Chapter 9

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.

9.1 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 9.1 *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 9.1 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).

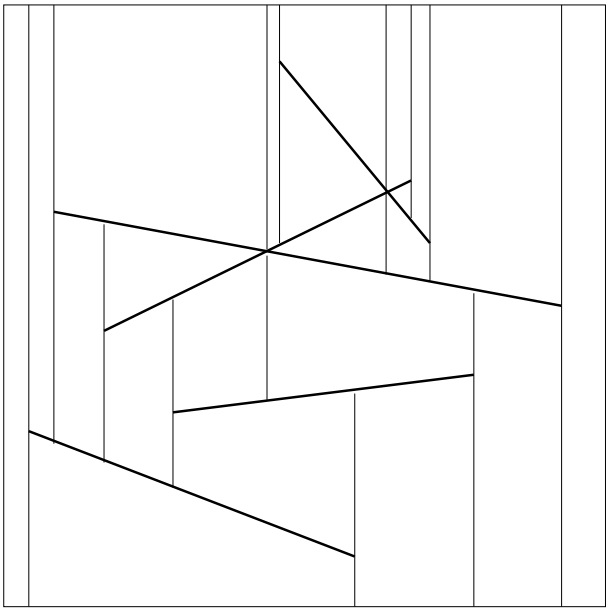

Figure 9.1: *The trapezoidal map of five line segments (depicted in bold)*

9.2 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 9.2). This is a more powerful alternative to Kirkpatrick's algorithm that handles only triangulations, and which is treated in Section 10.5. The preprocessing constructs the trapezoidal map of the segments (Figure 9.3) 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 line-sweep algorithm we saw in Section 4.2, which takes time $O((n + K) \log n)$.

9.3 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 5.2), for example.

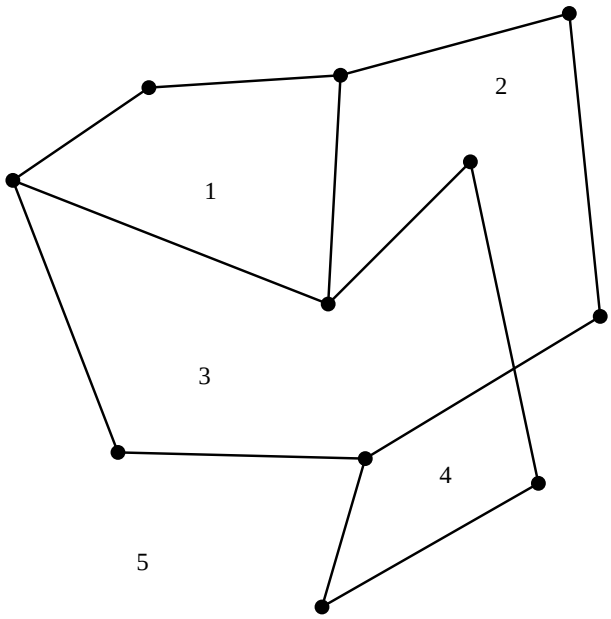

Figure 9.2: *The general point location problem defined by a set of (possibly intersecting) line segments. In this example, the segments partition the plane into 5 cells.*

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 9.1. Now we insert segment s_r (see Figure 9.4).

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 9.5 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.

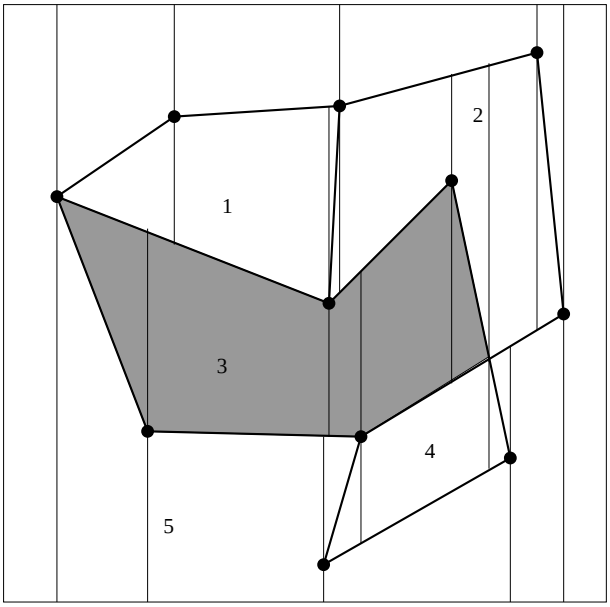

Figure 9.3: *The trapezoidal map is a refinement of the partition of the plane into cells. For example, cell 3 is a union of five trapezoids.*

The **Merge** step is shown in Figure 9.6. 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).

