Chapter 8

Voronoi Diagrams

8.1 The Post Office Problem

Suppose there are n post offices in a city, and a citizen would like to know which one is closest to him.[1] Modeling the city in the plane, we think of the post offices as a point set $P = \{p_1, \dots p_n\} \subset \mathbb{R}^2$, and the query location as a point $q \in \mathbb{R}^2$. The task is to find $p_i \in P$ that minimizes $\|p_i - q\|$.

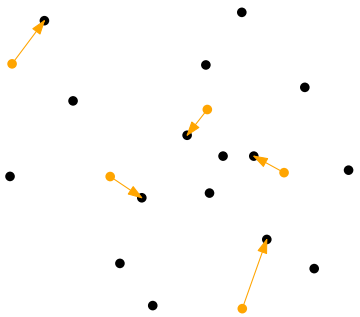

Figure 8.1: *Closest post offices for various query points.*

While the post offices P are considered stable, the query point q is not known in advance and can be changing frequently. Therefore, our long term goal is to come up with a (static) data structure on top of P that allows to answer *any* possible query efficiently.

As there can be only n possible answers, the idea is to apply the so-called *locus approach*: we subdivide the query space (in our case $\mathbb{R}^2$) into n regions according to the answer; the i-th region contains all points for which p_i is the closest. The resulting structure is called a *Voronoi diagram*; see Figure 8.2 for an example.

[1] Another—possibly historically more accurate—way to think of the problem: You want to send a letter to a person living in the city. For this you should know his zip code, which is the code of the post office closest to him.

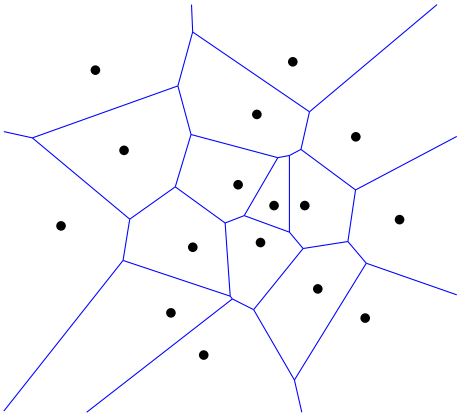

Figure 8.2: *The Voronoi diagram of a point set.*

Let us remark right away that such approach works for a variety of distance functions and spaces [2, 7]. So Voronoi diagram can be viewed as a broadly applicable paradigm. Without further qualification, the underlying distance function is Euclidean.

What exactly does a Voronoi diagram look like? As a warmup, suppose there are only two post offices: $P = \{p, p'\}$. Then the plane subdivides into two regions delimited by the *bisector* of p and p', i.e. the points that are equidistant to p and p'. The following proposition characterizes the shape of the bisector.

Proposition 8.1. *The bisector of two distinct points $p, p' \in \mathbb{R}^d$ is a hyperplane (a line when $d = 2$). It is orthogonal to the line pp' and goes through the midpoint of $\overline{pp'}$.*

Proof. Let us understand points as column vectors, so for any points $a = (a_1, \ldots, a_d)$ and $b = (b_1, \ldots, b_d)$ in $\mathbb{R}^d$ we have the identity $\|a-b\|^2 = \sum_{i=1}^d (a_i - b_i)^2 = \sum_{i=1}^d a_i^2 - 2\sum_{i=1}^d a_i b_i + \sum_{i=1}^d b_i^2 = \|a\|^2 - 2a^\top b + \|b\|^2$.

