* n *pairwise disjoint line segments. Find an algorithm to answer* vertical ray shooting *queries in* $O(\log n)$ *time. That is, preprocess the data such that given a query point* q *you can report in* $O(\log n)$ *time which segment is the first above* q *(or if there are none). Analyze the running time and the space consumption of the preprocessing.*

b) What happens if we allow intersections of the line segments? Explain in a few words how you have to adapt your solution and how the time and space complexity would change.

Exercise C.24. *Show that an* n*-vertex* x*-monotone polygon can be triangulated in time* $O(n)$*. (As usual a polygon is given as a list of its vertices in counterclockwise order. A polygon* P *is called* x*-monotone if for all vertical lines* ℓ*, the intersection* $\ell \cap P$ *has at most one component.)*

Questions

68. *What is the definition of a trapezoidal map?*
69. *How does the random incremental construction of a trapezoidal map proceed?* What are the main steps to be executed at each iteration?
70. *How can trapezoidal maps be used for the point location problem?*
71. *What is the configuration space framework?* Recall Section B.3.
72. *What is a naive way of defining a configuration in the case of trapezoids, and why does it fail?*
73. *What is a more successful way of defining a configuration?* Why do things work in this case?

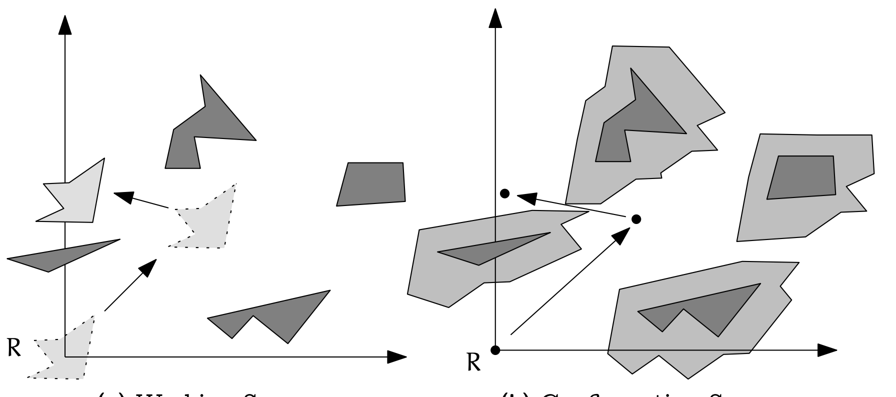
74. *What is the history graph, and how is it used to answer point location queries?*
75. *What is the performance of the random incremental construction of the trapezoidal map when used for the point-location problem?* Be precise!
76. *What probabilistic techniques are used in proving this performance bound?*
77. *How can you speed up the randomized incremental construction in the case where the input segments from a simple polygon? Sketch the necessary changes to the algorithm, and how they affect the analysis.*

Appendix D

Translational Motion Planning

In a basic instance of motion planning, a robot—modeled as a simple polygon R—moves by translation amidst a set $\mathcal{P}$ of polygonal obstacles. Throughout its motion the robot must not penetrate any of the obstacles but touching them is allowed. In other words, the interior of R must be disjoint from the obstacles at any time. Formulated in this way, we are looking at the motion planning problem in *working space* (Figure D.1a).

However, often it is useful to look at the problem from a different angle, in so-called *configuration space*. Starting point is the observation that the placement of R is fully determined by a vector $\vec{v} = (x, y) \in \mathbb{R}^2$ specifying the translation of R with respect to the origin. Hence, by fixing a *reference point* in R and considering it the origin, the robot becomes a point in configuration space (Figure D.1b). The advantage of this point of view is that it is much easier to think of a point moving around as compared to a simple polygon moving around.



(a) Working Space. **(b)** Configuration Space.

Figure D.1: *Working space and configuration space for a robot* R *and a collection of polygonal obstacles.*

The next question is: How do obstacles look like in configuration space? For an obstacle $P \in \mathcal{P}$ the set $\mathcal{C}(P) = \{\vec{v} \in \mathbb{R}^2 \mid R+\vec{v} \cap P \neq \emptyset\}$ in configuration space corresponds to the obstacle P in the original setting. We write $R+\vec{v}$ for the *Minkowski sum* $\{\vec{r}+\vec{v} \mid \vec{r} \in R\}$. Our interest is focused on the set $\mathcal{F} = \mathbb{R}^2 \setminus \bigcup_{P\in\mathcal{P}} \mathcal{C}(P)$ of *free placements* in which the robot does not intersect any obstacle.

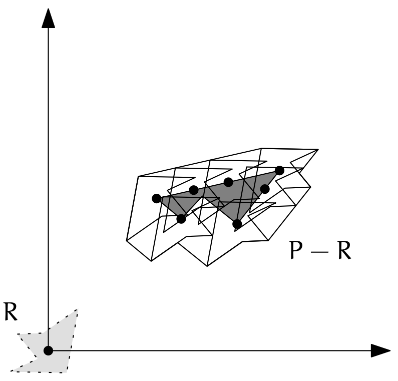


Figure D.2: *The Minkowski sum of an obstacle with an inverted robot.*

Proposition D.1. $\mathcal{C}(P) = P - R$.

Proof. $\vec{v} \in P - R \iff \vec{v} = \vec{p} - \vec{r}$, for some $\vec{p} \in P$ and $\vec{r} \in R$. On the other hand, $R + \vec{v} \cap P \neq \emptyset \iff \vec{r} + \vec{v} = \vec{p}$, for some $\vec{p} \in P$ and $\vec{r} \in R$. □

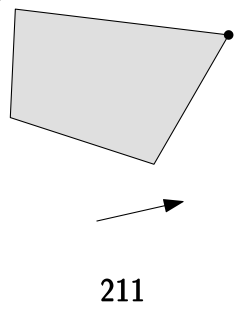
D.1 Complexity of Minkowski sums

Recall that the Minkowski sum of two point sets $P, Q \subseteq \mathbb{R}$ is defined as $P + Q = \{p + q \mid p \in P, q \in Q\}$.

Theorem D.2. *Let* P *be an* m*-vertex polygon and* Q *an* n*-vertex polygon. Then:*

1. *If both* P *and* Q *are convex, then their Minkowski sum* $P+Q$ *has at most* $m+n$ *vertices.*
2. *If* P *or* Q *is convex, then* $P + Q$ *has* $O(mn)$ *vertices.*
3. *In any case,* $P + Q$ *has* $O(m^2n^2)$ *vertices.*

Proof. The first claim can be proven using the notion of *extremal points*. If $\overrightarrow{d} = (d_x, d_y)$ is a direction in the plane and $P \subseteq \mathbb{R}^2$ is a set of points, then an *extremal point of* P *in direction* $\overrightarrow{d}$ is a point $p = (p_x, p_y) \in P$ that maximizes $p_x d_y + p_y d_x$:



It is easy to see that, if P and Q are convex polygons and $r \in P + Q$ is an extremal point of $P + Q$ in some direction $\vec{d}$, then r is the sum of some extremal points $p \in P$ and $q \in Q$ in direction $\vec{d}$. If $\vec{d}$ is chosen in "general position", then p, q, and r will be *vertices* of P, Q, and $P + Q$, respectively.

For every $\vec{d}$ in general position, let $p_{\vec{d}}$ and $q_{\vec{d}}$ be the extremal points in P and Q, respectively, in direction $\vec{d}$. Then the vertices of $P+Q$ are the precisely sums $p_{\vec{d}} + q_{\vec{d}}$ over all $\vec{d}$. But as $\vec{d}$ varies continuously and makes a full turn, the pair $(p_{\vec{d}}, q_{\vec{d}})$ can change at most $m + n$ times. This proves the first claim.

For the second claim we need the notion of *pseudodiscs*. A set $\mathcal{P} = \{P_1, \dots, P_n\}$ of objects in the plane is called a *set of pseudodiscs* if, for every distinct i and j, the boundaries of P_i and P_j intersect in at most two points. That is, the objects "behave" the way discs behave, even though they might not be actual discs.

(Note that it makes no sense to ask whether a single object P_i is a pseudodisc; it only makes sense to ask whether a *set* of objects is a set of pseudodiscs.)

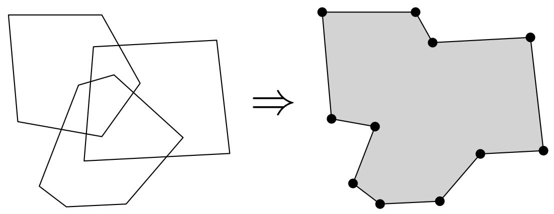
Now we need an auxiliary lemma:

Lemma D.3. *Let $\mathcal{P}$ be a set of convex objects in the plane that are pairwise interior-disjoint, and let Q be a convex object in the plane. Then the set $\mathcal{P} + Q = \{P + Q \mid P \in \mathcal{P}\}$ is a set of pseudodiscs.*

Proof. Suppose for a contradiction that the boundaries of $P_1 + Q$ and $P_2 + Q$ intersect four times, for some $P_1, P_2 \in \mathcal{P}$. This means that there are four directions $\vec{d}_1$, $\vec{d}_2$, $\vec{d}_3$, $\vec{d}_4$, in this circular order, such that P_1 is more extreme than P_2 in directions $\vec{d}_1$ and $\vec{d}_3$, while P_2 is more extreme than P_1 in directions $\vec{d}_2$ and $\vec{d}_4$. But such an alternation cannot happen since P_1 and P_2 are interior-disjoint. □

And we need a second auxiliary lemma:

Lemma D.4. *Let $\mathcal{P}$ be a set of polygonal pseudodiscs with n vertices in total. Then their union $U = \bigcup \mathcal{P}$ hast at most $2n$ vertices.*



Proof. We charge every vertex of the union U to a vertex of $\mathcal{P}$ in such a way that every vertex of $\mathcal{P}$ receives at most two charges.

Let v be a vertex of U. If v is a vertex of $\mathcal{P}$ then we simply charge v to itself. The other case is where v is the intersection point of two edges e_1, e_2, belonging to the boundaries of two distinct polygons $P_1, P_2 \in \mathcal{P}$. In such a case there must be an endpoint w of one edge e_1 or e_2 that lies inside the other polygon P_2 or P_1 (since $\mathcal{P}$ is a set of pseudodiscs). We charge v to w. It is clear that w can receive at most two charges (coming from the two edges adjacent to w). And every vertex of a polygon in $\mathcal{P}$ that is not contained in any other polygon receives at most one charge. □

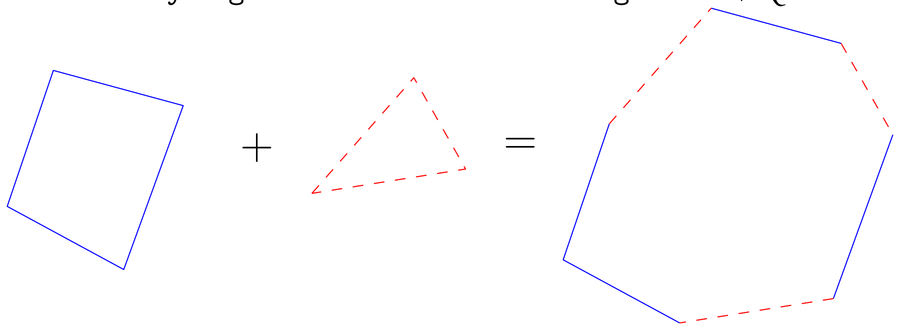
Now we are ready to prove the second claim of Theorem D.2: Suppose P is convex. Triangulate Q into $n-2$ triangles $T_1, T_2, \ldots, T_{n-2}$. Then $P + Q = \bigcup_i (P + T_i)$. By the first claim of our Theorem, each $P + T_i$ is a convex polygon with at most $m+3$ vertices. Therefore, by Lemmas D.3 and D.4, their union has at most $2(m+3)(n-2) = O(mn)$ vertices.

For the third part of our Theorem, let P and Q be arbitrary polygons. Triangulate them into triangles $S_1, \ldots, S_{m-2}$ and $T_1, \ldots, T_{n-2}$, respectively. Then $P + Q = \bigcup_{i,j}(S_i + T_j)$. Arguing as before, for every fixed i, the union $X_i = \bigcup_j (S_i + T_j)$ has at most $12(n-2)$ vertices and as many edges. Each vertex in $P + Q = \bigcup_i X_i$ is either a vertex of some X_i, or the intersection of two edges in two different X_i, X_j. There are at most $\binom{12(m-2)(n-2)}{2} = O(m^2n^2)$ vertices of the latter type. □

We leave as an exercise to show that each case of Theorem D.2 is tight in the worst case.

D.2 Minkowski sum of two convex polygons

Let P and Q be convex polygons, given as circular lists of vertices. Construct the corresponding circular lists of edges E_P and E_Q. Merge E_P and E_Q into a single list of edges E that is sorted by angle. Then E is the list of edges of $P + Q$:



Merging of two sorted lists can be done in linear time.

D.3 Constructing a single face

Theorem D.5. *Given a set S of n line segments and a point $x \in \mathbb{R}^2$, the face of $\mathcal{A}(S)$ that contains x can be constructed in $O(\lambda_3(n)\log n)$ expected time.*

Phrased in terms of translational motion planning this means the following.

Corollary D.6. *Consider a simple polygon R with k edges (robot) and a polygonal environment $\mathcal{P}$ that consists of n edges in total. The free space of all positions of R*

that can be reached by translating it without properly intersecting an obstacle from $\mathcal{P}$ *has complexity* $O(\lambda_3(kn))$ *and it can be constructed in* $O(\lambda_3(kn)\log(kn))$ *expected time.*

Below we sketch[1] a proof of Theorem D.5 using a randomized incremental construction, by constructing the trapezoidal map induced by the given set S of segments. As before, suppose without loss of generality that no two points (intersection points or endpoints) have the same x-coordinate.

In contrast to the algorithm you know, here we want to construct a single cell only, the cell that contains x. Whenever a segment closes a face, splitting it into two, we discard one of the two resulting faces and keep only the one that contains x. To detect whether a face is closed, use a disjoint-set (union-find) data structure on S. Initially, all segments are in separate components. The runtime needed for the disjoint-set data structure is $O(n\alpha(n))$, which is not a dominating factor in the bound we are heading for.

Insert the segments of S in order $s_1, \ldots, s_n$, chosen uniformly at random. Maintain (as a doubly connected edge list) the trapezoidal decomposition of the face f_i, $1 \leqslant i \leqslant n$, of the arrangement $\mathcal{A}_i$ of $\{s_1, \ldots, s_i\}$ that contains x.

As a third data structure, maintain a *history dag* (directed acyclic graph) on all trapezoids that appeared at some point during the construction. For each trapezoid, store the (at most four) segments that define it. The root of this dag corresponds to the entire plane and has no segments associated to it.

Those trapezoids that are part of the current face f_i appear as *active* leaves in the history dag. There are two more categories of vertices: Either the trapezoid was destroyed at some step by a segment crossing it; in this case, it is an interior vertex of the history dag and stores links to the (at most four) new trapezoids that replaced it. Or the trapezoid was cut off at some step by a segment that did not cross it but excluded it from the face containing x; these vertices are called *inactive* leaves and they will remain so for the rest of the construction.

Insertion of a segment s_{r+1} comprises the following steps.

1. Find the cells of the trapezoidal map f_r^* of f_r that s_{r+1} intersects by propagating s_{r+1} down the history dag.

2. Trace s_{r+1} through the active cells found in Step 1. For each split, store the new trapezoids with the old one that is replaced.

   Wherever in a split s_{r+1} connects two segments s_j and s_k, join the components of s_j and s_k in the union find data structure. If they were in the same component already, then f_r is split into two faces. Determine which trapezoids are cut off from f_{r+1} at this point by alternately exploring both components using the DCEL structure. (Start two depth-first searches one each from the two local trapezoids incident to s_{r+1}. Proceed in both searches alternately until one is finished. Mark

[1]For more details refer to the book of Agarwal and Sharir [1].

all trapezoids as discarded that are in the component that does not contain x.) In this way, the time spent for the exploration is proportional to the number of trapezoids discarded and every trapezoid can be discarded at most once.

3. Update the history dag using the information stored during splits. This is done only after all splits have been processed in order to avoid updating trapezoids that are discarded in this step.