9.4 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 9.2.

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 9.3. 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).

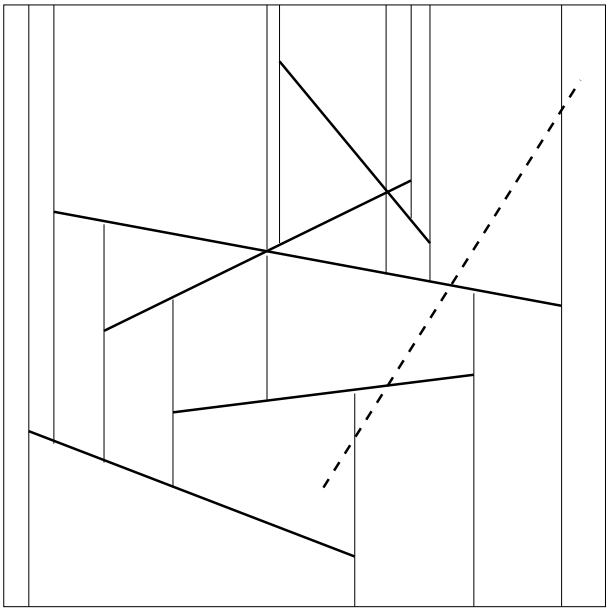

Figure 9.4: *A new segment (dashed) is to be inserted*

9.5 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.

9.5.1 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 9.2 *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

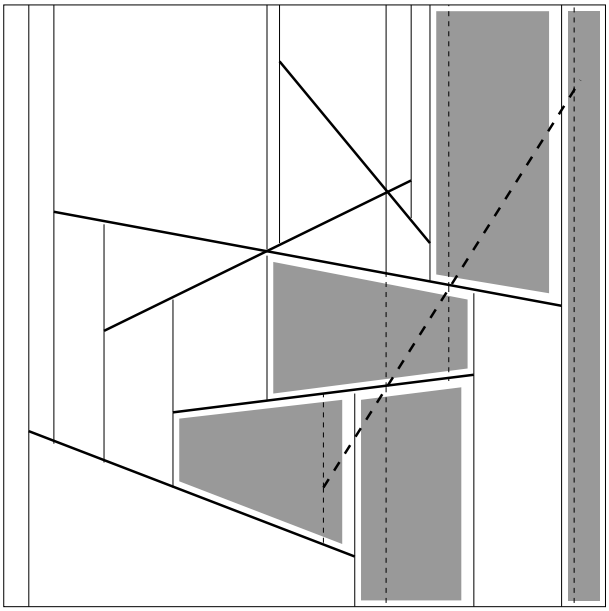

Figure 9.5: *The* **Trace** *and* **Split** *steps*

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 □. □

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 $\mathcal{T}_{r-1} \setminus \mathcal{T}_r$; 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 □ in Figure 9.7. 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.

To deal with this, we slightly adapt our notion of configuration.

Definition 9.3 *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*

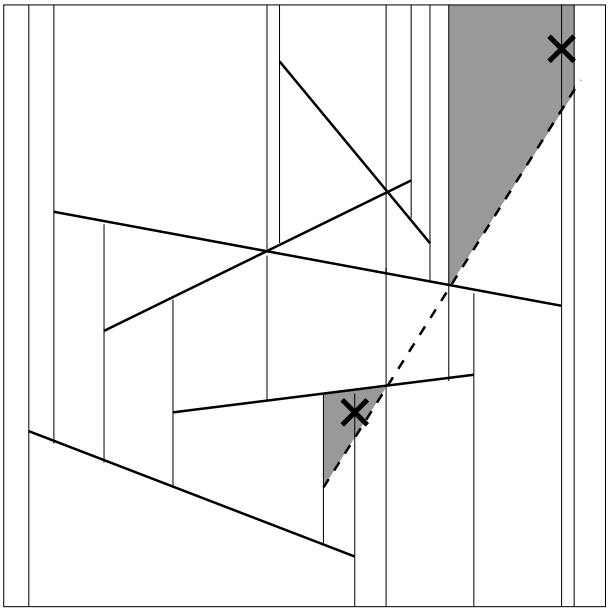

Figure 9.6: *The* **Merge** *steps*

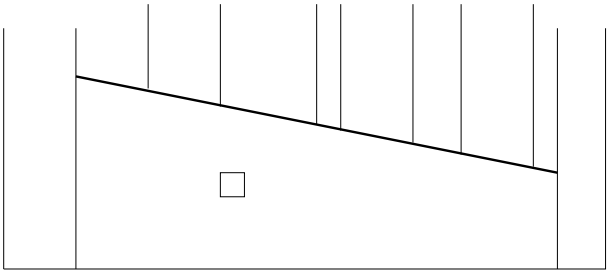

Figure 9.7: *Trapezoids may have arbitrarily large complexity*

subset of $X = S$*, see Figure 9.8. A trapezoid without any incident vertical edge is also considered a trapezoid with tail.*

As it turns out, we still have constantly many generators.