The bisector of p, p', by definition, consists of all points $x \in \mathbb{R}^d$ such that

$$\begin{aligned} \|p - x\| = \|p' - x\| &\iff \|p - x\|^2 = \|p' - x\|^2 \\ &\iff \|p\|^2 - 2p^\top x + \|x\|^2 = \|p'\|^2 - 2p'^\top x + \|x\|^2 \\ &\iff 2(p' - p)^\top x = \|p'\|^2 - \|p\|^2. \end{aligned}$$

As $p \neq p'$, this is the equation of a hyperplane orthogonal to the vector $p' - p$ (hence the line pp'). One can easily verify that the midpoint $x = (p + p')/2$ fits in the equation. □

Let us then denote by $H(p, p')$ the closed halfspace bounded by the bisector of p, p' that contains p. In this chapter we only study $\mathbb{R}^2$, so $H(p, p')$ is a halfplane (Figure 8.3). As we noted earlier, when there are only two post offices p and p', the plane is subdivided by $H(p, p')$ and $H(p', p)$.

Exercise 8.2.

(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 $\|x - p\| = \min_{r \in \ell} \|x - r\|$?

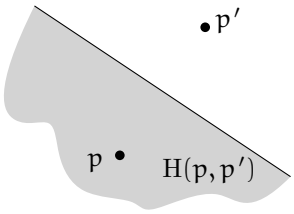

Figure 8.3: *The bisector of two points in* $\mathbb{R}^2$.

(b) For two distinct points $p, p' \in \mathbb{R}^2$*, what is the region that contains all points whose distance to* p *is exactly twice their distance to* p'*?*

8.2 Voronoi Diagram

Understanding the situation for two points essentially teaches us the law for the general case. In the following we formally define and study the Voronoi diagram for a given point set $P = \{p_1, \ldots, p_n\} \subset \mathbb{R}^2$.

Definition 8.3. *For* $i \in \{1, \ldots, n\}$*, the* **Voronoi cell** *of point* p_i *is defined as*

$$V_P(i) := \left\{ q \in \mathbb{R}^2 : \|q - p_i\| \leqslant \|q - p_j\|,\ \forall j \in \{1, \ldots, n\} \right\}.$$

Observe that (1) each Voronoi cell is non-empty since $p_i \in V_P(i)$; (2) the interiors of the cells are disjoint; and (3) the cells cover the entire plane. So these cells form a *subdivision* of the plane. It turns out that every cell looks quite regular:

Proposition 8.4. *For every* $i \in \{1, \ldots, n\}$*,*

$$V_P(i) = \bigcap_{j \neq i} H(p_i, p_j)\ .$$

In particular, it is a convex set whose boundary is piecewise linear (i.e. consisting of segments, rays or lines).

Proof. For every $j \neq i$, we have $\|q - p_i\| \leqslant \|q - p_j\|$ if and only if $q \in H(p_i, p_j)$. Hence $V_P(i)$ is exactly the intersection of these halfplanes (which are all convex); this is a convex set with piecewise linear boundary. □

Hence, the intersection of two different Voronoi cells can only be a line/ray/segment (called a **Voronoi edge**), a point (called a **Voronoi vertex**), or empty. By running over every pair intersection, we collect a set $VV(P)$ Voronoi vertices and a set $VE(P)$ of Voronoi edges. We also denote by $VR(P)$ the set of all Voronoi cells. The **Voronoi diagram** $VD(P)$ is formally defined as the tuple $(VV(P), VE(P), VR(P))$.

Lemma 8.5. *Every vertex* $v \in VV(P)$ *satisfies the following statements:*

(a) v *is incident to at least three cells from* $\mathrm{VR}(P)$*;*

(b) v *is the common endpoint of at least three edges from* $\mathrm{VE}(P)$*;*

(c) v *is the center of an empty circle* $C(v)$ *through at least three points from* P*; "empty" means that no point from* P *is strictly enclosed by* $C(v)$*.*

Proof. Consider a vertex $v \in \mathrm{VV}(P)$, which by definition is the intersection of some two Voronoi cells. But these two cannot be the only cells incident to v; otherwise due to convexity they should share an edge rather than a vertex. Hence v is incident to some $k \geqslant 3$ cells, proving (a).