The analysis is completely analogous to the case where the whole arrangement is constructed, except for the expected number of trapezoids created during the algorithm. Recall that any potential trapezoid τ is defined by at most four segments from S. Denote by t_r the expected number of trapezoids created by the algorithm after insertion of $s_1, \ldots, s_r$. Then in order for τ to be created at a certain step of the algorithm, one of these defining segments has to be inserted last. Therefore,

$$\Pr[\tau \text{ is created by inserting } s_r] \leqslant \frac{4}{r} \Pr[\tau \text{ appears in } f_r^*]$$

and

$$\begin{aligned}
t_r &= \sum_{\tau} \Pr[\tau \text{ is created in one of the first } r \text{ steps}] \\
&\leqslant \sum_{\tau} \sum_{i=1}^{r} \frac{4}{i} \Pr[\tau \text{ appears in } f_i^*] \\
&= \sum_{i=1}^{r} \frac{4}{i} \sum_{\tau} \Pr[\tau \text{ appears in } f_i^*] \\
&\overset{\text{Theorem 8.34}}{\leqslant} \sum_{i=1}^{r} \frac{4}{i} O(\lambda_3(i)) \\
&\leqslant \sum_{i=1}^{r} \frac{4}{i} c i \alpha(i) = 4c \sum_{i=1}^{r} \alpha(i) \leqslant 4cr\alpha(r) = O(\lambda_3(r)) \,.
\end{aligned}$$

Using the notation of the configuration space framework, the expected number of conflicts is bounded from above by

$$\begin{aligned}
\sum_{r=1}^{n-1} (k_1 - k_2 + k_3) &\leqslant 16(n-1) + 12n \sum_{r=1}^{n-1} \frac{\lambda_3(r+1)}{r(r+1)} \\
&\leqslant 16(n-1) + 12 \sum_{r=1}^{n-1} \frac{n}{r+1} \lambda_3(r+1) \frac{1}{r} \\
&\leqslant 16(n-1) + 12 \sum_{r=1}^{n-1} \frac{\lambda_3(n)}{r} \\
&= 16(n-1) + 12\lambda_3(n) H_{n-1} \\
&= O(\lambda_3(n) \log n) \,.
\end{aligned}$$

Exercise D.7. *Show that the Minkowski sum of two convex polygons* P *and* Q *with* m *and* n *vertices, respectively, is a convex polygon with at most* $m + n$ *edges. Give an* $O(m + n)$ *time algorithm to construct it.*

Exercise D.8. *Given an ordered set* $X = (x_1, \ldots, x_n)$ *and a weight function* $w : X \to \mathbb{R}^+$*, show how to construct in* $O(n)$ *time a binary search tree on* X *in which* x_k *has depth* $O(1 + \log(W/w(x_k)))$*, for* $1 \leqslant k \leqslant n$*, where* $W = \sum_{i=1}^{n} w(x_i)$.

Questions

78. *What is the configuration space model for (translational) motion planning, and what does it have to do with arrangements (of line segments)?* Explain the working space/configuration space duality and how to model obstacles in configuration space.
79. *What is a Minkowski sum?*
80. *What is the maximum complexity of the Minkowski sum of two polygons? What if one of them is convex? If both are convex?*
81. *How can one compute the Minkowski sum of two convex polygons in linear time?*
82. *Can one construct a single face of an arrangement (of line segments) more efficiently compared to constructing the whole arrangement?* Explain the statement of Theorem D.5 and give a rough sketch of the proof.

References

[1] Pankaj K. Agarwal and Micha Sharir, *Davenport-Schinzel sequences and their geometric applications*, Cambridge University Press, New York, NY, 1995.

Appendix E

Linear Programming

This lecture is about a special type of optimization problems, namely *linear programs*. We start with a geometric problem that can directly be formulated as a linear program.

E.1 Linear Separability of Point Sets

Let $P \subseteq \mathbb{R}^d$ and $Q \subseteq \mathbb{R}^d$ be two finite point sets in d-dimensional space. We want to know whether there exists a hyperplane that separates P from Q (we allow non-strict separation, i.e. some points are allowed to be on the hyperplane). Figure E.1 illustrates the 2-dimensional case.

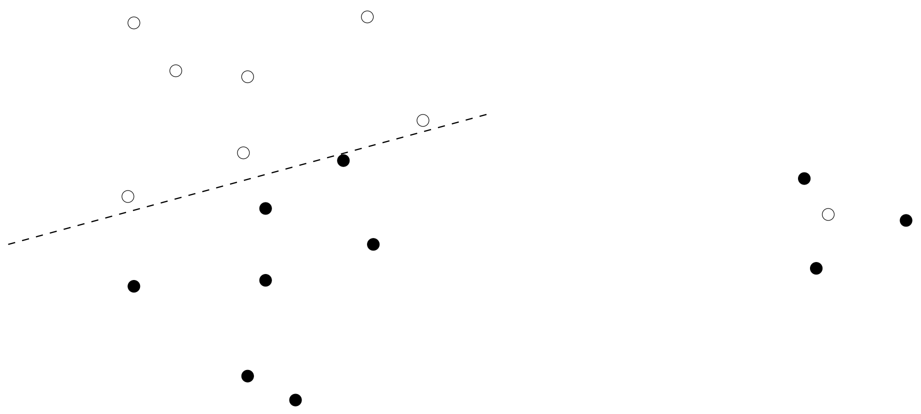


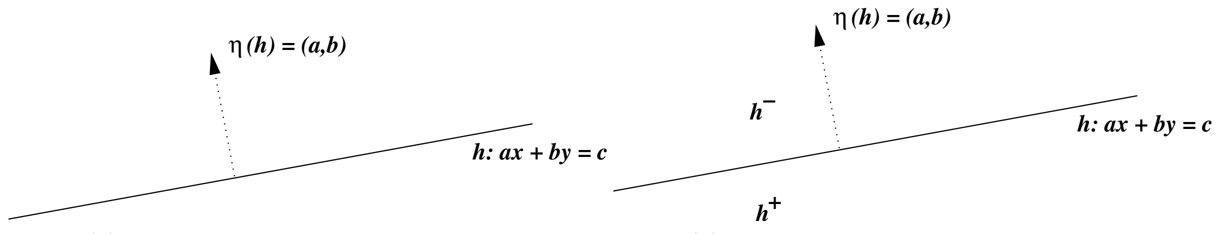
Figure E.1: *Left: there is a separating hyperplane; Right: there is no separating hyperplane*

How can we formalize this problem? A hyperplane is a set of the form

$$h = \{x \in \mathbb{R}^d : h_1x_1 + h_2x_2 + \cdots + h_dx_d = h_0\},$$

where $h_i \in \mathbb{R}, i = 0, \ldots, d$. For example, a line in the plane has an equation of the form $ax + by = c$.

The vector $\eta(h) = (h_1, h_2, \ldots, h_d) \in \mathbb{R}^d$ is called the *normal vector* of h. It is orthogonal to the hyperplane and usually visualized as in Figure E.2(a).



(a) The normal vector of a hyperplane **(b)** The two halfspaces of a hyperplane

Figure E.2: *The concepts of hyperplane, normal vector, and halfspace*

Every hyperplane h defines two closed *halfspaces*

$$\begin{aligned} h^+ &= \{x \in \mathbb{R}^d : h_1x_1 + h_2x_2 + \cdots + h_dx_d \leqslant h_0\}, \\ h^- &= \{x \in \mathbb{R}^d : h_1x_1 + h_2x_2 + \cdots + h_dx_d \geqslant h_0\}. \end{aligned}$$

Each of the two halfpsaces is the region of space "on one side" of h (including h itself). The normal vector $\eta(h)$ points into h^-, see Figure E.2(b). Now we can formally define linear separability.

Definition E.1. *Two point sets* $P \subseteq \mathbb{R}^d$ *and* $Q \subseteq \mathbb{R}^d$ *are called* linearly separable *if there exists a hyperplane* h *such that* $P \subseteq h^+$ *and* $Q \subseteq h^-$. *In formulas, if there exist real numbers* $h_0, h_1, \ldots, h_d$ *such that*

$$\begin{aligned} h_1p_1 + h_2p_2 + \cdots + h_dp_d &\leqslant h_0, \quad p \in P, \\ h_1q_1 + h_2q_2 + \cdots + h_dq_d &\geqslant h_0, \quad q \in Q. \end{aligned}$$

As we see from Figure E.1, such $h_0, h_1, \ldots, h_d$ may or may not exist. How can we find out?

E.2 Linear Programming

The problem of testing for linear separability of point sets is a special case of the general linear programming problem.

Definition E.2. *Given* $n, d \in \mathbb{N}$ *and real numbers*

$$\begin{aligned} &b_i \quad , \quad i = 1, \ldots, n, \\ &c_j \quad , \quad j = 1, \ldots, d, \\ &a_{ij} \quad , \quad i = 1, \ldots, n, \quad j = 1, \ldots, d, \end{aligned}$$

the linear program *defined by these numbers is the problem of finding real numbers* $x_1, x_2, \ldots, x_d$ *such that* *(i)* $\sum_{j=1}^{d} a_{ij}x_j \leqslant b_i, \quad i = 1, \ldots, n,$ *and*

(ii) $\sum_{j=1}^{d} c_j x_j$ *is as large as possible subject to (i).*

Let us get a geometric intuition: each of the n *constraints* in (i) requires $x = (x_1, x_2, \ldots, x_d) \in \mathbb{R}^d$ to be in the positive halfspace of some hyperplane. The intersection of all these halfspaces is the *feasible region* of the linear program. If it is empty, there is no solution—the linear program is called *infeasible*.

Otherwise—and now (ii) enters the picture—we are looking for a *feasible solution* x (a point inside the feasible region) that maximizes $\sum_{j=1}^{d} c_j x_j$. For every possible value γ of this sum, the feasible solutions for which the sum attains this value are contained in the hyperplane

$$\{x \in \mathbb{R}^d : \sum_{j=1}^{d} c_j x_j = \gamma\}$$

with normal vector $c = (c_1, \ldots, c_d)$. Increasing γ means to shift the hyperplane into direction c. The highest γ is thus obtained from the hyperplane that is most extreme in direction c among all hyperplanes that intersect the feasible region, see Figure E.3.

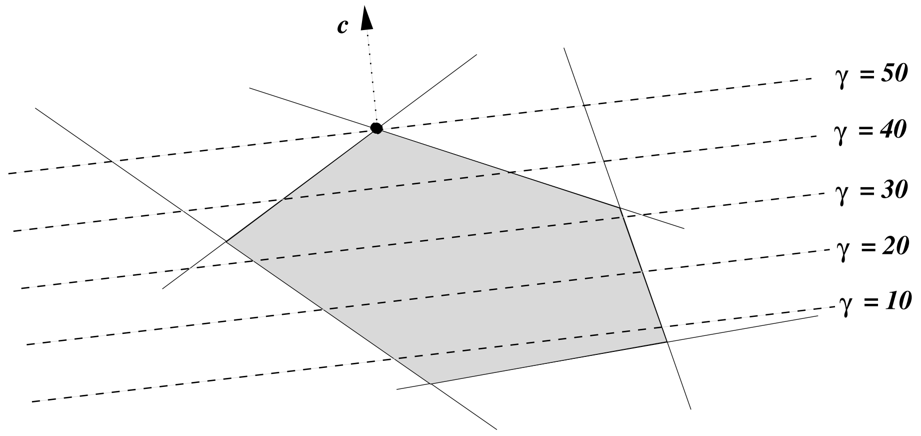


Figure E.3: *A linear program: finding the feasible solution in the intersection of five positive halfspaces that is most extreme in direction* c *(has highest value* $\gamma = \sum_{j=1}^{d} c_j x_j$*)*

In Figure E.3, we do have an *optimal solution* (a feasible solution x of highest value $\sum_{j=1}^{d} c_j x_j$), but in general, there might be feasible solutions of arbitrarily high γ-value. In this case, the linear program is called *unbounded*, see Figure E.4.

It can be shown that infeasibility and unboundedness are the only obstacles for the existence of an optimal solution. If the linear program is feasible and bounded, there exists an optimal solution.

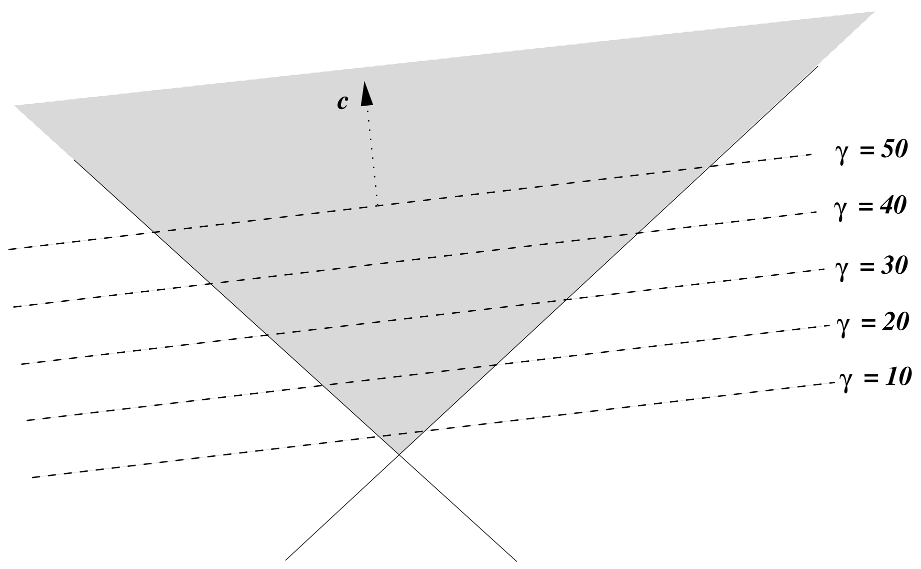


Figure E.4: *An unbounded linear program*

This is not entirely trivial, though. To appreciate the statement, consider the problem of finding a point (x, y) that (i) satisfies $y \geqslant e^x$ and (ii) has smallest value of y among all (x, y) that satisfy (i). This is not a linear program, but in the above sense it is feasible (there are (x, y) that satisfy (i)) and bounded (y is bounded below from 0 over the set of feasible solutions). Still, there is no optimal solution, since values of y arbitrarily close to 0 can be attained but not 0 itself.

Even if a linear program has an optimal solution, it is in general not unique. For example, if you rotate c in Figure E.3 such that it becomes orthogonal to the top-right edge of the feasible region, then every point of this edge is an optimal solution. Why is this called a *linear* program? Because all constraints are linear inequalities, and the objective function is a linear function. There is also a reason why it is called a linear *program*, but we won't get into this here (see [3] for more background).

Using vector and matrix notation, a linear program can succinctly be written as follows.

$$\begin{array}{lll} \text{(LP)} & \text{maximize} & c^\top x \\ & \text{subject to} & Ax \leqslant b \end{array}$$

Here, $c, x \in \mathbb{R}^d, b \in \mathbb{R}^n, A \in \mathbb{R}^{n \times d}$, and $\cdot^\top$ denotes the transpose operation. The inequality "$\leqslant$" is interpreted componentwise. The vector x represents the *variables*, c is called the *objective function vector*, b the *right-hand side*, and A the *constraint matrix.*

To *solve* a linear programs means to either report that the problem is infeasible or unbounded, or to compute an optimal solution x^*. If we can solve linear programs, we can also decide linear separability of point sets. For this, we check whether the linear program

$$\begin{array}{ll} \text{maximize} & 0 \\ \text{subject to} & h_1p_1 + h_2p_2 + \cdots + h_dp_d - h_0 \leqslant 0, \quad p \in P, \\ & h_1q_1 + h_2q_2 + \cdots + h_dq_d - h_0 \geqslant 0, \quad q \in Q. \end{array}$$

in the $d+1$ variables $h_0, h_1, h_2, \ldots, h_d$ and objective function vector $c = 0$ is feasible or not. The fact that some inequalities are of the form "$\geqslant$" is no problem, of course, since we can multiply an inequality by -1 to turn "$\geqslant$" into "$\leqslant$".

E.3 Minimum-area Enclosing Annulus

Here is another geometric problem that we can write as a linear program, although this is less obvious. Given a point set $P \subseteq \mathbb{R}^2$, find a *minimum-area annulus* (region between two concentric circles) that contains P; see Figure E.5 for an illustration.

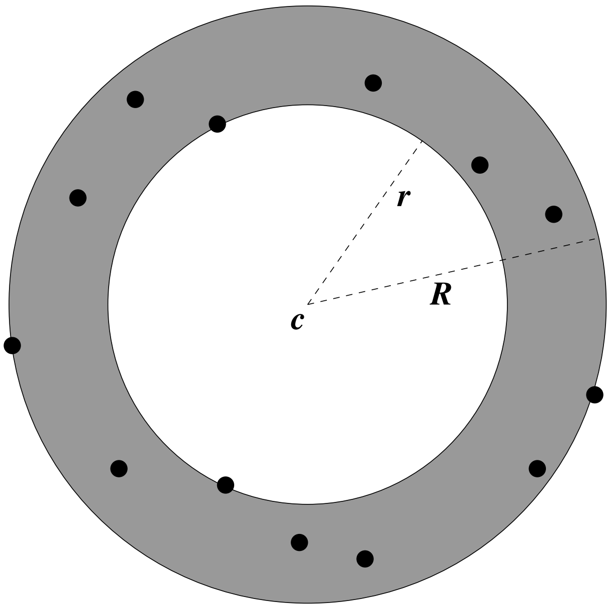


Figure E.5: *A minimum-area annulus containing* P

The optimal annulus can be used to test whether the point set P is (approximately) on a common circle which is the case if the annulus has zero (or small) area.