Lemma 9.4 *For every trapezoid with tail* $\square$ *in the trapezoidal map of* $R \subseteq S$*, there exists a set* $D \subseteq R$ *of at most six segments, such that* $\square$ *is in the trapezoidal map of* D*.*

Proof. We already know from Lemma 9.2 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. $\square$

Here is the complete specification of our configuration space $\mathcal{S} = (X, \Pi, D, K)$.

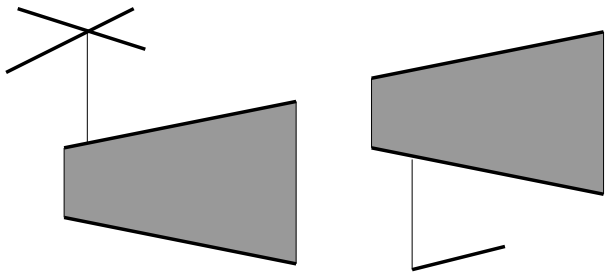

Figure 9.8: *A trapezoid with tail is a trapezoid together with at most one vertical edge attached to its upper or its lower segment.*

Definition 9.5 *Let $X = S$ be the set of segments, and Π the set of all trapezoids with tail. For each trapezoid with tail $\square$, $D(\square)$ is the set of at most 6 generators. $K(\square)$ is the set of all segments that intersect $\square$ 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 9.9.*

Then $\mathcal{S} = (X, \Pi, D, K)$ is a configuration space of dimension at most 6, by Lemma 9.4. The only additional property that we need to check is that $D(\square) \cap K(\square) = \emptyset$ for all trapezoids with tail, but this is clear since no generator of $\square$ properly intersects the trapezoid of $\square$ or cuts off part of its tail.

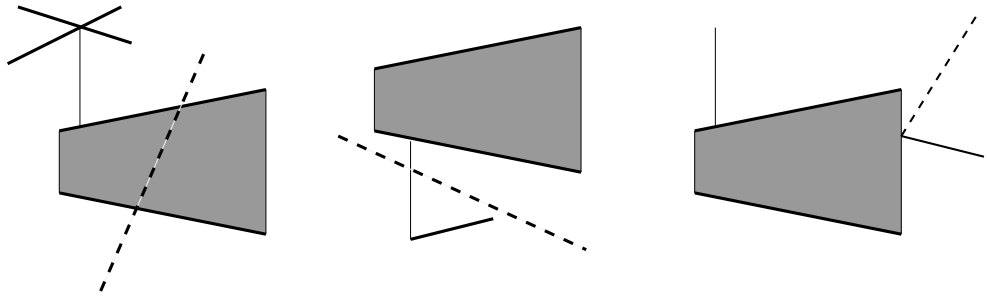

Figure 9.9: *A trapezoid with tail $\square$ is in conflict with a segment s (dashed) if s intersects $\square$ in the interior of the trapezoid (left), or cuts off part of the tail (middle), or is a new topmost segment generating a vertical side.*

9.5.2 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 9.6 *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 9.7 *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.

9.5.3 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.

9.5.4 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 8.5, 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 9.8 *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 9.9 *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 $\mathcal{T}_r$.

9.5.5 Applying the General Bounds

Let us now apply Theorem 9.7 and Theorem 9.9 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 9.10 *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 9.11 *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 9.12 *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 9.10, Observation 9.11 and Lemma 9.12, we obtain the following

Corollary 9.13 *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 9.7 and Theorem 9.9, we obtain the following final

Theorem 9.14 *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 9.7 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 9.9 is

$$O\left(n\sum_{r=1}^{n} \frac{t_r}{r^2}\right) = O(n \log n) + O\left(\frac{K}{n}\sum_{r=1}^{n} 1\right) = O(n \log n + K).$$

□

9.6 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 9.15 *Let* $S = \{s_1, \ldots, s_n\}$ *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 9.15 is by a typical application of Chernoff's bound followed by the union bound.

Recall (or please meet) Chernoff's bound:

Lemma 9.16 *Let $X_1, X_2, \ldots, X_n$ be independent 0/1 random variables, and let $X = X_1 + \cdots + X_n$. Let $p_i = \Pr[X_i = 1]$, and let $\mu = \mathrm{E}[X] = p_1 + \cdots + p_n$. 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 9.15:

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 $s_1, s_2, \ldots, s_n$ 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 8.3, we "run the movie backwards": To obtain $\mathcal{T}_{r-1}$ from $\mathcal{T}_r$, we delete a random segment from among $s_1, \ldots, s_r$; 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 $\mu = \mathrm{E}[X] = \sum_{r=1}^{n} \Pr[A_r] = 4 \ln n + O(1)$. 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. □

9.7 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 9.10. 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.

