Geometry: Combinatorics & Algorithms, Lecture Notes 2017
geometry.inf.ethz.ch/gca17-6.pdf

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 5.17). 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.

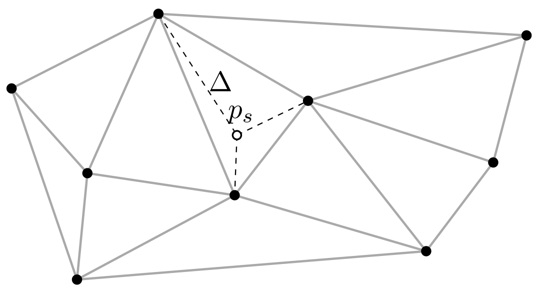


Figure 6.1: *Inserting* p_s *into* $\mathcal{DT}(s-1)$*: Step 1*

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

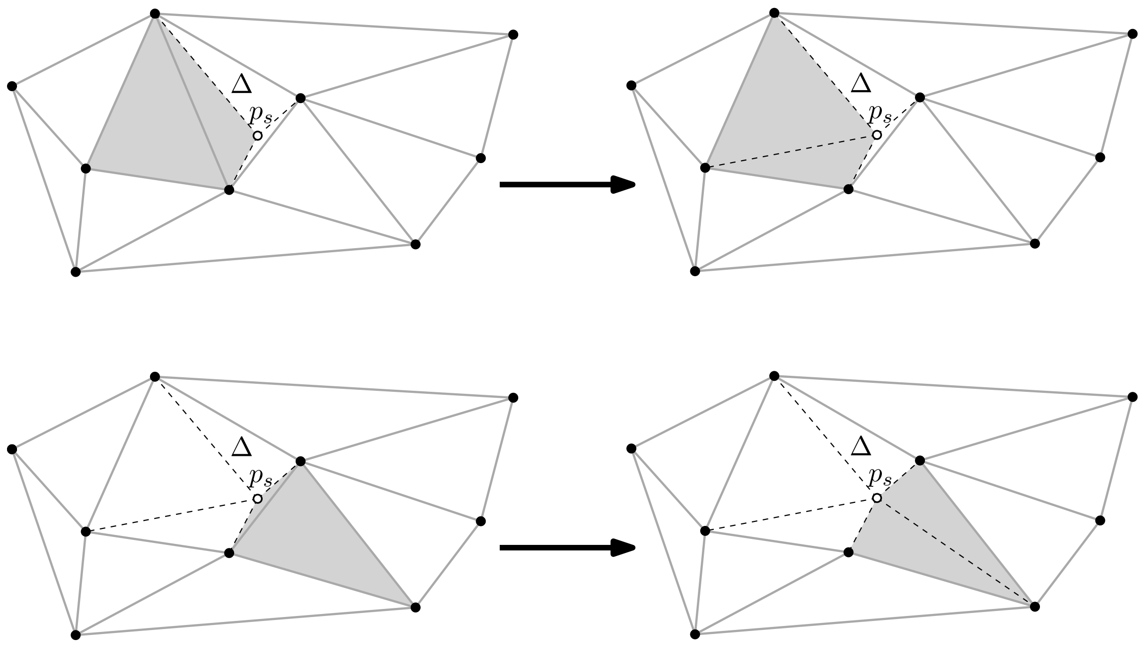


Figure 6.2: *Inserting* p_s *into* $\mathcal{DT}(s-1)$*: Step 2*

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]

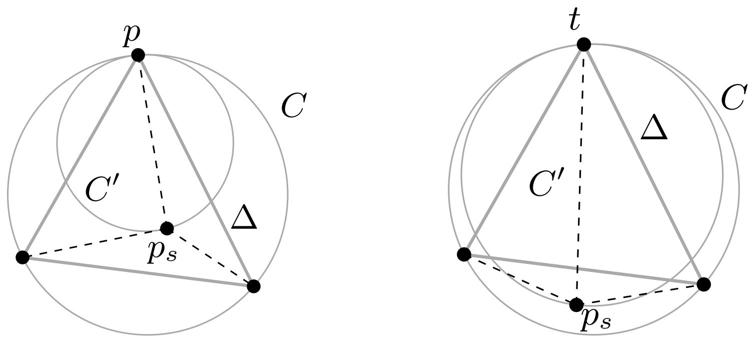
Proof. Let us consider one of the first three new edges, $\overline{p_s p}$, say. Since the triangle Δ has a circumcircle C strictly containing only p_s (Δ is in $\mathcal{DT}(s-1)$), we can shrink that circumcircle to a circle C' through p_s and p with no interior points, see Figure 6.3 (a). This proves 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.