Let us write this as an optimization problem in the variables $c = (c_1, c_2) \in \mathbb{R}^2$ (the center) and $r, R \in \mathbb{R}$ (the small and the large radius).

$$\begin{array}{ll} \text{minimize} & \pi(R^2 - r^2) \\ \text{subject to} & r^2 \leqslant \|p - c\|^2 \leqslant R^2, \quad p \in P. \end{array}$$

This neither has a linear objective function nor are the constraints linear inequalities. But a variable substitution will take care of this. We define new variables

$$u := r^2 - \|c\|^2, \tag{E.3}$$
$$v := R^2 - \|c\|^2. \tag{E.4}$$

Omitting the factor π in the objective function does not affect the optimal solution (only its value), hence we can equivalently work with the objective function $v - u = R^2 - r^2$. The constraint $r^2 \leqslant \|p - c\|^2$ is equivalent to $r^2 \leqslant \|p\|^2 - 2p^\top c + \|c\|^2$, or

$$u + 2p^\top c \leqslant \|p\|^2.$$

In the same way, $\|p - c\| \leqslant R$ turns out to be equivalent to

$$v + 2p^\top c \geqslant \|p\|^2.$$

This means, we now have a *linear* program in the variables u, v, c_1, c_2:

$$\begin{array}{ll} \text{maximize} & u - v \\ \text{subject to} & u + 2p^\top c \leqslant \|p\|^2, \quad p \in P \\ & v + 2p^\top c \geqslant \|p\|^2, \quad p \in P. \end{array}$$

From optimal values for u, v and c, we can also reconstruct r^2 and R^2 via (E.3) and (E.4). It cannot happen that r^2 obtained in this way is negative: since we have $r^2 \leqslant \|p - c\|^2$ for all p, we could still increase u (and hence r^2 to at least 0), which is a contradicition to $u - v$ being maximal.

Exercise E.5. *a) Prove that the problem of finding a largest disk inside a convex polygon can be formulated as a linear program! What is the number of variables in your linear program?*

b) Prove that the problem of testing whether a simple polygon is starshaped can be formulated as a linear program.

Exercise E.6. *Given a simple polygon* P *as a list of vertices along its boundary. Describe a linear time algorithm to decide whether* P *is star-shaped and—if so—to construct the so-called* kernel *of* P*, that is, the set of all star-points.*

E.4 Solving a Linear Program

Linear programming was first studied in the 1930's -1950's, and some of its original applications were of a military nature. In the 1950's, Dantzig invented the *simplex method* for solving linear programs, a method that is fast in practice but is not known to come with any theoretical guarantees [1].

The computational complexity of solving linear programs was unresolved until 1979 when Leonid Khachiyan discovered a polynomial-time algorithm known as the *ellipsoid method* [2]. This result even made it into the New York times.

From a computational geometry point of view, linear programs with a *fixed* number of variables are of particular interest (see our two applications above, with d and 4 variables, respectively, where d may be 2 or 3 in some relavant cases). As was first shown by Megiddo, such linear programs can be solved in time $O(n)$, where n is the number of constraints [4]. In the next lecture, we will describe a much simpler randomized $O(n)$ algorithm due to Seidel [5].

Questions

83. *What is a linear program?* Give a precise definition! How can you visualize a linear program? What does it mean that the linear program is infeasible / unbounded?
84. *Show an application of linear programming!* Describe a geometric problem that can be formulated as a linear program, and give that formulation!

References

[1] George B. Dantzig, *Linear programming and extensions*, Princeton University Press, Princeton, NJ, 1963.

[2] Leonid G. Khachiyan, Polynomial algorithms in linear programming. *U.S.S.R. Comput. Math. and Math. Phys*, **20**, (1980), 53–72.

[3] Jiří Matoušek and Bernd Gärtner, *Understanding and using linear programming*, Universitext, Springer, 2007.

[4] Nimrod Megiddo, Linear programming in linear time when the dimension is fixed. *J. ACM*, **31**, (1984), 114–127.

[5] Raimund Seidel, Small-dimensional linear programming and convex hulls made easy. *Discrete Comput. Geom.*, **6**, (1991), 423–434.

Appendix F

A randomized Algorithm for Linear Programming

Let us recall the setup from last lecture: we have a linear program of the form

$$\begin{array}{lll}(\mathrm{LP}) & \text{maximize} & c^\top x \\ & \text{subject to} & Ax \leqslant b,\end{array} \tag{F.1}$$

where $c, x \in \mathbb{R}^d$ (there are d variables), $b \in \mathbb{R}^n$ (there are n constraints), and $A \in \mathbb{R}^{n\times d}$. The scenario that we are interested in here is that d is a (small) constant, while n tends to infinity.

The goal of this lecture is to present a randomized algorithm (due to Seidel [2]) for solving a linear program whose expected runtime is $O(n)$. The constant behind the big-O will depend exponentially on d, meaning that this algorithm is practical only for small values of d.

To prepare the ground, let us first get rid of the unboundedness issue. We add to our linear program a set of $2d$ constraints

$$-M \leqslant x_i \leqslant M, \quad i = 1, 2, \dots d, \tag{F.2}$$

where M is a symbolic constant assumed to be larger than any real number that it is compared with. Formally, this can be done by computing with rational functions in M (quotients of polynomials of degree d in the "variable "M), rather than real numbers. The original problem is bounded if and only if the solution of the new (and bounded) problem does not depend on M. This is called the *big-M method.*

Now let H, $|H| = n$, denote the set of original constraints. For $h \in H$, we write the corresponding constraint as $a_h x \leqslant b_h$.

Definition F.3. *For* $Q, R \subseteq H, Q \cap R = \emptyset$*, let* $x^*(Q,R)$ *denote the lexicographically largest optimal solution of the linear program*

$$\begin{array}{llll}\textit{LP(Q,R)} & \textit{maximize} & c^\top x & \\ & \textit{subject to} & a_h x \leqslant b_h, & h \in Q \\ & & a_h x = b_h, & h \in R \\ & & -M \leqslant x_i \leqslant M, & i = 1, 2, \dots, d.\end{array}$$

If this linear program has no feasible solution, we set $x^*(Q, R) = \infty$*.*

What does this mean? We delete some of the original constraints (the ones not in $Q \cup R$, and we require some of the constraints to hold with equality (the ones in R). Since every linear equation $a_h x = b_h$ can be simulated by two linear inequalities $a_h x \leqslant b_h$ and $a_h x \geqslant b_h$, this again assumes the form of a linear program. By the big-M method, this linear program is bounded, but it may be infeasible. If it is feasible, there may be several optimal solutions, but choosing the lexicographically largest one leads to a unique solution $x^*(Q, R)$.

Our algorithm will compute $x^*(H, \emptyset)$, the lexicographically largest optimal solution of (F.1) subject to the additional bounding-box constraint (F.2). We also assume that $x^*(H, \emptyset) \neq \infty$, meaning that (F.1) is feasible. At the expense of solving an auxiliary problem with one extra variable, this may be assumed w.l.o.g. (Exercise).

Exercise F.4. *Suppose that you have an algorithm $\mathcal{A}$ for solving* feasible *linear programs of the form*

$$\begin{array}{lll} \textit{(LP)} & \textit{maximize} & c^\top x \\ & \textit{subject to} & Ax \leqslant b, \end{array}$$

where feasible means that there exists $\tilde{x} \in \mathbb{R}^d$ such that $A\tilde{x} \leqslant b$. Extend algorithm $\mathcal{A}$ such that it can deal with arbitrary (not necessarily feasible) linear programs of the above form.

F.1 Helly's Theorem

A crucial ingredient of the algorithm's analysis is that the optimal solution $x^*(H, \emptyset)$ is already determined by a constant number (at most d) of constraints. More generally, the following holds.

Lemma F.5. *Let $Q, R \subseteq H, Q \cap R = \emptyset$, such that the constraints in R are* independent*. This means that the set $\{x \in \mathbb{R}^d : a_h x = b_h, h \in R\}$ has dimension $d - |R|$.*

If $x^*(Q, R) \neq \infty$, then there exists $S \subseteq Q, |S| \leqslant d - |R|$ such that

$$x^*(S, R) = x^*(Q, R).$$

The proof uses *Helly's Theorem*, a classic result in convexity theory.

Theorem F.6 (Helly's Theorem [1]). *Let $C_1, \ldots C_n$ be $n \geqslant d + 1$ convex subsets of $\mathbb{R}^d$. If any $d + 1$ of the sets have a nonempty common intersection, then the common intersection of* all *n sets is nonempty.*

Even in $\mathbb{R}^1$, this is not entirely obvious. There it says that for every set of intervals with pairwise nonempty overlap there is one point contained in all the intervals. We will not prove Helly's Theorem here but just use it to prove Lemma F.5.

Proof. (**Lemma F.5**) The statement is trivial for $|Q| \leqslant d - |R|$, so assume $|Q| > d - |R|$. Let

$$L(R) := \{x \in \mathbb{R}^d : a_h x = b_h, h \in R\}$$

and

$$B := \{x \in \mathbb{R}^d : -M \leqslant x_i \leqslant M, i = 1, \dots, d\}.$$

For a vector $x = (x_1, \dots, x_d)$, we define $x_0 = c^\top x$, and we write $x > x'$ if $(x_0, x_1, \dots, x_d)$ is lexicographically larger than $(x'_0, x'_1, \dots, x'_d)$.

Let $x^* = x^*(Q, R)$ and consider the $|Q| + 1$ sets

$$C_h = \{x \in \mathbb{R}^d : a_h x \leqslant b_h\} \cap B \cap L(R), \quad h \in Q$$

and

$$C_0 = \{x \in \mathbb{R}^d : x > x^*\} \cap B \cap L(R).$$

The first observation (that may require a little thinking in case of C_0) is that all these sets are convex. The second observation is that their common intersection is empty. Indeed, any point in the common intersection would be a feasible solution $\tilde{x}$ of $LP(Q, R)$ with $\tilde{x} > x^* = x^*(Q, R)$, a contradiction to $x^*(Q, R)$ being the lexicographically largest optimal solution of $LP(Q, R)$. The third observation is that since $L(R)$ has dimension $d - |R|$, we can after an affine transformation assume that all our $|Q| + 1$ convex sets are actually convex subsets of $\mathbb{R}^{d-|R|}$.

Then, applying Helly's Theorem yields a subset of $d - |R| + 1$ constraints with an empty common intersection. Since all the C_h do have $x^*(Q, R)$ in common, this set of constraints must contain C_0. This means, there is $S \subseteq Q, |S| = d - |R|$ such that

$$x \in C_h \ \forall h \in S \quad \Rightarrow \quad x \notin C_0.$$

In particular, $x^*(S, R) \in C_h$ for all $h \in S$, and so it follows that $x^*(S, R) \leqslant x^* = x^*(Q, R)$. But since $S \subseteq Q$, we also have $x^*(S, R) \geqslant x^*(Q, R)$, and $x^*(S, R) = x^*(Q, R)$ follows. □

F.2 Convexity, once more

We need a second property of linear programs on top of Lemma F.5; it is also a consequence of convexity of the constraints, but a much simpler one.

Lemma F.7. *Let* $Q, R \subseteq H, Q \cap R \neq \emptyset$ *and* $x^*(Q, R) \neq \infty$. *Let* $h \in Q$. *If*

$$a_h x^*(Q \setminus \{h\}, R) > b_h,$$

then

$$x^*(Q, R) = x^*(Q \setminus \{h\}, R \cup \{h\}).$$

Before we prove this, let us get an intuition. The vector $x^*(Q \setminus \{h\}, R)$ is the optimal solution of $\mathrm{LP}(Q \setminus \{h\}, R)$. And the inequality $a_h x^*(Q \setminus \{h\}, R) > b_h$ means that the constraint h is violated by this solution. The implication of the lemma is that at the optimal solution of $\mathrm{LP}(Q, R)$, the constraint h must be satisfied with equality in which case this optimal solution is at the same time the optimal solution of the more restricted problem $\mathrm{LP}(Q \setminus \{h\}, R \cup \{h\})$.

Proof. Let us suppose for a contradition that

$$a_h x^*(Q, R) < b_h$$

and consider the line segment s that connects $x^*(Q, R)$ with $x^*(Q \setminus \{h\}, R)$. By the previous strict inequality, we can make a small step (starting from $x^*(Q, R)$) along this line segment without violating the constraint h (Figure F.1). And since both $x^*(Q, R)$ as well as $x^*(Q \setminus \{h\}, R)$ satisfy all other constraints in $(Q \setminus \{h\}, R)$, convexity of the constraints implies that this small step takes us to a feasible solution of $\mathrm{LP}(Q, R)$ again. But this solution is lexicographically larger than $x^*(Q, R)$, since we move towards the lexicographically larger vector $x^*(Q \setminus \{h\}, R)$; this is a contradiction. □

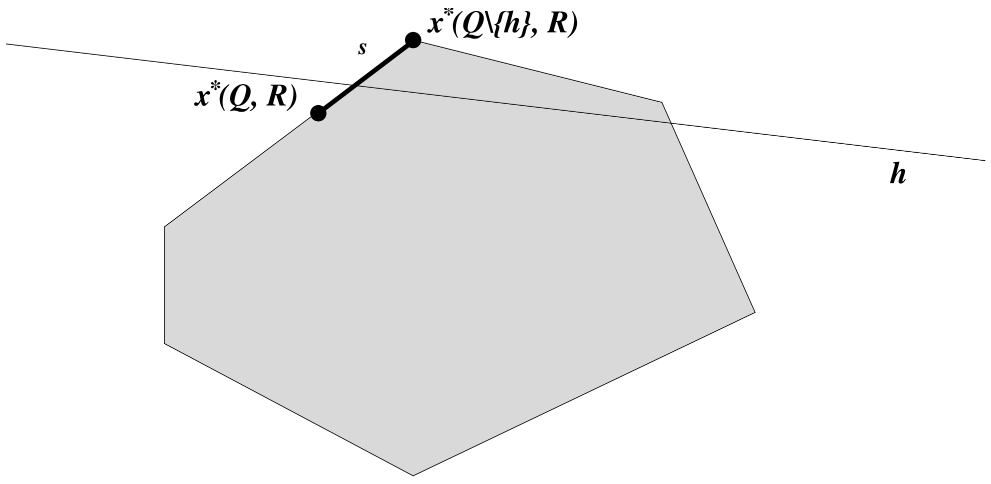


Figure F.1: *Proof of Lemma F.7*

F.3 The Algorithm

The algorithm reduces the computation of $x^*(H, \emptyset)$ to the computation of $x^*(Q, R)$ for various sets Q, R, where R is an *independent* set of constraints. Suppose you want to compute $x^*(Q, R)$ (assuming that $x^*(Q, R) \neq \infty$). If $Q = \emptyset$, this is "easy", since we have a constant-size problem, defined by R with $|R| \leqslant d$ and $2d$ bounding-box constraints $-M \leqslant x_i \leqslant M$.

Otherwise, we choose $h \in Q$ and recursively compute $x^*(Q \setminus \{h\}, R) \neq \infty$. We then check whether constraint h is violated by this solution. If not, we are done, since then $x^*(Q \setminus \{h\}, R) = x^*(Q, R)$ (Think about why!). But if h is violated, we can use Lemma F.7 to conclude that $x^*(Q, R) = x^*(Q \setminus \{h\}, R \cup \{h\})$, and we recursively compute the latter solution. Here is the complete pseudocode.

```
$\mathcal{LP}(Q, R)$:
    IF $Q = \emptyset$ THEN
        RETURN $x^*(\emptyset, R)$
    ELSE
        choose $h \in Q$ uniformly at random
        $x^* := \mathcal{LP}(Q \setminus \{h\}, R)$
        IF $a_h x^* \leqslant b_h$ THEN
            RETURN $x^*$
        ELSE
            RETURN $\mathcal{LP}(Q \setminus \{h\}, R \cup \{h\})$
        END
    END
```

To solve the original problem, we call this algorithm with $\mathcal{LP}(H, \emptyset)$. It is clear that the algorithm terminates since the first argument Q becomes smaller in every recursive call. It is also true (Exercise) that every set R that comes up during this algorithm is indeed an independent set of constraints and in particular has at most d elements. The correctness of the algorithm then follows from Lemma F.7.

Exercise F.8. *Prove that all sets* R *of constraints that arise during a call to algorithm* $\mathcal{LP}(H, \emptyset)$ *are independent, meaning that the set*

$$\{x \in \mathbb{R}^d : a_h x = b_h, h \in R\}$$

of points that satisfy all constraints in R *with equality has dimension* $d - |R|$.

F.4 Runtime Analysis

Now we get to the analysis of algorithm LP, and this will also reveal why the random choice is important.

We will analyze the algorithm in terms of the expected number of *violation tests* $a_h x^* \leqslant b_h$, and in terms of the expected number of *basis computations* $x^*(\emptyset, R)$ that it performs. This is a good measure, since these are the dominating operations of the algorithm. Moreover, both violation test and basis computation are "cheap" operations in the sense that they can be performed in time $f(d)$ for some f.

More specifically, a violation test can be performed in time $O(d)$; the time needed for a basis computation is less clear, since it amounts to solving a small linear program itself. Let us suppose that it is done by brute-force enumeration of all vertices of the bounded polyhedron defined by the at most $3d$ (in)equalities

$$a_h x = b_h, h \in R$$

and

$$-M \leqslant x_i \leqslant M, i = 1, \ldots, d.$$

F.4.1 Violation Tests

Lemma F.9. *Let $T(n, j)$ be the maximum expected number of violation tests performed in a call to $\mathcal{LP}(Q, R)$ with $|Q| = n$ and $|R| = d - j$. For all $j = 0, \ldots, d$,*

$$\begin{aligned} T(0, j) &= 0 \\ T(n, j) &\leqslant T(n-1, j) + 1 + \frac{j}{n} T(n-1, j-1), \quad n > 0. \end{aligned}$$

Note that in case of $j = 0$, we may get a negative argument on the right-hand side, but due to the factor $0/n$, this does not matter.