Without loss of generality let these incident cells be $V_P(1), \dots V_P(k)$ in circular order; see Figure 8.4. Since the cells subdivide the plane, there is no gap in between, thus $e_i := V_P(i) \cap V_P(i \bmod k + 1)$ is indeed an edge for all $1 \leqslant i \leqslant k$. This proves (b).

Since $v \in V_P(i)$ for all $i \leqslant k$, the points $p_1, \dots, p_k$ are simultaneously closest to v. (Any p_j where $j > k$ is strictly farther away; for otherwise v should have been incident to $V_P(j)$, too.) With r denoting this smallest distance, we have $\|v - p_i\| = r < \|v - p_j\|$ for $1 \leqslant i \leqslant k < j \leqslant n$. In other words, $p_1, \dots, p_k$ are on an empty circle of radius r centered at v. This proves (c). □

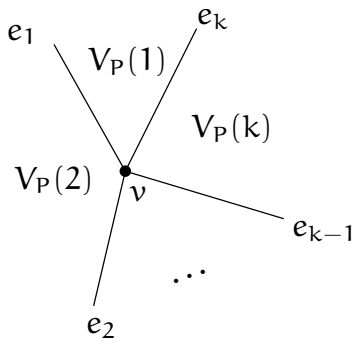

Figure 8.4: *Voronoi cells around* v*.*

Corollary 8.6. *If* P *is in general position (no four points cocircular), then every vertex* $v \in \mathrm{VV}(P)$ *satisfies the following statements:*

(a) v *is incident to exactly three cells from* $\mathrm{VR}(P)$*;*

(b) v *is the common endpoint of exactly three edges from* $\mathrm{VE}(P)$*;*

(c) v *is the center of an empty circle* $C(v)$ *through exactly three points from* P*.* □

Lemma 8.7. *There is an unbounded Voronoi edge shared by* $V_P(i)$ *and* $V_P(j)$*, if and only if* $\overline{p_i p_j} \cap P = \{p_i, p_j\}$ *and* $\overline{p_i p_j} \subseteq \partial\mathrm{conv}(P)$*.*

Proof. Denote by $b_{i,j}$ the bisector of p_i and p_j, and let $\mathcal{D}$ denote the family of circles with center on $b_{i,j}$ and passing through p_i, p_j. It is not hard to see that the following statements are equivalent:

- There is an unbounded Voronoi edge shared by $V_P(i)$ and $V_P(j)$.
- There is a ray $\rho \subset b_{i,j}$ such that for all $r \in \rho$ and $k \notin \{i, j\}$, we have $\|r - p_k\| > \|r - p_i\| = \|r - p_j\|$.
- There is a ray $\rho \subset b_{i,j}$ such that every circle $D \in \mathcal{D}$ with center on ρ is empty. (Figure 8.5)

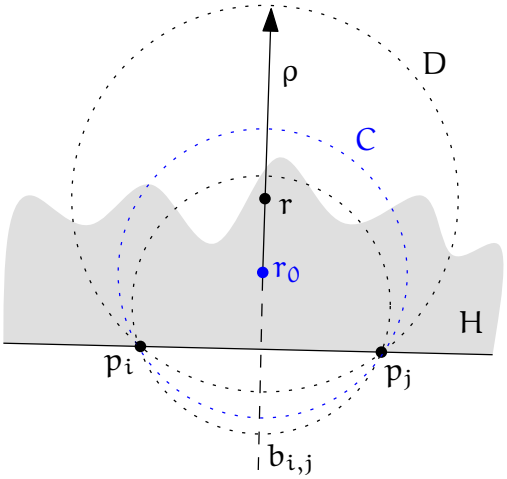

Figure 8.5: *The correspondence between $\overline{p_i p_j}$ appearing on $\partial\mathrm{conv}(P)$ and the existence of a divergent family of empty disks.*

Assuming the last statement, we have two observations. First, no points from P is on the segment $\overline{p_i p_j}$ except p_i, p_j. Second, the open halfplane H bounded by line $p_i p_j$ and containing the infinite part of ρ contains no point from P. Therefore $\overline{p_i p_j}$ appears on $\partial\mathrm{conv}(P)$.