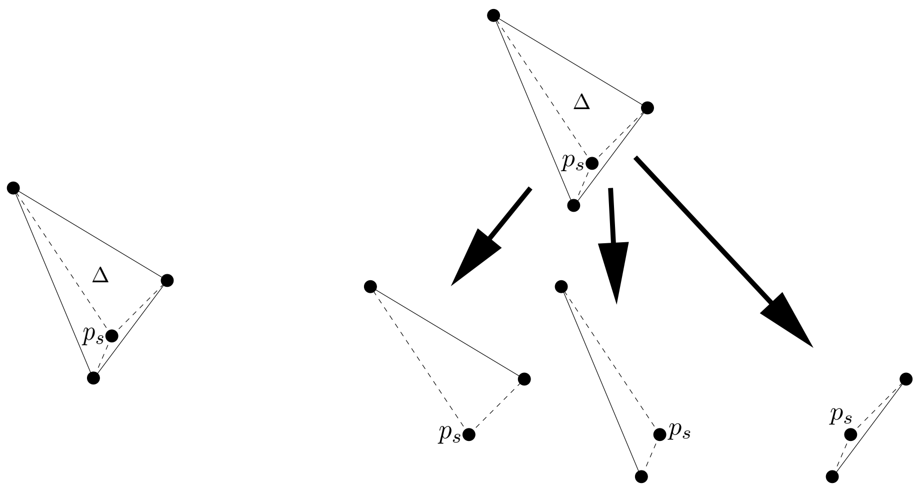


Figure 6.4: *The history graph: one triangle gets replaced by three triangles*

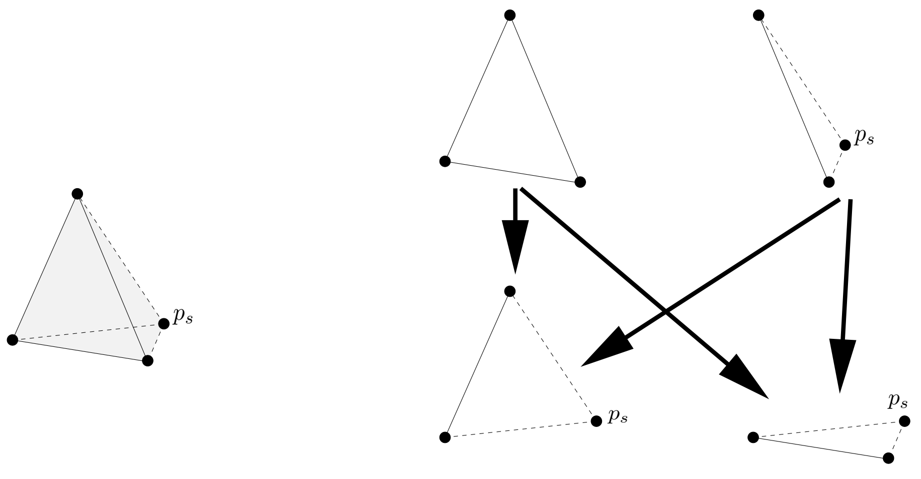


Figure 6.5: *The history graph: two triangles get replaced by two triangles*

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 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 seen to 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 early 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)

$$E\left[\sum_{s=1}^{n}\left(\sum_{\triangle \in \tau_s} |K(\triangle)|\right)\right] = O\left(\sum_{s=1}^{n}\frac{n-s}{s}\right) \leqslant O\left(n\sum_{s=1}^{n}\frac{1}{s}\right) = O(n \log n).$$

□

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

27. *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".*
28. *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 $\mathcal{DT}(s)$?
29. *What are the two types of triangles that the history graph contains?*