Proof. If $|Q| = \emptyset$, there is no violation test. Otherwise, we recursively call $\mathcal{LP}(Q \setminus \{h\}, R)$ for some h which requires at most $T(n-1, j)$ violation tests in expectation. Then there is one violation test within the call to $\mathcal{LP}(Q, R)$ itself, and depending on the outcome, we perform a second recursive call $\mathcal{LP}(Q \setminus \{h\}, R \cup \{h\})$ which requires an expected number of at most $T(n-1, j-1)$ violation tests. The crucial fact is that the probability of performing a second recursive call is at most j/n.

To see this, fix some $S \subseteq Q, |S| \leqslant d - |R| = j$ such that $x^*(Q, R) = x^*(S, R)$. Such a set S exists by Lemma F.5. This means, we can remove from Q all constraints except the ones in S, without changing the solution.

If $h \notin S$, we thus have

$$x^*(Q, R) = x^*(Q \setminus \{h\}, R),$$

meaning that we have already found $x^*(Q, R)$ after the first recursive call; in particular, we will then have $a_h x^* \leqslant b_h$, and there is no second recursive call. Only if $h \in S$ (and this happens with probability $|S|/n \leqslant j/n$), there can be a second recursive call. □

The following can easily be verified by induction.

Theorem F.10.

$$T(n, j) \leqslant \sum_{i=0}^{j} \frac{1}{i!} j! n.$$

Since $\sum_{i=0}^{j} \frac{1}{i!} \leqslant \sum_{i=0}^{\infty} \frac{1}{i!} = e$, we have $T(n, j) = O(j!n)$. If $d \geqslant j$ is constant, this is $O(n)$.

F.4.2 Basis Computations

To count the basis computations, we proceed as in Lemma F.9, except that the "1" now moves to a different place.

Lemma F.11. *Let* $B(n, j)$ *be the maximum expected number of basis computations performed in a call to* $\mathcal{LP}(Q, R)$ *with* $|Q| = n$ *and* $|R| = d - j$*. For all* $j = 0, \dots, d$*,*

$$\begin{aligned} B(0, j) &= 1 \\ B(n, j) &\leqslant B(n-1, j) + \frac{j}{n} B(n-1, j-1), \quad n > 0. \end{aligned}$$

Interestingly, $B(n, j)$ turns out to be much smaller than $T(n, j)$ which is good since a basic computation is much more expensive than a violation test. Here is the bound that we get.

Theorem F.12.

$$B(n, j) \leqslant (1 + H_n)^j = O(\log^j n),$$

where H_n *is the* n*-th Harmonic number.*

Proof. This is also a simple induction, but let's do this one since it is not entirely obvious. The statement holds for $n = 0$ with the convention that $H_0 = 0$. It also holds for $j = 0$, since Lemma F.11 implies $B(n, 0) = 1$ for all n. For $n, j > 0$, we inductively obtain

$$\begin{aligned} B(n, j) &\leqslant (1 + H_{n-1})^j + \frac{j}{n}(1 + H_{n-1})^{j-1} \\ &\leqslant \sum_{k=0}^{j} \binom{j}{k} (1 + H_{n-1})^{j-k} (\frac{1}{n})^k \\ &= (1 + H_{n-1} + \frac{1}{n})^j = (1 + H_n)^j. \end{aligned}$$

The second inequality uses the fact that the terms $(1 + H_{n-1})^j$ and $\frac{j}{n}(1 + H_{n-1})^{j-1}$ are the first two terms of the sum in the second line. □

F.4.3 The Overall Bound

Putting together Theorems F.10 and F.12 (for $j = d$, corresponding to the case $R = \emptyset$), we obtain the following

Theorem F.13. *A linear program with* n *constraints in* d *variables (*d *a constant) can be solved in time* $O(n)$*.*

Questions

85. *What is Helly's Theorem?* Give a precise statement and outline the application of the theorem for linear programming (Lemma F.5).

86. *Outline an algorithm for solving linear programs*! Sketch the main steps of the algorithm and the correctness proof! Also explain how one may achieve the preconditions of feasibility and boundedness.

87. *Sketch the analysis of the algorithm!* Explain on an intuitive level how randomization helps, and how the recurrence relations for the expected number of violation tests and basis computations are derived. What is the expected runtime of the algorithm?

References

[1] Herbert Edelsbrunner, *Algorithms in combinatorial geometry*, vol. 10 of *EATCS Monographs on Theoretical Computer Science*, Springer, 1987.

[2] Raimund Seidel, Small-dimensional linear programming and convex hulls made easy. *Discrete Comput. Geom.*, **6**, (1991), 423–434.

Appendix G

Smallest Enclosing Balls

This problem is related to the linear programming problem, but in a way it is much simpler, since a unique optimal solution always exists.

We let P be a set of n points in $\mathbb{R}^d$. We are interested in finding a closed ball of smallest radius that contains all the points in P, see Figure G.1.

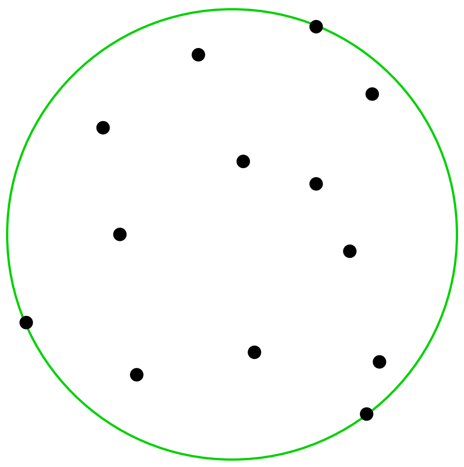


Figure G.1: *The smallest enclosing ball of a set of points in the plane*

As an “application”, imagine a village that wants to build a firehouse. The location of the firehouse should be such that the maximum travel time to any house of the village is as small as possible. If we equate travel time with Euclidean distance, the solution is to place the firehouse in the center of the smallest ball that covers all houses.

Existence It is not a priori clear that a smallest ball enclosing P exists, but this follows from standard arguments in calculus. As you usually don’t find this worked out in papers and textbooks, let us quickly do the argument here.

Fix P and consider the continuous function $\rho : \mathbb{R}^d \to \mathbb{R}$ defined by

$$\rho(c) = \max_{p \in P} \|p - c\|, c \in \mathbb{R}^d$$

Thus, $\rho(c)$ is the radius of the smallest ball centered at c that encloses all points of P. Let q be any point of P, and consider the closed ball

$$B = B(q, \rho(q)) := \{c \in \mathbb{R}^2 \mid \|c - q\| \leqslant \rho(q)\}.$$

Since B is compact, the function ρ attains its minimum over B at some point c_{opt}, and we claim that c_{opt} is the center of a smallest enclosing ball of P. For this, consider any center $c \in \mathbb{R}^2$. If $c \in B$, we have $\rho(c) \geqslant \rho(c_{\text{opt}})$ by optimality of c_{opt} in B, and if $c \notin B$, we get $\rho(c) \geqslant \|c - q\| > \rho(q) \geqslant \rho(c_{\text{opt}})$ since $q \in B$. In any case, we get $\rho(c) \geqslant \rho(c_{\text{opt}})$, so c_{opt} is indeed a best possible center.

Uniqueness Can it be that there are two distinct smallest enclosing balls of P? No, and to rule this out, we use the concept of *convex combinations* of balls. Let $B = B(c, \rho)$ be a closed ball with center c and radius $\rho > 0$. We define the *characteristic function* of B as the function $f_B : \mathbb{R}^2 \to R$ given by

$$f_B(x) = \frac{\|x - c\|^2}{\rho^2}, \quad x \in \mathbb{R}^2.$$

The name characteristic function comes from the following easy

Observation G.1. *For $x \in \mathbb{R}^2$, we have*

$$x \in B \quad \Leftrightarrow \quad f_B(x) \leqslant 1.$$

Now we are prepared for the convex combination of balls.

Lemma G.2. *Let $B_0 = B(c_0, \rho_0)$ and $B_1 = (c_1, \rho_1)$ be two distinct balls with characteristic functions f_{B_0} and f_{B_1}. For $\lambda \in (0, 1)$, consider the function f_λ defined by*

$$f_\lambda(x) = (1 - \lambda) f_{B_0}(x) + \lambda f_{B_1}(x).$$

Then the following three properties hold.

(i) f_λ *is the characteristic function of a ball* $B_\lambda = (c_\lambda, \rho_\lambda)$. B_λ *is called a (proper)* convex combination *of* B_0 *and* B_1*, and we simply write*

$$B_\lambda = (1 - \lambda)B_0 + \lambda B_1.$$

(ii) $B_\lambda \supseteq B_0 \cap B_1$ *and* $\partial B_\lambda \supseteq \partial B_0 \cap \partial B_1$.

(iii) $\rho_\lambda < \max(\rho_0, \rho_1)$.

A proof of this lemma requires only elementary calculations and can be found for example in the PhD thesis of Kaspar Fischer [3]. Here we will just explain what the lemma means. The family of balls $B_\lambda, \lambda \in (0,1)$ "interpolates" between the balls B_0 and B_1: while we increase λ from 0 to 1, we continuously transform B_0 into B_1. All intermediate balls B_λ "go through" the intersection of the original ball boundaries (a sphere of dimension $d-2$). In addition, each intermediate ball contains the intersection of the original balls. This is property (ii). Property (iii) means that all intermediate balls are smaller than the larger of B_0 and B_1. Figure G.2 illustrates the situation.

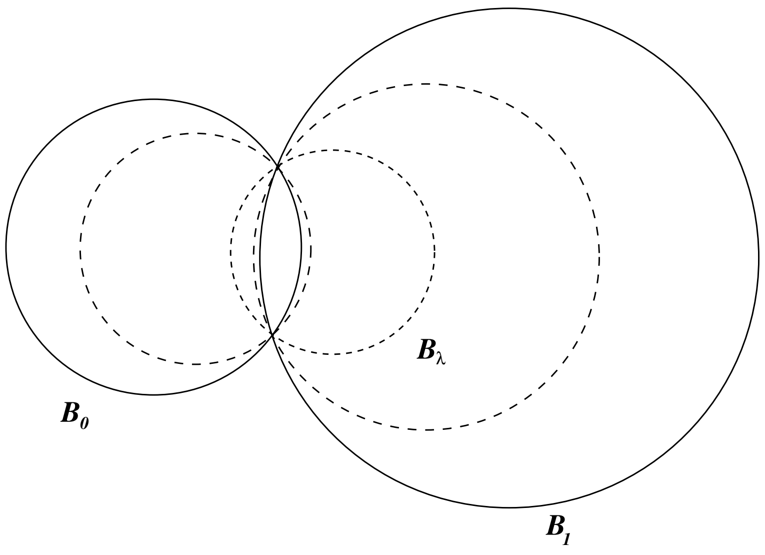


Figure G.2: *Convex combinations* B_λ *of two balls* B_0, B_1

Using this lemma, we can easily prove the following

Theorem G.3. *Given a finite point set* $P \subseteq \mathbb{R}^d$*, there exists a* unique *ball of smallest radius that contains* P*. We will denote this ball by* $B(P)$*.*

Proof. If $P = \{p\}$, the unique smallest enclosing ball is $\{p\}$. Otherwise, any smallest enclosing ball of P has positive radius ρ_{opt}. Assume there are two distinct smallest enclosing balls B_0, B_1. By Lemma G.2, the ball

$$B_{\frac{1}{2}} = \frac{1}{2}B_0 + \frac{1}{2}B_1$$

is also an enclosing ball of P (by property (ii)), but it has smaller radius than ρ_{opt} (by property (iii), a contradiction to B_0, B_1 being smallest enclosing balls. □

Bases When you look at the example of Figure G.1, you notice that only three points are essential for the solution, namely the ones on the boundary of the smallest enclosing ball. Removing all other points from P would not change the smallest enclosing ball. Even in cases where more points are on the boundary, it is always possible to find a subset of at most three points (in the $\mathbb{R}^2$ case) with the same smallest enclosing ball. This is again a consequence of *Helly's Theorem* (Theorem 4.9).

Theorem G.4. *Let $P \subseteq \mathbb{R}^d$ be a finite point set. There is a subset $S \subseteq P, |S| \leqslant d+1$ such that $B(P) = B(S)$.*

Proof. If $|P| < d+1$, we may choose $S = P$. Otherwise, let ρ_{opt} be the radius of the smallest enclosing ball $B(P)$ of $P = \{p_1, \ldots, p_n\}$. Now define

$$C_i = \{x \in \mathbb{R}^d : \|x - p_i\| < \rho_{opt}\}, \quad i = 1, \ldots, n$$

to be the open ball around p_i with radius ρ_{opt}. We know that the common intersection of all the C_i is empty, since any point in the intersection would be a center of an enclosing ball of P with radius smaller than ρ_{opt}. Moreover, the C_i are convex, so Helly's Theorem implies that there is a subset S of $d+1$ points whose C_i's also have an empty common intersection. For this set S, we therefore have no enclosing ball of radius smaller than ρ_{opt} either. Hence, $B(S)$ has radius at leat ρ_{opt}; but since $S \subseteq P$, the radius of $B(S)$ must also be at most ρ_{opt}, and hence it is equal to ρ_{opt}. But then $B(S) = B(P)$ follows, since otherwise, both $B(P)$ and $B(S)$ would be smallest enclosing balls of S, a contradiction. □

The previous theorem motivates the following

Definition G.5. *Let $P \subseteq \mathbb{R}^d$ be a finite point set. A* basis *of P is an inclusion-minimal subset $S \subseteq P$ such that $B(P) = B(S)$.*

It follows that any basis of P has size at most $d+1$. If the points are in general position (no $k+3$ on a common k-dimensional sphere), then P has a unique basis, and this basis is formed by the set of points on the boundary of $B(P)$.

G.1 The trivial algorithm

Theorem G.4 immediately implies the following (rather inefficient) algorithm for computing $B(P)$: for every subset $S \subseteq P, |S| \leqslant d+1$, compute $B(S)$ (in fixed dimension d, this can be done in constant time), and return the one with largest radius.

Indeed, this works: for all $S \subseteq P$, the radius of $B(S)$ is at most that of $B(P)$, and there must be at least one $S, |S| \leqslant d+1$ (a basis of P) with $B(S) = B(P)$. It follows that the ball $B(T)$ being returned has the same radius as $B(P)$ and is therefore equal by $T \subseteq P$.

Assuming that d is fixed, the runtime of this algorithm is

$$O\left(\sum_{i=0}^{d+1} \binom{n}{i}\right) = O(n^{d+1}).$$

If $d = 2$ (the planar case), the trivial algorithm has runtime $O(n^3)$. In the next section, we discuss an algorithm that is substantially better than the trivial one in any dimension.

In order to adapt Seidel's randomized linear programming algorithm to the problem of computing smallest enclosing balls, we need the following statements.

Exercise G.6. *(i) Let $P, R \subseteq \mathbb{R}^d, P \cap R = \emptyset$. If there exists a ball that contains P and has R on the boundary, then there is also a unique smallest such ball which we denote by $B(P, R)$.*

(ii) Let $P, R \subseteq \mathbb{R}^d, P \cap R = \emptyset$. If $B(P, R)$ exists and $p \in P$ satisfies $p \notin B(P \setminus \{p\}, R)$, then p is on the boundary of $B(P, R)$, meaning that $B(P, R) = B(P \setminus \{p\}, R \cup \{p\})$.

Prove these two statements!

G.2 Welzl's Algorithm

The idea of this algorithm is the following. Given $P \subseteq \mathbb{R}^d$, the algorithm first recursively computes $B(P \setminus \{p\})$ where $p \in P$ is chosen uniformly at random. Then there are two cases: if $p \in B(P \setminus \{p\})$ we have $B(P) = B(P \setminus \{p\})$ (it's always good to rethink why this holds) and so we are done already. If $p \notin B(P \setminus \{p\})$, we still need to work, but the key fact (that we prove below) is that in this case, p has to be on the *boundary* of $B(P)$. We can therefore recursively compute the smallest enclosing ball of $P \setminus \{p\}$ that has p on its boundary, and this is a simpler problem because it intuitively has one degree of freedom less.

Let us formalize this idea.

Definition G.7. *Let $P, R \subseteq \mathbb{R}^d$ be disjoint finite point sets. We define $B(P, R)$ as the smallest ball that contains P and has the points of R on its boundary (*if *this ball exists and is unique).*

It is not hard to see that existence cannot always be guaranteed; for example if R is contained in the convex hull of P, there can be no ball that contains P and has even a single point of R on its boundary. But the following Lemma gives a number of useful properties.

Lemma G.8. *Let $P, R \subseteq \mathbb{R}^d$ be disjoint finite point sets, where R is affinely independent.*

(i) If there is any ball that contains P and has R on its boundary, then a unique smallest such ball $B(P, R)$ exists.

(ii) If $B(P, R)$ exists and $p \notin B(P \setminus \{p\}, R)$, then p is on the boundary of $B(P, R)$, meaning that $B(P, R) = B(P \setminus \{p\}, R \cup \{p\})$.

(iii) If $B(P, R)$ exists, there is a subset $S \subseteq P$ of size $|S| \leqslant d + 1 - |R|$ such that $B(P, R) = B(S, R)$.

Before we can prove this, we need a helper lemma.

Lemma G.9. *Let* $R \subseteq \mathbb{R}^d, |R| \geqslant 1$ *be an affinely independent point set. Then the set*

$$C(R) := \{c \in \mathbb{R}^d \mid c \text{ is the center of a ball that has } R \text{ on its boundary}\} \qquad (G.10)$$

is a linear subspace of dimension $d+1-|R|$.