Conversely, assume $\overline{p_i p_j} \cap P = \{p_i, p_j\}$ and $\overline{p_i p_j} \subseteq \partial\mathrm{conv}(P)$. Then one of the open halfplanes H bounded by line $p_i p_j$ contains no point from P. Since all points from $P \setminus \{p_i, p_j\}$ are strictly away from the segment $\overline{p_i p_j}$, there exists an empty circle $C \in \mathcal{D}$ provided its center r_0 is sufficiently far away from the line. Let $\rho \subseteq b_{i,j} \cap H$ be a ray emanating from r_0. Any circle $D \in \mathcal{D}$ centered on ρ encloses only a subset of $H \cup C$. As neither H nor C contains any point from P, the circle D is empty. This establishes the last statement, and the proof is complete. □

8.3 Duality With Delaunay Triangulations

A *straight-line dual* of a plane graph G is a geometric graph G' defined as follows: For each face of G, designate a point in $\mathbb{R}^2$ as its representative. Connect two representatives if their corresponding faces are adjacent in G.

Note that the notion depends heavily on the plane embedding G (the word "face" does not make sense for an abstract G), as well as the choice for representatives. In general, G' may have crossings.

Every Voronoi diagram can be treated as a plane graph. It is particularly natural to pick p_i as the representative for face $V_P(i)$. With this choice, the dual has no crossing and satisfies interesting properties. We thus call it *the* straight-line dual of a Voronoi diagram.

Theorem 8.8 (Delaunay [3]). *Let $P \subset \mathbb{R}^2$ be a set of $n \geqslant 3$ points in general position (no three points collinear and no four points cocircular). The straight-line dual of $\mathrm{VD}(P)$ is exactly the Delaunay graph of P.*

Proof. We write $G := \mathrm{VD}(P)$ (understood as a plane graph). Denote by G' its straight-line dual, and by D the Delaunay graph of P. Note that $V(G') = P = V(D)$. We aim to show $E(G') = E(D)$.

First we argue $E(G') \subseteq E(D)$. By construction of the dual, every edge $p_ip_j \in E(G')$ originates from adjacent cells $V_P(i), V_P(j)$ in G.

- If the cells share an unbounded Voronoi edge, then by Lemma 8.7, p_ip_j is on $\partial\mathrm{conv}(P)$ which is also contained in $E(D)$.
- Otherwise, the cells share a bounded Voronoi edge uv. By Corollary 8.6(b)(c), the Voronoi vertex v is incident to exactly three Voronoi cells $V_P(i)$, $V_P(j)$ and some $V_P(k)$, and the circle through p_i, p_j, p_k is empty. In particular p_ip_j is in $E(D)$ by Lemma 6.17.

Conversely we argue $E(G') \supseteq E(D)$. Any edge in $E(D)$ appears in some Delaunay triangle $p_ip_jp_k$ with empty circumcircle. The center v of the circle thus has p_i, p_j, p_k as its closest points. So v must be incident to the cells $V_P(i), V_P(j), V_P(k)$. Therefore, by construction of the dual we know that p_ip_j, p_jp_k, p_kp_i are edges in $E(G')$. □

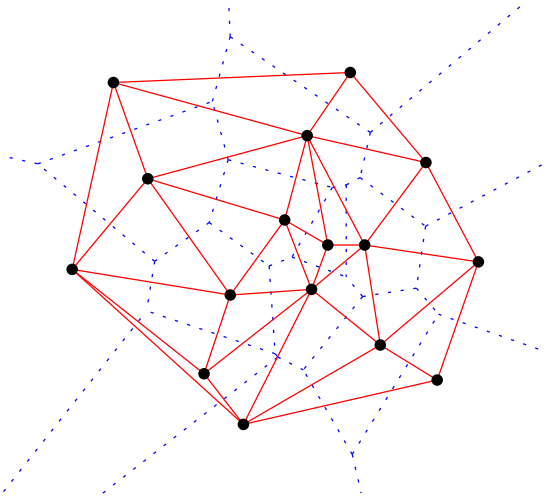

Figure 8.6: *The Voronoi diagram of a point set and its dual Delaunay triangulation.*