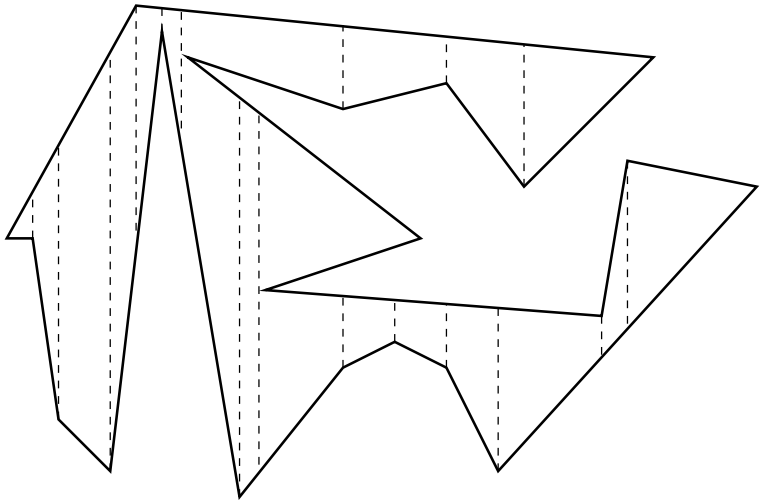

Figure 9.10: *The trapezoidal map inside a simple polygon*

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 9.24. This immediately allows us to improve over the statement of Exercise 2.19.

Corollary 9.17 *A simple polygon with n vertices can be triangulated in expected time $O(n \log n)$.*

Proof. By Theorem 9.14, 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 9.18 *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 9.19 *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 9.3.

By Lemma 9.6 (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 9.19 gives us in expectation.

Lemma 9.20 *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 9.19, the expected time is bounded by $O(n + \ell(r))$, where

$$\ell(r) = \frac{1}{\binom{n}{r}} \sum_{R \subseteq X, |R|=r} \sum_{y \in X \setminus R} |\{\square \in \mathcal{T}(R) : y \in K(\square)\}|.$$

This looks very similar to the bound on the quantity $k(r)$ that we have derived in Section 8.5 to count the expected number of conflicts in general configuration spaces:

$$k(r) \leqslant \frac{1}{\binom{n}{r}} \sum_{R \subseteq X, |R|=r} \frac{d}{r} \sum_{y \in X \setminus R} |\{\Delta \in \mathcal{T}(R) : y \in K(\Delta)\}|.$$

Indeed, the difference is only an additional factor of $\frac{d}{r}$ in the latter. For $k(r)$, we have then computed the bound

$$k(r) \leqslant k_1(r) - k_2(r) + k_3(r) \leqslant \frac{d}{r}(n-r)t_r - \frac{d}{r}(n-r)t_{r+1} + \frac{d^2}{r(r+1)}(n-r)t_{r+1}, \quad (9.21)$$

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 9.20, 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 9.7. It remains to analyze the **Find** step, and this is where the history flattening leads to a speedup. Adapting Lemma 9.8 and its proof accordingly, we obtain the following.

Lemma 9.22 *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 9.8, 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

$$\kappa(h) = \sum_{r=N(h-1)}^{N(h)-1} k_h(r).$$

[1] this amounts to $O(n)$ throughout the algorithm and can therefore be neglected

As in (9.21), we get

$$k_h(r) \leqslant \frac{d}{r}(N(h)-r)t_r - \frac{d}{r}(N(h)-r)t_{r+1} + \frac{d^2}{r(r+1)}(N(h)-r)t_{r+1},$$

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 9.18 follows.

Exercise 9.23 *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 9.24 *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

35. *What is the definition of a trapezoidal map?*

36. *How does the random incremental construction of a trapezoidal map proceed?* What are the main steps to be executed at each iteration?

37. *How can trapezoidal maps be used for the point location problem?*

38. *What is the configuration space framework?* Recall Section 8.3.

39. *What is a naive way of defining a configuration in the case of trapezoids, and why does it fail?*

40. *What is a more successful way of defining a configuration?* Why do things work in this case?

41. *What is the history graph, and how is it used to answer point location queries?*

42. *What is the performance of the random incremental construction of the trapezoidal map when used for the point-location problem?* Be precise!

43. *What probabilistic techniques are used in proving this performance bound?*

44. *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.*