Proof. We have $c \in C(R)$ if and only if there exists a number ρ^2 such that

$$\rho^2 = \|c-p\|^2 = c^Tc - 2c^Tp + p^Tp, \quad p \in R. \qquad (G.11)$$

Defining $\mu = \rho^2 - c^Tc$, this implies

$$\mu = p^Tp - 2c^Tp, \quad p \in R. \qquad (G.12)$$

The set of all (c, μ) satisfying the latter $|R|$ equations is a linear subspace L of R^{d+1}.

Claim. L has dimension $d+1-|R|$.

To see this, let us write (G.12) in matrix form as follows.

$$\begin{pmatrix} p_{11} & p_{12} & \cdots & p_{1d} & 1 \\ p_{21} & p_{22} & \cdots & p_{2d} & 1 \\ \vdots & \vdots & & \vdots & \vdots \\ p_{|R|1} & p_{|R|2} & \cdots & p_{|R|d} & 1 \end{pmatrix} \begin{pmatrix} 2c_1 \\ 2c_2 \\ \vdots \\ 2c_d \\ \mu \end{pmatrix} = \begin{pmatrix} p_1^Tp_1 \\ p_2^Tp_2 \\ \cdots \\ p_{|R|}^Tp_{|R|} \end{pmatrix}, \qquad (G.13)$$

where $R = p_1, \dots, p_{|R|}$. We know from linear algebra that the dimension of the solution space L is $d+1$ minus the rank of the matrix. But this rank is $|R|$ since the rows are linearly independent by affine independence of R (we leave this easy argument to the reader, also as a good exercise to recall the definition of affine independence).

It only remains to show that $C(R)$ is also a linear subspace, and of the same dimension as L. But this holds since the linear function $f : C(R) \to L$ given by

$$f(c) = (c, p_1^Tp_1 - 2c^Tp_1)$$

is a bijection between $C(R)$ and L. The function is clearly injective, but it is also surjective: if $(c, \mu) \in L$, we satisfy (G.11) with $\rho^2 := \mu + c^Tc$, meaning that $c \in C(R)$ and

$$f(c) = (c, p_1^Tp_1 - 2c^Tp_1) = (c, \|c-p_1\|^2 - c^Tc) = (c, \rho^2 - c^Tc) = (c, \mu).$$

□

Now we can proceed with the proof of Lemma G.8.

Proof. The proof of (i) works along the same lines as the one for $B(P)$ in Section G, and it differs only for $R \neq \emptyset$. In this case, choose $s \in R$ arbitrarily.

Let $C(P, R)$ be the set of all $c \in \mathbb{R}^d$ that are centers of balls containing P, and with R on the boundary. Note that $C(P, R)$ is closed, since it results from intersecting the linear space $C(R)$ with the closed set

$$\{c \in \mathbb{R}^d \mid \max_{p \in P} \|c - p\| \leqslant \max_{p \in R} \|p - c\|\}.$$

By assumption, the set $C(P, R)$ is nonempty. We define

$$\rho(c) = \|s - c\|, \quad c \in C(P, R).$$

Thus, $\rho(c)$ is the radius of the *unique* ball centered at c that encloses all points of P and has all points of R on the boundary. To prove that there is some smallest ball containing P and with R on the boundary, we need to show that the continuous function ρ attains a minimum over $C(P, R)$. To be able to restrict attention to a closed and bounded (hence compact) subset of $C(P, R)$, we choose some $c_0 \in C(P, R)$; then $\rho(c_0)$ is certainly an upper bound for the minimum value, meaning that any $c \in C(P, R)$ outside the compact set

$$\{c \in C(P, R) \mid \rho(c) \leqslant \rho(c_0)\}$$

canot be a candidate for the center of a smallest ball. Within this compact set, we do get a minimum c_{opt}, and this is the desired center of a smallest enclosing ball of P that has R on the boundary.

To prove uniquenes, we invoke again the convex combination of balls: Assuming that there are two smallest balls B_0, B_1, then the ball

$$\frac{1}{2}B_0 + \frac{1}{2}B_1$$

is a smaller ball that still contains P and still has R on its boundary (Lemma G.2 (ii)), a contradiction.

Now for part (ii). Again, convex combinations of balls come to our help. Consider the two balls $B(P, R)$ and $B(P \setminus \{p\}, R)$ (note that existence of the former implies existence of the latter via part (i)). If p is not on the boundary of $B(P, R)$, we have the situation of Figure G.3.

Then there is some small $\varepsilon > 0$ such that the ball

$$(1 - \varepsilon)B(P, R) + \varepsilon B(P \setminus \{p\}, R)$$

(drawn dashed in Figure G.3) still contains P and has R on the boundary, but has smaller volume than $B(P, R)$ by Lemma G.2 (iii), a contradiction.

Now we turn to (iii).

□

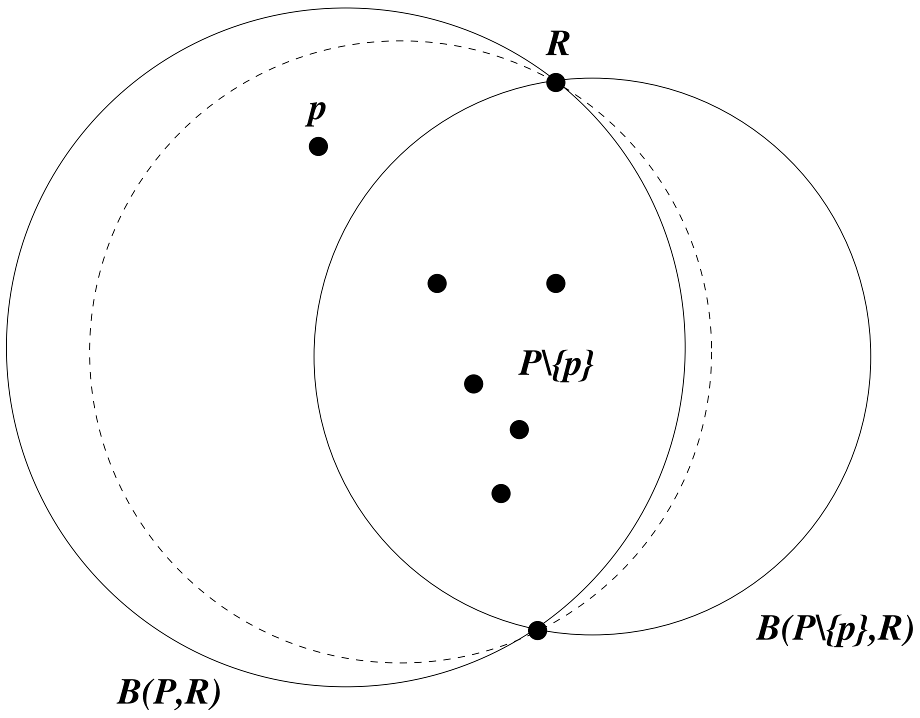


Figure G.3: *Proof of Lemma G.8(ii)*

G.3 The Swiss Algorithm

The name of this algorithm comes from the democratic way in which it works. Let us describe it for the problem of locating the firehouse in a village.

Here is how it is done the Swiss way: a meeting of all n house owners is scheduled, and every house owner is asked to put a ballot of paper with his/her name on it into a voting box. Then a constant number r (to be determined later) of ballots is drawn at random from the voting box, and the selected house owners have the right to negotiate a location for the firehouse among them. They naturally do this in a selfish way, meaning that they agree on the center of the smallest enclosing ball D of just *their* houses as the proposed location.

The house owners that were not in the selected group now fall into two classes: those that are happy with the proposal, and those that are not. Let's say that a house owner p is happy if and only if his/her house is also covered by D. In other words, p is happy if and only if the proposal would have been the same with p as an additional member of the selected group.

Now, the essence of Swiss democracy is to negotiate until everybody is happy, so as long as there are any unhappy house owners at all, the whole process is repeated. But in order to give the unhappy house owners a higher chance of influencing the outcome of the next round, their ballots in the voting box are being doubled before drawing r ballots again. Thus, there are now two ballots for each unhappy house owner, and one for each happy one.

After round k, a house owner that has been unhappy after exactly i of the k rounds has therefore 2^i ballots in the voting box for the next round.

The obvious question is: how many rounds does it take until all house owners are happy? So far, it is not even clear that the meeting ever ends. But Swiss democracy is efficient, and we will see that the meeting actually ends after an expected number of $O(\log n)$ rounds. We will do the analysis for general dimension d (just imagine the village and its houses to lie in $\mathbb{R}^d$).

G.4 The Forever Swiss Algorithm

In the analysis, we want to argue about a fixed round k, but the above algorithm may never get to this round (for large k, we even strongly hope that it never gets there). But for the purpose of the analysis, we formally let the algorithm continue even if everybody is happy after some round (in such a round, no ballots are being doubled).

We call this extension the Forever Swiss Algorithm. A round is called *controversial* if it renders at least one house owner unhappy.

Definition G.14.

(i) Let m_k be the random variable for the total number of ballots after round k of the Forever Swiss Algorithm. We set $m_0 = n$, the initial number of ballots.

(ii) Let C_k be the event that the first k rounds in the Forever Swiss Algorithm are controversial.

A lower bound for $E(m_k)$ Let $S \subseteq P$ be a basis of P. Recall that this means that S is inclusion-minimal with $B(S) = B(P)$.

Observation G.15. *After every controversial round, there is an unhappy house owner in S.*

Proof. Let Q be the set of selected house owners in the round. Let us write $B \geqslant B'$ for two balls if the radius of B is at least the radius of B'.

If all house owners in S were happy with the outcome of the round, we would have

$$B(Q) = B(Q \cup S) \geqslant B(S) = B(P) \geqslant B(Q),$$

where the inequalities follow from the corresponding superset relations. The whole chain of inequalities would then imply that $B(P)$ and $B(Q)$ have the same radius, meaning that they must be equal (we had this argument before). But then, *nobody* would be unhappy with the round, a contradiction to the current round being controversial. □

Since $|S| \leqslant d + 1$ by Theorem G.4, we know that after k rounds, some element of S must have doubled its ballots at least $k/(d + 1)$ times, given that all these rounds were controversial. This implies the following lower bound on the total number m_k of ballots.

Lemma G.16.

$$E(m_k) \geqslant 2^{k/(d+1)} \operatorname{prob}(C_k), \quad k \geqslant 0.$$

Proof. By the partition theorem of conditional expectation, we have

$$E(m_k) = E(m_k \mid C_k) \operatorname{prob}(C_k) + E(m_k \mid \overline{C_k}) \operatorname{prob}(\overline{C_k}) \geqslant 2^{k/(d+1)} \operatorname{prob}(C_k).$$

□

An upper bound for $E(m_k)$ The main step is to show that the expected increase in the number of ballots from one round to the next is bounded.

Lemma G.17. *For all* $m \in N$ *and* $k > 0$,

$$E(m_k \mid m_{k-1} = m) \leqslant m\left(1 + \frac{d+1}{r}\right).$$

Proof. Since exactly the "unhappy ballots" are being doubled, the expected increase in the total number of ballots equals the expected number of unhappy ballots, and this number is

$$\frac{1}{\binom{m}{r}} \sum_{|R|=r} \sum_{h \notin R} [h \text{ is unhappy with } R] = \frac{1}{\binom{m}{r}} \sum_{|Q|=r+1} \sum_{h \in Q} [h \text{ is unhappy with } Q \setminus \{h\}]. \tag{G.18}$$

Claim: Every $(r+1)$-element subset Q contains at most $d+1$ ballots such that h is unhappy with $Q \setminus \{h\}$.

To see the claim, choose a basis S, $|S| \leqslant d+1$, of the ball resulting from drawing ballots in Q. Only the removal of a ballot h belonging to some house owner $p \in S$ can have the effect that Q and $Q \setminus \{h\}$ lead to different balls. Moreover, in order for this to happen, the ballot h must be the *only* ballot of the owner p. This means that at most one ballot h per owner $p \in S$ can cause h to be unhappy with $Q \setminus \{h\}$.

We thus get

$$\frac{1}{\binom{m}{r}} \sum_{|R|=r} \sum_{h \notin R} [h \text{ is unhappy with } R] \leqslant (d+1)\frac{\binom{m}{r+1}}{\binom{m}{r}} = (d+1)\frac{m-r}{r+1} \leqslant (d+1)\frac{m}{r}. \tag{G.19}$$

By adding m, we get the new expected total number $E(m_k \mid m_{k-1} = m)$ of ballots. □

From this, we easily get our actual upper bound on $E(m_k)$.

Lemma G.20.

$$E(m_k) \leqslant n\left(1 + \frac{d+1}{r}\right)^k, \quad k \geqslant 0.$$

Proof. We use induction, where the case $k = 0$ follows from $m_0 = n$. For $k > 0$, the partition theorem of conditional expectation gives us

$$\begin{aligned} E(m_k) &= \sum_{m \geqslant 0} E(m_k \mid m_{k-1} = m) \operatorname{prob}(m_{k-1} = m) \\ &\leqslant \left(1 + \frac{d+1}{r}\right) \sum_{m \geqslant 0} m \operatorname{prob}(m_{k-1} = m) \\ &= \left(1 + \frac{d+1}{r}\right) E(m_{k-1}). \end{aligned}$$

Applying the induction hypothesis to $E(m_{k-1})$, the lemma follows. □

Putting it together Combining Lemmas G.16 and G.20, we know that

$$2^{k/(d+1)} \operatorname{prob}(C_k) \leqslant n \left(1 + \frac{d+1}{r}\right)^k,$$

where C_k is the event that there are k or more controversial rounds.

This inequality gives us a useful upper bound on $\operatorname{prob}(C_k)$, because the left-hand side power grows faster than the right-hand side power as a function of k, given that r is chosen large enough.

Let us choose $r = c(d+1)^2$ for some constant $c > \log_2 e \approx 1.44$. We obtain

$$\operatorname{prob}(C_k) \leqslant n \left(1 + \frac{1}{c(d+1)}\right)^k / 2^{k/(d+1)} \leqslant n 2^{k \log_2 e/(c(d+1)) - k/(d+1)},$$

using $1 + x \leqslant e^x = 2^{x \log_2 e}$ for all x. This further gives us

$$\operatorname{prob}(C_k) \leqslant n\alpha^k, \tag{G.21}$$

$$\alpha = \alpha(d, c) = 2^{(\log_2 e - c)/c(d+1)} < 1.$$

This implies the following tail estimate.

Lemma G.22. *For any $\beta > 1$, the probability that the Forever Swiss Algorithm performs at least $\lceil \beta \log_{1/\alpha} n \rceil$ controversial rounds is at most*

$$1/n^{\beta - 1}.$$

Proof. The probability for at least this many controversial rounds is at most

$$\operatorname{prob}(C_{\lceil \beta \log_{1/\alpha} n \rceil}) \leqslant n\alpha^{\lceil \beta \log_{1/\alpha} n \rceil} \leqslant n\alpha^{\beta \log_{1/\alpha} n} = n n^{-\beta} = 1/n^{\beta - 1}.$$

□

In a similar way, we can also bound the expected number of controversial rounds of the Forever Swiss Algorithm. This also bounds the expected number of rounds of the Swiss Algorithm, because the latter terminates upon the first non-controversial round.

Theorem G.23. *For any fixed dimension* d*, and with* $r = \lceil \log_2 e(d+1)^2 \rceil > \log_2 e(d+1)^2$*, the Swiss algorithm terminates after an expected number of* $O(\log n)$ *rounds.*

Proof. By definition of C_k (and using $E(X) = \sum_{m \geqslant 1} \text{prob}(X \geqslant m)$ for a random variable with values in $\mathbb{N}$), the expected number of rounds of the Swiss Algorithm is

$$\sum_{k \geqslant 1} \text{prob}(C_k).$$

For any $\beta > 1$, we can use (G.21) to bound this by

$$\begin{aligned}
\sum_{k=1}^{\lceil \beta \log_{1/\alpha} n \rceil - 1} 1 + n \sum_{k=\lceil \beta \log_{1/\alpha} n \rceil}^{\infty} \alpha^k &= \lceil \beta \log_{1/\alpha} n \rceil - 1 + n \frac{\alpha^{\lceil \beta \log_{1/\alpha} n \rceil}}{1-\alpha} \\
&\leqslant \beta \log_{1/\alpha} n + n \frac{\alpha^{\beta \log_{1/\alpha} n}}{1-\alpha} \\
&= \beta \log_{1/\alpha} n + \frac{n^{-\beta+1}}{1-\alpha} \\
&= \beta \log_{1/\alpha} n + o(1).
\end{aligned}$$

□

What does this mean for $d = 2$? In order to find the location of the firehouse efficiently (meaning in $O(\log n)$ rounds), 13 ballots should be drawn in each round. The resulting constant of proportionality in the $O(\log n)$ bound will be pretty high, though. To reduce the number of rounds, it may be advisable to choose r somewhat larger.

Since a single round can be performed in time $O(n)$ for fixed d, we can summarize our findings as follows.

Theorem G.24. *Using the Swiss Algorithm, the smallest enclosing ball of a set of* n *points in fixed dimension* d *can be computed in expected time* $O(n \log n)$*.*

The Swiss algorithm is a simplification of an algorithm by Clarkson [1, 2].