As a remark, the proof in fact establishes a correspondence between Voronoi vertices and Delaunay triangles: Given a Voronoi vertex, the three points from its incident

cells form a Delaunay triangle; vice versa, given a Delaunay triangle, the center of its circumcircle is a Voronoi vertex.

It is not hard to remove the general position assumption in Theorem 8.8. In this case, a Voronoi vertex of degree $k > 3$ corresponds in the dual to a convex k-gon with cocircular vertices. If we triangulate all these polygons in the dual arbitrarily, then we obtain a Delaunay triangulation of P. Therefore, the dual of the Voronoi diagram is a supergraph of a Delaunay triangulation.

Corollary 8.9. $|VE(P)| \leqslant 3n - 3 - h$ *and* $|VV(P)| \leqslant 2n - 2 - h$, *where* $h := |P \cap \partial\mathrm{conv}(P)|$ *is the number of points on the convex hull boundary.*

Proof. We assume general position (otherwise the proof can be adapted easily). Every Voronoi edge corresponds to an edge in the Delaunay triangulation. Every Voronoi vertex corresponds to a triangle in the Delaunay triangulation. So the counts follow from Lemma 6.4. □

Corollary 8.10. *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 for P can be obtained using randomized incremental construction within the asserted time and space bounds. Using the correspondence between Voronoi vertices/edges and Delaunay triangles/edges, we may generate the Voronoi diagram in $O(n)$ additional time and space. □

Exercise 8.11. *Consider the Delaunay triangulation* $\mathcal{T}$ *for a set* $P \subset \mathbb{R}^2$ *of* $n \geqslant 3$ *points in general position. Prove or disprove:*

(a) Every edge of $\mathcal{T}$ *intersects its dual Voronoi edge.*

(b) Every vertex of $VD(P)$ *is contained in its dual Delaunay triangle.*

Exercise 8.12. *Given a Voronoi diagram of some unknown point set* P*, can you compute* P *along with a Delaunay triangulation in linear time?*

8.4 A Lifting Map View

Recall that the lifting map $\ell\colon (x, y) \mapsto (x, y, x^2 + y^2)$ raises a point in the plane vertically to the unit paraboloid $\mathcal{U}$ in the space. We used it in Section 6.3 to prove that the Lawson Flip Algorithm terminates. Interestingly, it also provides a lens for the Voronoi diagram, which we now take an excursion to investigate.

For $p \in \mathbb{R}^2$ let $H_p \subseteq \mathbb{R}^3$ be the plane tangent to $\mathcal{U}$ at $\ell(p)$. We denote by $h_p(q)$ the vertical projection of a point $q \in \mathbb{R}^2$ onto the plane H_p (see Figure 8.7).

Lemma 8.13. $\|\ell(q) - h_p(q)\| = \|p - q\|^2$, *for any points* $p, q \in \mathbb{R}^2$.

Exercise 8.14. *Prove Lemma 8.13.* Hint: *First determine the equation of the plane* H_p *tangent to* $\mathcal{U}$ *at* $\ell(p)$.