The bound of the previous Theorem already compares farovably with the bound of $O(n^{d+1})$ for the trivial algorithm, see Section G.1, but it does not stop here. We can even solve the problem in expected linear time $O(n)$, by using an adaptation of Seidel's linear programming algorithm [4].

Exercise G.25. *Let* H *be a set with* n *elements and* $f : 2^H \to \mathbb{R}$ *a function that maps subsets of* H *to real numbers. We say that* $h \in H$ violates $G \subseteq H$ *if* $f(G \cup \{h\}) \neq f(G)$ *(it follows that* $h \notin G$*). We also say that* $h \in H$ *is* extreme *in* G *if* $f(G \setminus \{h\}) \neq f(G)$ *(it follows that* $h \in G$*).*

Now we define two random variables $V_r, X_r : \binom{H}{r} \to \mathbb{R}$ *where* V_r *maps an* r*-element set* R *to the number of elements that violate* R*, and* X_r *maps an* r*-element set* R *to the number of extreme elements in* R*.*

Prove the following equality for $0 \leqslant r < n$*:*

$$\frac{E(V_r)}{n-r} = \frac{E(X_{r+1})}{r+1}.$$

Exercise G.26. *Imagine instead of doubling the ballots of the unhappy house owners in the Swiss Algorithm, we would multiply their number by some integer* $t \in \mathbb{N}$*. Does the analysis of the algorithm improve (i.e., does one get a better bound on the expected number of rounds, following the same approach)?*

Exercise G.27. *We have shown that for* $d = 2$ *and sample size* $r = 13$*, the Swiss algorithm takes an expected number of* $O(\log n)$ *rounds. Compute the constants, i.e., find numbers* c_1, c_2 *such that the expected number of rounds is always bounded by* $c_1 \log_2 n + c_2$*. Try to make* c_1 *as small as possible.*

G.5 Smallest Enclosing Balls in the Manhattan Distance

We can also compute smallest enclosing balls w.r.t. distances other than the Euclidean distance. In general, if $\delta : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is a metric, the smallest enclosing ball problem with respect to δ is the following.

Given $P \subseteq \mathbb{R}^d$, find $c \in \mathbb{R}^d$ and $\rho \in \mathbb{R}$ such that

$$d(c,p) \leqslant \rho, \quad p \in P,$$

and ρ is as small as possible.

For example, if $d(x,y) = \|x-y\|_\infty = \max_{i=1}^d |x_i - y_i|$, the problem is to find a smallest axis-parallel cube that contains all the points. This can be done in time $O(d^2n)$ by finding the smallest enclosing box. The largest side-length of the box corresponds to the largest extent of the point set in any of the coordinate directions; to obtain a smallest enclosing cube, we simply extend the box along the other directions until all side lengths are equal.

A more interesting case is $d(x,y) = \|x-y\|_1 = \sum_{i=1}^d |x_i - y_i|$. This is the *Manhattan distance*. There, the problem can be written as

$$\begin{array}{ll} \text{minimize} & \rho \\ \text{subject to} & \sum_{i=1}^d |p_{ji} - c_i| \leqslant \rho, \quad j = 1, \ldots, n. \end{array}$$

where p_j is the j-th point and p_{ji} it's i-th coordinate Geometrically, the problem is now that of finding a smallest *cross polytope* (generalized octahedron) that contains the points. Algebraically, we can reduce it to a linear program, as follows.

We replace all $|p_{ji} - c_i|$ by new variables y_{ji} and add the additional constraints $y_{ji} \geqslant p_{ji} - c_i$ and $y_{ji} \geqslant c_i - pji$. The problem now is a linear program.

$$\begin{array}{lll} \text{minimize} & \rho & \\ \text{subject to} & \sum_{i=1}^{d} y_{ji} \leqslant \rho, & j = 1, \ldots, n \\ & y_{ji} \geqslant p_{ji} - c_i, & \forall i, j \\ & y_{ji} \geqslant c_i - p_{ji}, & \forall i, j. \end{array}$$

The claim is that the solution to this linear program also solves the original problem. For this, we need to observe two things: first of all, every optimal solution $(\tilde{c}, \tilde{\rho})$ to the original problem induces a feasible solution to the LP with the same value (simply set $y_{ji} := |p_{ji} - \tilde{c}_i|$), so the LP solution has value equal to $\tilde{\rho}$ or better. The second is that every optimal solution $((\tilde{y}_{ji})_{i,j}, \tilde{\rho})$ to the LP induces a feasible solution to the orginal problem with the same value: by $\sum_{i=1}^{d} \tilde{y}_{ji} \leqslant \tilde{\rho}$ and $\tilde{y}_{ji} \geqslant |p_{ji} - c_i|$, we also have $\sum_{i=1}^{d} |p_{ji} - c_i| \leqslant \tilde{\rho}$. This means, the original problem has value $\tilde{\rho}$ or better. From these two observations it follows that both problems have the same optimal value ρ_{opt}, and an LP solution of this value yields a smallest enclosing ball of P w.r.t. the Manhatten distance.

Questions

88. *Formulate the Swiss Algorithm for computing smallest enclosing balls, and discuss its relation with the Forever Swiss algorithm that we employ for the analysis!*

89. *The analysis of the Forever Swiss algorithm depends on a lower and an upper bound for the expected number of ballots after k controversial rounds. Sketch how these lower and upper bounds can be obtained, and how termination of the algorithm (with high probability) can be derived from them.*

90. *What is the expected runtime of the Swiss Algorithm for computing the smallest enclosing ball of a set of n points in fixed dimension d?*

91. *How can you compute smallest enclising balls in the Manhattan metric?*

References

[1] Kenneth L. Clarkson, A Las Vegas algorithm for linear programming when the dimension is small. In *Proc. 29th Annu. IEEE Sympos. Found. Comput. Sci.*, pp. 452–456, 1988.

[2] Kenneth L. Clarkson, Las Vegas algorithms for linear and integer programming when the dimension is small. *J. ACM*, **42**, (1995), 488–499.

[3] Kaspar Fischer, *Smallest enclosing balls of balls. Combinatorial structure and algorithms*. Ph.D. thesis, ETH Zürich, 2005.

[4] Emo Welzl, Smallest enclosing disks (balls and ellipsoids). In H. Maurer, ed., *New Results and New Trends in Computer Science*, vol. 555 of *Lecture Notes Comput. Sci.*, pp. 359–370, Springer, 1991.

Appendix H

Epsilon Nets

H.1 Motivation

Here is our scenario for this chapter. We are given a set A of points in $\mathbb{R}^d$ and a family $\mathcal{R}$ of *ranges* $r \subseteq \mathbb{R}^d$, for example the set of all balls, halfspaces, or convex sets in $\mathbb{R}^d$. A is huge and probably not even known completely; similarly, $\mathcal{R}$ may not be accessible explicitly (in the examples above, it is an uncountable set). Still, we want to learn something about A and some $r \in \mathcal{R}$.

The situation is familiar, definitely, if we don't insist on the geometric setting. For example, let A be the set of consumers living in Switzerland, and let $\tilde{r}$ be the subset of consumers who frequently eat a certain food product, say Lindt chocolate. We have similar subsets for other food products, and together, they form the family of ranges $\mathcal{R}$.

If we want to learn something about $\tilde{r}$, e.g. the ratio $\frac{|\tilde{r}|}{|A|}$ (the fraction of consumers frequently eating Lindt chocolate), then we typically sample a subset S of A and see what portion of S lies in $\tilde{r}$. We want to believe that

$$\frac{|\tilde{r} \cap S|}{|S|} \quad \text{approximates} \quad \frac{|\tilde{r}|}{|A|},$$

and statistics tells us to what extent this is justified. In fact, consumer surveys are based on this approach: in our example, S is a sample of consumers who are being asked about their chocolate preferences. After this, the quantity $|\tilde{r} \cap S|/|S|$ is known and used to predict the "popularity" $|\tilde{r}|/|A|$ of Lindt chocolate among Swiss consumers.

In this chapter, we consider a different kind of approximation. Suppose that we are interested in the most popular food products in Switzerland, the ones which are frequently eaten by more than an ε-fraction of all consumers, for some fixed $0 \leqslant \varepsilon \leqslant 1$. The goal is to find a small subset N of consumers that "represent" all popular products. Formally, we want to find a set $N \subseteq A$ such that

$$\text{for all } r: \quad \frac{|r|}{|A|} > \varepsilon \quad \Rightarrow \quad r \cap N \neq \emptyset.$$

Such a subset is called an *epsilon net.* Obviously, $N = A$ is an epsilon net for all ε, but as already mentioned above, the point here is to have a *small* set N.

Epsilon nets are very useful in many contexts that we won't discuss here. But already in the food consumption example above, it is clear that a small representative set of consumers is a good thing to have; for example if you quickly need a statement about a particular popular food product, you know that you will find somebody in your representative set who knows the product.

The material of this chapter is classic and goes back to Haussler and Welzl [1].

H.2 Range spaces and ε-nets.

Here is the formal framework. Let X be a (possibly infinite) set and $\mathcal{R} \subseteq 2^X$. The pair $(X, \mathcal{R})$ is called a *range space*[1], with X its *points* and the elements of $\mathcal{R}$ its *ranges*.

Definition H.1. *Let* $(X, \mathcal{R})$ *be a range space. Given* $A \subseteq X$*, finite, and* $\varepsilon \in \mathbb{R}$*,* $0 \leqslant \varepsilon \leqslant 1$*, a subset* N *of* A *is called an* ε-net of A (w.r.t. $\mathcal{R}$) *if*

$$\textit{for all } r \in \mathcal{R}: \quad |r \cap A| > \varepsilon|A| \quad \Rightarrow \quad r \cap N \neq \emptyset \ .$$

This definition is easy to write down, but it is not so easy to grasp, and this is why we will go through a couple of examples below. Note that we have a slightly more general setup here, compared to the motivating Section H.1 where we had $X = A$.

Examples Typical examples of range spaces in our geometric context are

- $(\mathbb{R}, \mathcal{H}_1)$ with $\mathcal{H}_1 := \{(-\infty, a] \mid a \in \mathbb{R}\} \cup \{[a, \infty) \mid a \in \mathbb{R}\}$ (*half-infinite intervals*), and
- $(\mathbb{R}, \mathcal{I})$ with $\mathcal{I} := \{[a, b] \| a, b \in \mathbb{R}, a \leqslant b\}$ (*intervals*),

and higher-dimensional counter-parts

- $(\mathbb{R}^d, \mathcal{H}_d)$ with $\mathcal{H}_d$ the closed *halfspaces* in $\mathbb{R}^d$ bounded by hyperplanes,
- $(\mathbb{R}^d, \mathcal{B}_d)$ with $\mathcal{B}_d$ the closed *balls* in $\mathbb{R}^d$,
- $(\mathbb{R}^d, \mathcal{S}_d)$ with $\mathcal{S}_d$ the d-dimensional *simplices* in $\mathbb{R}^d$, and
- $(\mathbb{R}^d, \mathcal{C}_d)$ with $\mathcal{C}_d$ the *convex sets* in $\mathbb{R}^d$.

[1]In order to avoid confusion: A range space is nothing else but a set system, sometimes also called hypergraph. It is the context, where we think of X as points and $\mathcal{R}$ as ranges in some geometric ambient space, that suggests the name at hand.

ε-Nets w.r.t. $(\mathbb{R}, \mathcal{H}_1)$ are particularly simple to obtain. For $A \subseteq \mathbb{R}$, $N := \{\min A, \max A\}$ is an ε-net for every ε—it is even a 0-net. That is, there are ε-nets of size 2, independent from $|A|$ and ε.

The situation gets slightly more interesting for the range space $(\mathbb{R}, \mathcal{I})$ with intervals. Given ε and A with elements

$$a_1 < a_2 < \cdots < a_n ,$$

we observe that an ε-net must contain at least one element from any contiguous sequence $\{a_i, a_{i+1}, \ldots, a_{i+k-1}\}$ of $k > \varepsilon n$ (i.e. $k \geqslant \lfloor \varepsilon n \rfloor + 1$) elements in A. In fact, this is a necessary and sufficient condition for ε-nets w.r.t. intervals. Hence,

$$\{a_{\lfloor \varepsilon n \rfloor + 1}, a_{2\lfloor \varepsilon n \rfloor + 2}, \ldots\}$$

is an ε-net of size[2] $\left\lfloor \frac{n}{\lfloor \varepsilon n \rfloor + 1} \right\rfloor \leqslant \lceil \frac{1}{\varepsilon} \rceil - 1$. So while the size of the net depends now on ε, it is still independent of $|A|$.

No point in a large range. Let us start with a simple exercise, showing that large ranges are easy to "catch". Assume that $|r \cap A| > \varepsilon |A|$ for some fixed r and ε, $0 \leqslant \varepsilon \leqslant 1$.

Now consider the set S obtained by drawing s elements uniformly at random from A (with replacement). We write

$$S \sim A^s,$$

indicating that S is chosen uniformly at random from the set A^s of s-element sequences over A.

What is the probability that $S \sim A^s$ fails to intersect with r, i.e. $S \cap r = \emptyset$? For $p := \frac{|r \cap A|}{|A|}$ (note $p > \varepsilon$) we get[3]

$$\text{prob}(S \cap r = \emptyset) = (1-p)^s < (1-\varepsilon)^s \leqslant e^{-\varepsilon s} .$$

That is, if $s = \frac{1}{\varepsilon}$ then this probability is at most $e^{-1} \approx 0.368$, and if we choose $s = \lambda \frac{1}{\varepsilon}$, then this probability decreases exponentially with λ: It is at most $e^{-\lambda}$.

For example, if $|A| = 10000$ and $|r \cap A| > 100$ (r contains more than 1% of the points in A), then a sample of 300 points is disjoint from r with probability at most $e^{-3} \approx 0.05$.

Smallest enclosing balls. Here is a potential use of this for a geometric problem. Suppose A is a set of n points in $\mathbb{R}^d$, and we want to compute the smallest enclosing ball of A. In fact, we are willing to accept some mistake, in that, for some given ε, we want a small ball that contains all but at most εn points from A. So let's choose a sample S of $\lambda \frac{1}{\varepsilon}$ points drawn uniformly (with replacement) from A and compute the smallest enclosing

[2]The number L of elements in the set is the largest ℓ such that $\ell(\lfloor \varepsilon n \rfloor + 1) \leqslant n$, hence $L = \left\lfloor \frac{n}{\lfloor \varepsilon n \rfloor + 1} \right\rfloor$. Since $\lfloor \varepsilon n \rfloor + 1 > \varepsilon n$, we have $\frac{n}{\lfloor \varepsilon n \rfloor + 1} < \frac{1}{\varepsilon}$, and so $L < \frac{1}{\varepsilon}$, i.e. $L \leqslant \lceil \frac{1}{\varepsilon} \rceil - 1$.

[3]We make use of the inequality $1 + x \leqslant e^x$ for all $x \in \mathbb{R}$.

ball B of S. Now let $r := \mathbb{R}^d \setminus B$, the complement of B in $\mathbb{R}^d$, play the role of the range in the analysis above. Obviously $r \cap S = \emptyset$, so it is unlikely that $|r \cap A| > \varepsilon|A|$, since—if so—the probability of $S \cap r = \emptyset$ was at most $e^{-\lambda}$.

It is important to understand that this was complete nonsense!

For the probabilistic analysis above we have to first choose r and then draw the sample—and not, as done in the smallest ball example, first draw the sample and then choose r *based on the sample.* That cannot possibly work, since we could always choose r simply as the complement $\mathbb{R}^d \setminus S$—then clearly $r \cap S = \emptyset$ and $|r \cap A| > \varepsilon|A|$, unless $|S| \geqslant (1-\varepsilon)|A|$.

While you hopefully agree on this, you might find the counterargument with $r = \mathbb{R}^d \setminus S$ somewhat artificial, e.g. complements of balls cannot be that selective in 'extracting' points from A. It is exactly the purpose of this chapter to understand to what extent this advocated intuition is justified or not.

H.3 Either almost all is needed or a constant suffices.

Let us reveal the spectrum of possibilities right away, although its proof will have to await some preparatory steps.

Theorem H.2. *Let $(X, \mathcal{R})$ be an infinite range space. Then one of the following two statements holds.*

(1) For every $n \in \mathbb{N}$ there is a set $A_n \subseteq X$ with $|A_n| = n$ such that for every ε, $0 \leqslant \varepsilon \leqslant 1$, an ε-net must have size at least $(1-\varepsilon)n$.

(2) There is a constant δ depending on $(X, \mathcal{R})$, such that for every finite $A \subseteq X$ and every ε, $0 < \varepsilon \leqslant 1$, there is an ε-net of A w.r.t. $\mathcal{R}$ of size at most $\frac{8\delta}{\varepsilon} \log_2 \frac{4\delta}{\varepsilon}$ (independent of the size of A).

That is, either we have always ε-nets of size independent of $|A|$, or we have to do the trivial thing, namely choosing all but εn points for an ε-net. Obviously, the range spaces $(\mathbb{R}, \mathcal{H}_1)$ and $(\mathbb{R}, \mathcal{I})$ fall into category (2) of the theorem.

For an example for (1), consider $(\mathbb{R}^2, \mathcal{C}_2)$. For any $n \in \mathbb{N}$, let A_n be a set of n points in convex position. For every $N \subseteq A_n$ there is a range $r \in \mathcal{C}_2$, namely the convex hull of $A_n \setminus N$, such that $A_n \cap r = A_n \setminus N$ (hence, $r \cap N = \emptyset$); see Figure H.1. Therefore, $N \subseteq A_n$ cannot be an ε-net of A_n w.r.t. $\mathcal{C}_2$ if $|A_n \setminus N| = n - |N| > \varepsilon n$. Consequently, an ε-net must contain at least $n - \varepsilon n = (1-\varepsilon)n$ points.[4]

So what distinguishes $(\mathbb{R}^2, \mathcal{C}_2)$ from $(\mathbb{R}, \mathcal{H}_1)$ and $(\mathbb{R}, \mathcal{I})$? And which of the two cases applies to the many other range spaces we have listed above? Will all of this eventually tell us something about our attempt of computing a small ball covering all but at most εn out of n given points? This and more should be clear by the end of this chapter.

[4]If we were satisfied with any abstract example for category (1), we could have taken $(X, 2^X)$ for any infinite set X.

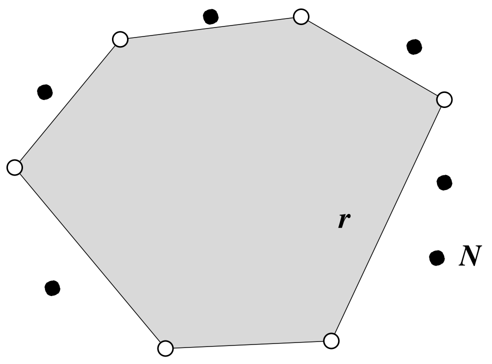


Figure H.1: *If* $\mathcal{R}$ *consists of all convex sets in the plane, then only trivial epsilon nets exist: for every subset* N *(black points) of a set* A_n *in convex position, the range* $r = \operatorname{conv}(A_n \setminus N)$ *fails to intersect* N.

H.4 What makes the difference: VC-dimension

Given a range space $(X, \mathcal{R})$ and $A \subseteq X$, we let

$$\mathcal{R}|_A := \{r \cap A \mid r \in \mathcal{R}\},$$

the *projection of* $\mathcal{R}$ *to* A. Even if $\mathcal{R}$ is infinite, $\mathcal{R}|_A$ is always of size at most 2^n if A is an n-element set. The significance of projections in our context becomes clear if we rewrite Definition H.1 in terms of projections: $N \subseteq A$ is an ε-net if

$$\text{for all } r \in \mathcal{R}|_A: \quad |r| > \varepsilon|A| \quad \Rightarrow \quad r \cap N \neq \emptyset.$$

All of a sudden, the conditions for an ε-net have become discrete, and they only depend on the finite range space $(A, \mathcal{R}|_A)$.

Note that, for A a set of n points in convex position in the plane, $\mathcal{C}_2|_A = 2^A$; we get every subset of A by an intersection with a convex set (this is also the message of Figure H.1). That is $|\,\mathcal{C}_2|_A\,| = 2^n$, the highest possible value.

For A a set of n points in $\mathbb{R}$, we can easily see that[5] $|\,\mathcal{I}|_A\,| = \binom{n+1}{2} + 1 = O(n^2)$. A similar argument shows that $|\,\mathcal{H}_1|_A\,| = 2n$. Now comes the crucial definition.

[5]Given A as $a_1 < a_2 \cdots < a_n$ we can choose another $n+1$ points b_i, $0 \leqslant i \leqslant n$, such that

$$b_0 < a_1 < b_1 < a_2 < b_2 < \cdots b_{n-1} < a_n < b_n .$$

Each nonempty intersection of A with an interval can be uniquely written as $A \cap [b_i, b_j]$ for $0 \leqslant i < j \leqslant n$. This gives $\binom{n+1}{2}$ plus one for the empty set.

Definition H.3. *Given a range space* $(X, \mathcal{R})$*, a subset* A *of* X *is* shattered by $\mathcal{R}$ *if* $\mathcal{R}|_A = 2^A$*. The* VC-dimension[6] of $(X, \mathcal{R})$*,* $\mathrm{VCdim}(X, \mathcal{R})$*, is the cardinality (possibly infinite) of the largest subset of* X *that is shattered by* $\mathcal{R}$*. If no set is shattered (i.e. not even the empty set which means that* $\mathcal{R}$ *is empty), we set the VC-dimension to* -1*.*

We had just convinced ourselves that $(\mathbb{R}^2, \mathcal{C}_2)$ has arbitrarily large sets that can be shattered. Therefore, $\mathrm{VCdim}(\mathbb{R}^2, \mathcal{C}_2) = \infty$.

Consider now $(\mathbb{R}, \mathcal{I})$. Two points $A = \{a, b\}$ can be shattered, since for each of the 4 subsets, $\emptyset$, $\{a\}$, $\{b\}$, and $\{a, b\}$, of A, there is an interval that generates that subset by intersection with A. However, for $A = \{a, b, c\}$ with $a < b < c$ there is no interval that contains a and c but not b. Hence, $\mathrm{VCdim}(\mathbb{R}, \mathcal{I}) = 2$.

Exercise H.4. *What is* $\mathrm{VCdim}(\mathbb{R}, \mathcal{H}_1)$*?*

Exercise H.5. *Prove that if* $\mathrm{VCdim}(X, \mathcal{R}) = \infty$*, then we are in case (1) of Theorem H.2, meaning that only trivial epsilon nets always exist.*

The size of projections for finite VC-dimension. Here is the (for our purposes) most important consequence of finite VC dimension: there are only polynomially many ranges in every projection.

Lemma H.6 (Sauer's Lemma). *If* $(X, \mathcal{R})$ *is a range space of finite VC-dimension at most* δ*, then*

$$|\mathcal{R}|_A| \leqslant \Phi_\delta(n) := \sum_{i=0}^{\delta} \binom{n}{i}$$

for all $A \subseteq X$ *with* $|A| = n$*.*

Proof. First let us observe that $\Phi : \mathbb{N}_0 \cup \{-1\} \times \mathbb{N}_0 \to \mathbb{N}_0$ is defined by the recurrence[7]

$$\Phi_\delta(n) = \begin{cases} 0 & \delta = -1, \\ 1 & n = 0 \text{ and } \delta \geqslant 0, \text{ and} \\ \Phi_\delta(n-1) + \Phi_{\delta-1}(n-1) & \text{otherwise.} \end{cases}$$

Second, we note that the VC-dimension cannot increase by passing from $(X, \mathcal{R})$ to a projection (A, R), $R := \mathcal{R}|_A$. Hence, it suffices to consider the finite range space (A, R)—which is of VC-dimension at most δ—and show $|R| \leqslant \Phi_\delta(n)$ (since Φ is monotone in δ).

[6] 'VC' in honor of the Russian statisticians V. N. Vapnik and A. Ya. Chervonenkis, who discovered the crucial role of this parameter in the late sixties.

[7] We recall that the binomial coefficients $\binom{n}{k}$ (with $k, n \in \mathbb{N}_0$) satisfy the recurrence $\binom{n}{k} = 0$ if $n < k$, $\binom{n}{0} = 1$, and $\binom{n}{k} = \binom{n-1}{k} + \binom{n-1}{k-1}$.

Now we proceed to a proof by induction of this inequality. If $A = \emptyset$ or $R = \emptyset$ the statement is trivial. Otherwise, we consider the two 'derived' range spaces for some fixed $x \in A$:

$$(A \setminus \{x\}, R - x), \qquad \text{with } R - x := \{r \setminus \{x\} \mid r \in R\}$$

(note $R - x = R|_{A \setminus \{x\}}$) and

$$(A \setminus \{x\}, R^{(x)}), \qquad \text{with } R^{(x)} := \{r \in R \mid x \notin r, r \cup \{x\} \in R\}.$$

Observe that the ranges in $R^{(x)}$ are exactly those ranges in $R-x$ that have two preimages under the map

$$R \ni \quad r \quad \mapsto \quad r \setminus \{x\} \quad \in R - x\ ,$$

all other ranges have a unique preimage. Consequently,

$$|R| = |R - x| + |R^{(x)}|\ .$$

We have $|R-x| \leqslant \Phi_\delta(n-1)$. If $A' \subseteq A \setminus \{x\}$ is shattered by $R^{(x)}$, then $A' \cup \{x\}$ is shattered by R. Hence, $(A \setminus \{x\}, R^{(x)})$ has VC-dimension at most $\delta - 1$ and $|R^{(x)}| \leqslant \Phi_{\delta-1}(n-1)$. Summing up, it follows that

$$|R| \leqslant \Phi_\delta(n-1) + \Phi_{\delta-1}(n-1) = \Phi_\delta(n)$$

which yields the assertion of the lemma. □

In order to see that the bound given in the lemma is tight, consider the range space

$$\left(X, \bigcup_{i=0}^{\delta} \binom{X}{i}\right).$$

Obviously, a set of more than δ elements cannot be shattered (hence, the VC-dimension is at most δ), and for any finite $A \subseteq X$, the projection of the ranges to A is $\bigcup_{i=0}^{\delta} \binom{A}{\delta}$—with cardinality $\Phi_\delta(|A|)$.

We note that a rough, but for our purposes good enough estimate for Φ is given by[8]

$$\Phi_\delta(n) \leqslant n^\delta \quad \text{for } \delta \geqslant 2.$$

We have seen now that the maximum possible size of projections either grows exponentially (2^n in case of infinite VC-dimension) or it is bounded by a polynomial n^δ in case of finite VC-dimension δ). The latter is the key to the existence of small ε-nets. Before shedding light on this, let us better understand when the VC-dimension is finite.

[8]A better estimate, at least for $\delta \geqslant 3$, is given by $\Phi_\delta(n) < \left(\frac{en}{\delta}\right)^d$ for all $n, d \in \mathbb{N}$, $d \leqslant n$.

H.5 VC-dimension of Geometric Range Spaces

Halfspaces. Let us investigate the VC-dimension of $(\mathbb{R}^2, \mathcal{H}_2)$. It is easily seen that three points in the plane can be shattered by halfplanes, as long a they do not lie on a common line. Hence, the VC-dimension is at least 3. Now consider 4 points. If three of them lie on a common line, there is no way to separate the middle point on this line from the other two by a halfplane. So let us assume that no three points lie on a line. Either three of them are vertices of a triangle that contains the fourth point—then we cannot possibly separate the fourth point from the remaining three points by a halfplane. Or the four points are vertices of a convex quadrilateral—then there is no way of separating the endpoints from a diagonal from the other two points. Consequently, four points cannot be shattered, and $\mathrm{VCdim}(\mathbb{R}^2, \mathcal{H}_2) = 3$ is established.

The above argument gets tedious in higher dimensions, if it works in a rigorous way at all. Fortunately, we can employ a classic: By Radon's Lemma (Theorem 4.8), every set A of $\geqslant d+2$ points in $\mathbb{R}^d$ can be partitioned into two disjoint subsets A_1 and A_2 such that $\mathrm{conv}(A_1) \cap \mathrm{conv}(A_2) \neq \emptyset$. We get, as an easy implication, that a set A of at least $d+2$ points in $\mathbb{R}^d$ cannot be shattered by halfspaces. Indeed, let $A_1 \mathbin{\dot\cup} A_2$ be a partition as guaranteed by Radon's Lemma. Now every halfspace containing A_1 must contain at least one point of A_2, hence $h \cap A = A_1$ is impossible for a halfspace h and thus A is not shattered by $\mathcal{H}_d$. Moreover, it is easily seen that the vertex set of a d-dimensional simplex (there are $d+1$ vertices) can be shattered by halfspaces (each subset of the vertices forms a face of the simplex and can thus be separated from the rest by a hyperplane). We summarize that

$$\mathrm{VCdim}(\mathbb{R}^d, \mathcal{H}_d) = d + 1 \ .$$

Let us now consider the range space $(\mathbb{R}^d, \check{\mathcal{H}}_d)$, where $\check{\mathcal{H}}_d$ denotes the set of all closed halfspaces below non-vertical hyperplanes[9]–we call these *lower halfspaces*. Since $\check{\mathcal{H}}_d \subseteq \mathcal{H}_d$, the VC-dimension of $(\mathbb{R}^d, \check{\mathcal{H}}_d)$ is at most $d+1$, but, in fact, it not too difficult to show

$$\mathrm{VCdim}(\mathbb{R}^d, \check{\mathcal{H}}_d) = d \ . \tag{H.7}$$

(Check this claim at least for $d = 2$.) This range space is a geometric example where the bound of Sauer's Lemma is attained. Indeed, for any set A of n points in $\mathbb{R}^d$ in general position[10], it can be shown that

$$\left|\check{\mathcal{H}}_d\big|_A\right| = \Phi_d(n).$$

[9]A hyperplane is called non-vertical if it can be specified by a linear equation $\sum_{i=1}^{d} \lambda_i x_i = \lambda_{d+1}$ with $\lambda_d \neq 0$; see also Section 1.2

[10]No $i+2$ on a common i-flat for $i \in \{1, 2, \ldots, d-1\}$; in particular, no $d+1$ points on a common hyperplane.

Balls. It is easy to convince oneself that the VC-dimension of disks in the plane is 3: Three points not on a line can be shattered and four points cannot. Obviously not, if one of the points is in the convex hull of the other, and for four vertices of a convex quadrilateral, it is not possible for both diagonals to be separated from the endpoints of the respective other diagonal by a circle (if you try to draw a picture of this, you see that you get two circles that intersect four times, which we know is not be the case).

A more rigorous argument which works in all dimensions is looming with the help of (H.7), if we employ the following transformation called *lifting map* that we have already encountered for $d = 2$ in Section 5.3:

$$\begin{array}{rcl} \mathbb{R}^d & \longrightarrow & \mathbb{R}^{d+1} \\ (x_1, x_2, \ldots, x_d) = p & \mapsto & \ell(p) = (x_1, x_2, \ldots, x_d, \, {x_1}^2 + {x_2}^2 + \cdots + {x_d}^2) \end{array}$$

(For a geometric interpretation, this is a vertical projection of $\mathbb{R}^d$ to the unit paraboloid $x_{d+1} = {x_1}^2 + {x_2}^2 + \cdots + {x_d}^2$ in $\mathbb{R}^{d+1}$.) The remarkable property of this transformation is that it maps balls in $\mathbb{R}^d$ to halfspaces in $\mathbb{R}^{d+1}$ in the following sense.

Consider a ball $B_d(c, \rho)$ ($c = (c_1, c_2, \ldots, c_d) \in \mathbb{R}^d$ the center, and $\rho \in \mathbb{R}^+$ the radius). A point $p = (x_1, x_2, \ldots, x_d)$ lies in this ball if and only if

$$\begin{aligned} & \sum_{i=1}^{d} (x_i - c_i)^2 \leqslant \rho^2 \Leftrightarrow \sum_{i=1}^{d} ({x_i}^2 - 2x_i c_i + {c_i}^2) \leqslant \rho^2 \\ \Leftrightarrow \quad & \left(\sum_{i=1}^{d} (-2c_i) x_i \right) + ({x_1}^2 + {x_2}^2 + \cdots + {x_d}^2) \leqslant \rho^2 - \sum_{i=1}^{d} {c_i}^2 \ ; \end{aligned}$$

this equivalently means that $\ell(p)$ lies below the non-vertical hyperplane (in $\mathbb{R}^{d+1}$)

$$\begin{aligned} h = h(c, \rho) &= \left\{ x \in \mathbb{R}^{d+1} \;\middle|\; \sum_{i=1}^{d+1} h_i x_i = h_{d+2} \right\} \quad \text{with} \\ (h_1, h_2, \ldots, h_d, h_{d+1}, h_{d+2}) &= \left((-2c_1), (-2c_2), \ldots, (-2c_d), 1, \rho^2 - \sum_{i=1}^{d} {c_i}^2 \right) \ . \end{aligned}$$

It follows that a set $A \subseteq \mathbb{R}^d$ is shattered by $\mathcal{B}_d$ (the set of closed balls in $\mathbb{R}_d$) if and only if $\ell(A) := \{\ell(p) \mid p \in A\}$ is shattered by $\check{\mathcal{H}}_{d+1}$. Assuming (H.7), this readily yields

$$\mathrm{VCdim}(\mathbb{R}^d, \mathcal{B}_d) = d + 1 \ .$$

The lifting map we have employed here is a special case of a more general paradigm called *linearization* which maps non-linear conditions to linear conditions in higher dimensions.

We have clarified the VC-dimension for all examples of range spaces that we have listed in Section H.2, except for the one involving simplices. The following exercise should help to provide an answer.

Exercise H.8. *For a range space* $(X, \mathcal{R})$ *of VC-dimension* d *and a number* $k \in \mathbb{N}$, *consider the range space* $(X, \mathcal{I})$ *where ranges form intersections of at most* k *ranges from* $\mathcal{R}$, *that is,* $\mathcal{I} = \{\bigcap_{i=1}^{k} R_i \,|\, \forall i : R_i \in \mathcal{R}\}$. *Give an upper bound for the VC-dimension of* $(X, \mathcal{I})$.

H.6 Small ε-Nets, an Easy Warm-up Version

Let us prove a first bound on the size of ε-nets when the VC-dimension is finite.