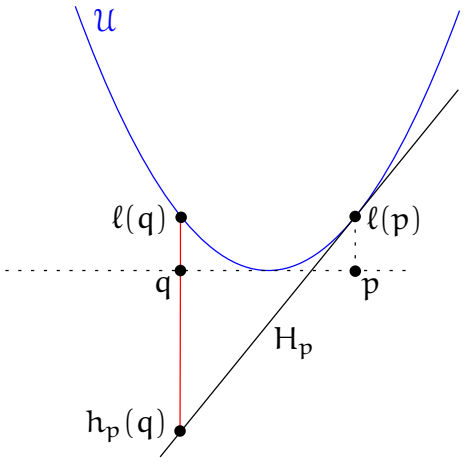

Figure 8.7: *A cross section of the lifting map interpretation for Voronoi diagram.*

Theorem 8.15. *Let H_p^+ be the closed halfspace above plane H_p. Define $\mathcal{H} := \bigcap_{p \in P} H_p^+$. Then the vertical projection of $\partial\mathcal{H}$ onto the xy-plane forms the Voronoi diagram of P. That is, the faces/edges/vertices of $\partial\mathcal{H}$ project to Voronoi cells/edges/vertices.*

Proof. Consider a point q' on the face defined by the plane H_p. Let $q \in \mathbb{R}^2$ be its vertical projection onto the xy-plane, so $q' = h_p(q)$. Note that $\ell(q)$ is above q', while all planes other than H_p are below q', thus $\|\ell(q) - h_p(q)\| \leqslant \|\ell(q) - h_r(q)\|$ for all $r \in P$. By Lemma 8.13, p is the closest point from P to q. □

8.5 Planar Point Location

One last bit is still missing in order to solve the post office problem optimally.

Theorem 8.16. *Given a triangulation $\mathcal{T}$ for a set $P \subset \mathbb{R}^2$ of n points, one can build in $O(n)$ time and space a data structure which allows for finding in $O(\log n)$ time a triangle $\Delta \in \mathcal{T}$ that contains any given query point $q \in \text{conv}(P)$.*

The data structure is known as *Kirkpatrick's hierarchy*. Before discussing it in detail, let us put things together to solve the post office problem.

Corollary 8.17 (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 which allows for reporting in $O(\log n)$ time a nearest point $p \in P$ to any given query point $q \in \text{conv}(P)$.*

Proof. First we construct the Voronoi diagram $\text{VD}(P)$ in expected $O(n \log n)$ time; see Corollary 8.10. It has exactly n convex cells. Truncate every unbounded cell by $\partial\text{conv}(P)$ into a bounded one, since we are concerned with query points within $\text{conv}(P)$ only.[2] Now

[2] We even know how to decide in $O(\log n)$ time whether or not a given point lies within $\text{conv}(P)$, see Exercise 5.28.

that all the cells are convex polygons, we may triangulate all of them in overall $O(n)$ time (the procedure only traverses each edge twice, and the number of edges is $O(n)$ by Corollary 8.9). We put a label "p_i" on all triangles in cell $V_p(i)$. Now we have a triangulation of the point set P. Apply Theorem 8.16 to build a Kirkpatrick's hierarchy, which takes $O(n)$ time and space.

When we receive a query point $q \in \text{conv}(P)$, we use the data structure to find in $O(\log n)$ time a triangle containing q. Output the label of the triangle, which is exactly the nearest point from P to q. □

8.6 Kirkpatrick's Hierarchy

We will now develop a data structure for point location in a triangulation, as described in Theorem 8.16. For simplicity we assume that the triangulation $\mathcal{T}$ we work with is maximal planar, that is, the outer face is a triangle as well. This can easily be achieved by wrapping a huge triangle Δ around $\mathcal{T}$ and triangulating the vacuum in between Δ and $\mathcal{T}$ in linear time (how?).

The main idea for the data structure is to construct a sequence

$$\mathcal{T} = \mathcal{T}_0,\ \mathcal{T}_1,\ \ldots,\ \mathcal{T}_{h-1},\ \mathcal{T}_h = \{\Delta\}$$

of triangulations such that the vertices of $\mathcal{T}_i$ are a proper subset of the vertices of $\mathcal{T}_{i-1}$, for $i = 1, \ldots, h$. Hence the triangulations get coarser as we move forward.

Given a query point q, we hunt for a triangle in $\mathcal{T}_0$ that contains q by tracing the sequence backwards:

- Start from the big triangle $\Delta \in \mathcal{T}_h$ which certainly contains q;
- Then find a triangle in the finer triangulation $\mathcal{T}_{h-1}$ that contains q;
- ...
- Finally, find a triangle in the target triangulation $\mathcal{T}_0$ that contains q.

Locating the query point.

1. Let $T_h := \Delta$.
2. For each $i = h, \ldots, 1$, examine all triangles in $\mathcal{T}_{i-1}$ that intersects T_i, until we find a triangle T_{i-1} that contains q.
3. Output T_0.

Proposition 8.18. *The search procedure can be implemented to use at most $3ch$ orientation tests, provided every triangle in $\mathcal{T}_i$ intersects at most c triangles in $\mathcal{T}_{i-1}$.*

Proof. In the data structure we link each triangle in $\mathcal{T}_i$ to at most c intersecting triangles in $\mathcal{T}_{i-1}$. With this implementation, step 2 examines at most ch triangles in total. For each triangle, three orientation tests suffice to determine if it contains q. □

We will show next how to construct the sequence so that both c and h are small. Concretely, we will make c a constant and $h = O(\log n)$.

Thinning. Suppose we have $\mathcal{T}_{i-1}$ at hand and want to construct $\mathcal{T}_i$ by removing several vertices and re-triangulating. Note that removing a vertex p (and its incident edges) from $\mathcal{T}_{i-1}$ creates a hole, which is a star-shaped polygon with p being its star-point.

Lemma 8.19. *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 8.20. *Prove Lemma 8.19.*

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].

Since we want h to be small, we had better remove a decent number of vertices. These vertices should have low degrees, since the degree is a natural upper bound for the number of triangles in $\mathcal{T}_{i-1}$ intersecting the triangles after re-triangulation.

Our working plan is thus to remove a constant proportion of *independent* (i.e. pairwise non-adjacent) low-degree vertices. The following lemma asserts the existence of such set of vertices in every triangulation.

Lemma 8.21. *In every triangulation of $n \geqslant 3$ points, there is a set of at least $\lceil n/18 \rceil$ independent vertices whose degrees are at most 8. Moreover, such a set can be found in $O(n)$ time.*

Proof. Let $G = (V, E)$ denote the graph of the triangulation, which we treat as an abstract planar graph. We may assume without loss of generality that G is maximal planar. (Otherwise use Theorem 2.33 to combinatorially triangulate G arbitrarily in linear time. Any independent set in the resulting graph is independent in the old graph, and the degree of a vertex can only increase.)

For $n = 3$ the statement is trivially true. Next assume $n \geqslant 4$. The total degree of G is $\sum_{v \in V} \deg_G(v) = 2|E| < 6n$ by Corollary 2.5. On the other hand, G is 3-connected by Theorem 2.30, so every vertex has degree at least 3. Let $W \subseteq V$ denote the set of vertices of degree at most 8. Then we have

$$\begin{aligned} 6n > \sum_{v \in V} \deg_G(v) &= \sum_{v \in W} \deg_G(v) + \sum_{v \in V \setminus W} \deg_G(v) \\ &\geqslant 3|W| + 9(n - |W|) = 9n - 6|W|, \end{aligned}$$

hence $|W| > n/2$.

Let us pick an independent set greedily: In each iteration, pick a remaining vertex in W, then eliminate itself and its neighbors. Repeat until all vertices in W have been eliminated.

By construction, the picked vertices are independent and have degrees at most 8. Each iteration eliminates at most nine vertices (the picked vertex and its at most eight neighbors) from W, so upon termination we have picked at least $|W|/9 \geqslant \lceil n/18 \rceil$ vertices.