Theorem H.9. *Let* $n \in \mathbb{N}$ *and* $(X, \mathcal{R})$ *be a range space of VC-dimension* $d \geqslant 2$. *Then for any* $A \subseteq X$ *with* $|A| = n$ *and any* $\varepsilon \in \mathbb{R}^+$ *there exists an* ε*-net* N *of* A *w.r.t.* $\mathcal{R}$ *with* $|N| \leqslant \lceil \frac{d \ln n}{\varepsilon} \rceil$.

Proof. We restrict our attention to the finite projected range space (A, R), $R := \mathcal{R}|_A$, for which we know $|R| \leqslant \Phi_d(n) \leqslant n^d$. It suffices to show that there is a set $N \subseteq A$ with $|N| \leqslant \frac{d \ln n}{\varepsilon}$ which contains an element from each $r \in R_\varepsilon := \{r \in R \colon |r| > \varepsilon n\}$.

Suppose, for some $s \in \mathbb{N}$ (to be determined), we let $N \sim A^s$. For each $r \in R_\varepsilon$, we know that $\text{prob}(r \cap N = \emptyset) < (1-\varepsilon)^s \leqslant e^{-\varepsilon s}$. Therefore,

$$\begin{aligned} \text{prob}(N \text{ is not } \varepsilon\text{-net of } A) &= \text{prob}(\exists r \in R_\varepsilon : r \cap N = \emptyset) \\ &= \text{prob}\Big(\bigvee_{r \in R_\varepsilon} (r \cap N = \emptyset)\Big) \\ &\leqslant \sum_{r \in R_\varepsilon} \text{prob}(r \cap N = \emptyset) < |R_\varepsilon| e^{-\varepsilon s} \leqslant n^d e^{-\varepsilon s} \,. \end{aligned}$$

It follows that if s is chosen so that $n^d e^{-\varepsilon s} \leqslant 1$, then $\text{prob}(N \text{ is not } \varepsilon\text{-net of } A) < 1$ and there remains a positive probability for the event that N is an ε-net of A. Now

$$n^d e^{-\varepsilon s} \leqslant 1 \quad \Leftrightarrow \quad n^d \leqslant e^{\varepsilon s} \quad \Leftrightarrow \quad d \ln n \leqslant \varepsilon s.$$

That is, for $s = \lceil \frac{d \ln n}{\varepsilon} \rceil$, the probability of obtaining an ε-net is positive, and therefore an ε-net of that size has to exist.[11] □

If we are willing to invest a little more in the size of the random sample N, then the probability of being an ε-net grows dramatically. More specifically, for $s = \lceil \frac{d \ln n + \lambda}{\varepsilon} \rceil$, we have

$$n^d e^{-\varepsilon s} \leqslant n^d e^{-d \ln n - \lambda} = e^{-\lambda} \,,$$

and, therefore, a sample of that size is an ε-net with probability at least $1 - e^{-\lambda}$.

[11]This line of argument "If an experiment produces a certain object with positive probability, then it has to exist", as trivial as it is, admittedly needs some time to digest. It is called *The Probabilistic Method*, and was used and developed to an amazing extent by the famous Hungarian mathematician Paul Erdős starting in the thirties.

We realize that we need $\frac{d \ln n}{\varepsilon}$ sample size to compensate for the (at most) n^d subsets of A which we have to hit—it suffices to ensure positive success probability. The extra $\frac{\lambda}{\varepsilon}$ allows us to boost the success probability.

Also note that if A were shattered by $\mathcal{R}$, then $R = 2^A$ and $|R| = 2^n$. Using this bound instead of n^d in the proof above would require us to choose s to be roughly $\frac{n \ln 2}{\varepsilon}$, a useless estimate which even exceeds n unless ε is large (at least $\ln 2 \approx 0.69$).

Smallest enclosing balls, again It is time to rehabilitate ourselves a bit concerning the suggested procedure for computing a small ball containing all but at most $\varepsilon|A|$ points from an n-point set $A \subseteq \mathbb{R}^d$.

Let $(\mathbb{R}^d, {\mathcal{B}_d}^{\text{compl}})$ be the range space whose ranges consist of all complements of closed balls in $\mathbb{R}^d$. This has the same VC-dimension $d+1$ as $(\mathbb{R}^d, \mathcal{B}_d)$. Indeed, if $A \cap r = A'$ for a ball r, then $A \cap (\mathbb{R}^d \setminus r) = A \setminus A'$, so A is shattered by $\mathcal{B}_d$ if and only if A is shattered by ${\mathcal{B}_d}^{\text{compl}}$.

Hence, if we choose a sample N of size $s = \lceil \frac{(d+1)\ln n + \lambda}{\varepsilon} \rceil$ then, with probability at least $1 - e^{-\lambda}$ this is an ε-net for A w.r.t. ${\mathcal{B}_d}^{\text{compl}}$. Let us quickly recall what this means: whenever the complement of some ball B has empty intersection with N, then this complement contains at most an $\varepsilon|A|$ points of A. As a consequence, the smallest ball enclosing N has at most $\varepsilon|A|$ points of A outside, with probability at least $1 - e^{-\lambda}$.

H.7 Even Smaller ε-Nets

We still need to prove part 2 of Theorem H.2, the existence of ε-nets whose size is independent of A. For this, we employ the same strategy as in the previous section, i.e. we sample elements from A uniformly at random, with replacement; a refined analysis will show that—compared to the bound of Theorem H.9—much less elements suffice. Here is the main technical lemma.

Lemma H.10. *Let $(X, \mathcal{R})$ be a range space of VC-dimension $\delta \geqslant 2$, and let $A \subseteq X$ be finite. If $x \sim A^m$ for $m \geqslant 8/\varepsilon$, then for the set N_x of elements occurring in x, we have*

$$\text{prob}(N_x \textit{ is not an } \varepsilon\textit{-net for } A \textit{ w.r.t. } \mathcal{R}) \leqslant 2\Phi_\delta(2m)2^{-\frac{\varepsilon m}{2}} .$$

Before we prove this, let us derive the bound of Theorem H.2 from it. Let us recall the statement.

Theorem H.2 (2). Let $(X, \mathcal{R})$ be a range space with VC-dimension $\delta \geqslant 2$. Then for every finite $A \subseteq X$ and every ε, $0 < \varepsilon \leqslant 1$, there is an ε-net of A w.r.t. $\mathcal{R}$ of size at most $\frac{8\delta}{\varepsilon} \log_2 \frac{4\delta}{\varepsilon}$ (independent of the size of A).

We want that

$$2\Phi_\delta(2m)2^{-\frac{\varepsilon m}{2}} < 1,$$

since then we know that an ε-net of size m exists. We have

$$\begin{aligned}
& 2\Phi_\delta(2m)2^{-\frac{\varepsilon m}{2}} < 1 \\
\Leftarrow\quad & 2(2m)^\delta < 2^{\frac{\varepsilon m}{2}} \\
\Leftrightarrow\quad & 1 + \delta \log_2(2m) < \frac{\varepsilon m}{2} \\
\Leftarrow\quad & 2\delta \log_2 m < \frac{\varepsilon m}{2} \\
\Leftrightarrow\quad & \frac{4\delta}{\varepsilon} < \frac{m}{\log_2 m}.
\end{aligned}$$

In the second to last implication, we have used $\delta \geqslant 2$ and $m \geqslant 8$. Now we claim that the latter inequality is satisfied for $m = 2\frac{4\delta}{\varepsilon}\log_2\frac{4\delta}{\varepsilon}$. To see this, we need that $\frac{m}{\log_2 m} > \alpha$ for $m = 2\alpha\log_2\alpha$ and $\alpha = \frac{4\delta}{\varepsilon}$. We compute

$$\frac{m}{\log_2 m} = \frac{2\alpha\log_2\alpha}{\log_2(2\alpha\log_2\alpha)} = \frac{2\alpha\log_2\alpha}{1+\log_2\alpha+\log_2\log_2\alpha} = \alpha\frac{2\log_2\alpha}{1+\log_2\alpha+\log_2\log_2\alpha} \geqslant \alpha$$

as long as

$$\log_2\alpha \geqslant 1 + \log_2\log_2\alpha \Leftrightarrow \log_2\alpha \geqslant \log_2 2\log_2\alpha \Leftrightarrow \alpha \geqslant 2\log_2\alpha,$$

which holds as long as $\alpha \geqslant 2$, and this is satisfied for $\alpha = \frac{4\delta}{\varepsilon}$.

Proof. (Lemma H.10) By going to the projection $(A, \mathcal{R}|_A)$, we may assume that $X = A$, so we have a finite range space over n elements (see Section H.4). Fix $\varepsilon \in \mathbb{R}^+$. For $t \in \mathbb{N}$, a range $r \in \mathcal{R}$ and $x \in A^t$ (a t-vector of elements from A), we define

$$\text{count}(r, x) = |\{i \in [t] : x_i \in r\}|.$$

Now we consider two events over A^{2m}. The first one is the bad event of not getting an ε-net when we choose m elements at random from A (plus another m elements that we ignore for the moment):

$$Q := \left\{ xy \in A^{2m} \;\middle|\; x \in A^m, y \in A^m, \exists r \in R_\varepsilon : \text{count}(r, x) = 0 \right\}.$$

Recall that $R_\varepsilon = \{r \in \mathcal{R} : |r| > \varepsilon n\}$. Thus $\text{prob}(Q) = \text{prob}(N_x \text{ is not } \varepsilon\text{-net})$ which is exactly what we want to bound.

The second auxiliary event looks somewhat weird at first:

$$J := \left\{ xy \in A^{2m} \;\middle|\; x \in A^m, y \in A^m, \exists r \in R_\varepsilon : \text{count}(r, x) = 0 \text{ and } \text{count}(r, y) \geqslant \frac{\varepsilon m}{2} \right\}.$$

This event satisfies $J \subseteq Q$ and contains pairs of sequences x and y with somewhat contradicting properties for some r. While the first part x fails to contain any element from r, the second part y has many elements from r.

Claim 1. $\text{prob}(J) \leqslant \text{prob}(Q) \leqslant 2\text{prob}(J)$.

The first inequality is a consequence of $J \subseteq Q$. To prove the second inequality, we show that

$$\frac{\text{prob}(J)}{\text{prob}(Q)} = \frac{\text{prob}(J \cap Q)}{\text{prob}(Q)} = \text{prob}(J \mid Q) \geqslant \frac{1}{2}.$$

So suppose that $xy \in Q$, with "witness" r, meaning that $r \in R_\varepsilon$ and $\text{count}(r, x) = 0$. We show that $xy \in J$ with probability at least $1/2$, for every fixed such x and y chosen randomly from A^m. This entails the claim.

The random variable $\text{count}(r, y)$ is a sum of m independent Bernoulli experiments with success probability $p := |r|/n > \varepsilon$ (thus expectation p and variance $p(1-p)$). Using linearity of expectation and variance (the latter requires independence of the experiments), we get

$$\begin{aligned} E(\text{count}(r,y)) &= pm, \\ \text{Var}(\text{count}(r,y)) &= p(1-p)m. \end{aligned}$$

Now we use Chebyshew's inequality[12] to bound the probability of the "bad" event that $\text{count}(r, y) < \frac{\varepsilon m}{2}$. We have

$$\begin{aligned} & \text{prob}\left(\text{count}(r,y) < \frac{\varepsilon m}{2}\right) \\ \leqslant\ & \text{prob}\left(\text{count}(r,y) < \frac{pm}{2}\right) \\ \leqslant\ & \text{prob}\left(|\text{count}(r,y) - E(\text{count}(r,y))| > \frac{pm}{2}\right) \\ =\ & \text{prob}\left(|\text{count}(r,y) - E(\text{count}(r,y))| > \frac{1}{2(1-p)}\text{Var}(\text{count}(r,y))\right) \\ \leqslant\ & \frac{4(1-p)^2}{p(1-p)m} = \frac{4(1-p)}{pm} \leqslant \frac{4}{pm} < \frac{4}{\varepsilon m} \leqslant \frac{1}{2}, \end{aligned}$$

since $m \geqslant \frac{8}{\varepsilon}$. Hence

$$\text{prob}\left(\text{count}(r,y) \geqslant \frac{\varepsilon m}{2}\right) \geqslant \frac{1}{2},$$

and the claim is proved.

Now the weird event J reveals its significance: we can nicely bound its probability. The idea is this: We enumerate A as $A = \{a_1, a_2, \ldots, a_n\}$ and let the *type* of $z \in A^t$ be the n-sequence

$$\text{type}(z) = (\text{count}(a_1, z), \text{count}(a_2, z), \ldots, \text{count}(a_n, z)).$$

For example, the type of $z = (1, 2, 4, 2, 2, 4)$ w.r.t. $A = \{1, 2, 3, 4\}$ is $\text{type}(z) = (1, 3, 0, 2)$.

Now fix an arbitrary type τ. We will bound the conditional probability $\text{prob}(xy \in J \mid xy \text{ has type } \tau)$, and the value that we get is independent of τ. It follows that the same value also bounds $\text{prob}(J)$.

[12] $\text{prob}(|X - E(X)| \geqslant k\text{Var}(X)) \leqslant \frac{1}{k^2\text{Var}(X)}$

Claim 2. $\text{prob}(xy \in J \mid xy \text{ has type } \tau) \leqslant \Phi_\delta(2m)2^{-\varepsilon m/2}$.

To analyze the probability in question, we need to sample $z = xy$ uniformly at random from all sequences of type τ. This can be done as follows: take an arbitrary sequence z' of type τ, and then apply a random permutation $\pi \in S_{2m}$ to obtain $z = \pi(z')$, meaning that

$$z_i = z'_{\pi(i)} \text{ for all } i.$$

Why does this work? First of all, π preserves the type, and it is easy to see that all sequences z of type τ can be obtained in this way. Now we simply count the number of permutations that map z' to a fixed z and see that this number only depends on τ, so it is the same for all z. Indeed, for every element $a_i \in A$, there are $\tau_i!$ many ways of mapping the a_i's in z' to the a_i's in z. The number of permutations that map z' to z is therefore given by

$$\prod_{i=1}^{n} \tau_i!.$$

By these considerations,

$$\text{prob}(xy \in J \mid xy \text{ has type } \tau) = \text{prob}(\pi(z') \in J).$$

To estimate this, we let S be the set of distinct elements in z' (this is also a function of the type τ). Since $(X, \mathcal{R})$ has VC-dimension δ, we know from Lemma H.6 that $|\,\mathcal{R}|_S\,| \leqslant \Phi_\delta(2m)$, so at most that many different subsets T of S can be obtained by intersections with ranges $r \in \mathcal{R}$, and in particular with ranges $r \in R_\varepsilon$.

Now we look at some fixed such T, consider a permutation π and write $\pi(z') = xy$. We call T a *witness* for π if no element of x is in T, but at least $\varepsilon m/2$ elements of y are in T. According to this definition,

$$\pi(z') \in J \quad \Leftrightarrow \quad \pi \text{ has some witness } T \subseteq S.$$

By the union bound,

$$\text{prob}(\pi(z') \in J) = \text{prob}(\exists T\colon T \text{ is a witness for } \pi) \leqslant \sum_T \text{prob}(T \text{ is a witness for } \pi).$$

Suppose that z' contains $\ell \geqslant \varepsilon m/2$ occurences of elements of T (for smaller ℓ, T cannot be a witness). In order for $\pi(z') = xy$ to be in J, no element of T can appear in the first half x, but all ℓ occurrences of elements from T must fall into the second half y. Therefore, the probability of T being a witness for a random permutation is

$$\frac{\binom{m}{\ell}}{\binom{2m}{\ell}} = \frac{m(m-1)\cdots(m-\ell+1)}{2m(2m-1)\cdots(2m-\ell+1)} \leqslant 2^{-\ell} \leqslant 2^{-\frac{\varepsilon m}{2}},$$

since among all the $\binom{2m}{\ell}$ equally likely ways in which π distributes the ℓ occurences in z, exactly the $\binom{m}{\ell}$ equally likely ways of putting them into y are good.[13] Summing up

[13]This argument can be made more formal by explicitly counting the permutations that map $\{1, 2, \ldots, \ell\}$ to the set $\{1, 2, \ldots, m\}$. We leave this to the reader.

over all at most $\Phi_\delta(2m)$ sets T, we get

$$\text{prob}(xy \in J \mid xy \text{ has type } \tau) \leqslant \Phi_\delta(2m)2^{-\frac{\varepsilon m}{2}},$$

for all τ.

The proof is finished by combining the two claims:

$$\text{prob}(N_x \text{ is not } \varepsilon\text{-net}) = \text{prob}(Q) \leqslant 2\text{prob}(J) \leqslant 2\Phi_\delta(2m)2^{-\frac{\varepsilon m}{2}}.$$

□

Questions

92. *What is a range space?* Give a definition and a few examples.
93. *What is an epsilon net?* Provide a formal definition, and explain in words what it means.
94. *What is the VC dimension of a range space* Give a definition, and compute the VC dimension of a few example range spaces.
95. *Which range spaces always have small epsilon nets (and how small), and which ones don't?* Explain Theorem H.2.
96. *How can you compute small epsilon nets?* Explain the general idea, and give the analysis behind the weaker bound of Theorem H.9.
97. *How can you compute smaller epsilon nets?* Sketch the main steps of the proof of the stronger bound of Theorem H.2.

References

[1] David Haussler and Emo Welzl, Epsilon-nets and simplex range queries. *Discrete Comput. Geom.*, **2**, (1987), 127–151.