Regarding the running time, if G is represented by adjacency lists, for example, we can obtain the neighborhood of any vertex $v \in W$ in $\deg_G(v) = O(1)$ time. As there are at most $|W|$ iterations, the greedy procedure runs in overall $O(n)$ time. □

Proof of Theorem 8.16. We construct the hierarchy $T_0, \dots T_h$ iteratively. Let $T_0 = T$, and derive T_i from T_{i-1} as follows. We remove an independent set of T_{i-1} provided by Lemma 8.21. This creates several holes, each being a star-shaped polygon. We re-triangulate the holes by Lemma 8.19, and the result is T_i. Each new triangle in T_i keeps pointers to the intersecting triangles in T_{i-1}; the number of needed pointers is at most $c = 8$.

The above steps cost time linear n_i, the number of vertices in T_i. Since at least $\lceil n_{i-1}/18 \rceil$ vertices are removed, we have

$$n_i \leqslant \frac{17}{18} n_{i-1} \leqslant \cdots \leqslant \left(\frac{17}{18}\right)^i n$$

Therefore, the total cost for building the hierarchy is proportional to

$$\sum_{i=0}^{h} n_i \leqslant n \cdot \sum_{i=0}^{h} \left(\frac{17}{18}\right)^i < n \cdot \sum_{i=0}^{\infty} \left(\frac{17}{18}\right)^i = 18n \in O(n).$$

Similarly the space consumption is linear.

The number of levels amounts to $h = \log_{18/17} n$. Thus by Proposition 8.18 each query needs at most $3ch = 3 \cdot 8 \cdot \log_{18/17} n < 292 \log n$ 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 query time is somewhat formidable. 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]: $1 \log n + 2\sqrt{\log n} + O(\sqrt[4]{\log n})$.

Comparison with history graph. Similar to Kirkpatrick's hierarchy, the history graph for the incremental construction (Chapter 7) is also used to locate query points in a triangulation. But the two data structures have fundamental differences. First, the history graph is built during the construction of a Delaunay triangulation, whereas Kirkpatrick's hierarchy is built on top of any given triangulation (in our case a triangulation of the Voronoi diagram). Second, the history graph does not guarantee a logarithmic time for an arbitrary query point—not even in the probabilistic sense. In fact, the analysis there only bounds the expected total running time over all rounds, and the random choice of insertion (=query) points turns out crucial.

Exercise 8.22. *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 okay). Is it possible to move the disk out of these obstacles? See Figure 8.8.*

More formally, the question is whether there is a continuous curve $\gamma : [0,1] \to \mathbb{R}^2$ 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.

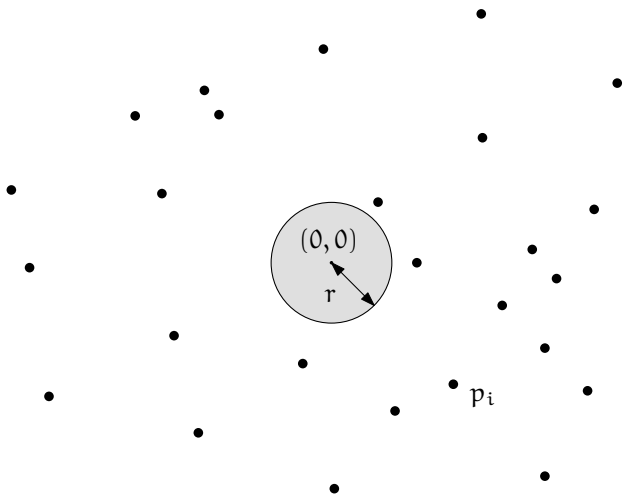

Figure 8.8: *Motion planning: Illustration for Exercise 8.22.*

Exercise 8.23. *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.*

Exercise 8.24. *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 8.25. *Show that the bounds given in Theorem 8.16 are optimal in the algebraic computation tree model.*

Questions

34. *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 8.5, Lemma 8.7.

35. *What is the correspondence between the Voronoi diagram and the Delaunay triangulation for a set of points in $\mathbb{R}^2$?* Prove duality (Theorem 8.8) and explain where general position is needed.

36. *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.

37. *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.

38. *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 8.21 and Theorem 8.16.

39. *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.

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.