The data structure we will employ is known as *Kirkpatrick's hierarchy*. But before discussing it in detail, let us put things together in terms of the post office problem.

Corollary 8.18 (Nearest Neighbor Search) *Given a set $P \subset \mathbb{R}^2$ of n points, one can build in expected $O(n \log n)$ time an $O(n)$ size data structure that allows for any query point $q \in \text{conv}(P)$ to find in $O(\log n)$ time a nearest neighbor of q among the points from P.*

Proof. First construct the Voronoi Diagram V of P in expected $O(n \log n)$ time. It has exactly n convex faces. Every unbounded face can be cut by the convex hull boundary into a bounded and an unbounded part. As we are concerned with query points within $\text{conv}(P)$ only, we can restrict our attention to the bounded parts.[3] Any convex polygon can easily be triangulated in time linear in its number of edges (= number of vertices). As V has at most $3n - 6$ edges and every edge appears in exactly two faces, V can be triangulated in $O(n)$ time overall. Label each of the resulting triangles with the point from p, whose Voronoi region contains it, and apply the data structure from Theorem 8.17. □

8.5.1 Kirkpatrick's Hierarchy

We will now the develop the data structure for point location in a triangulation, as described in Theorem 8.17. For simplicity we assume that the triangulation T we work with is a maximal planar graph, that is, the outer face is a triangle as well. This can easily be achieved by an initial normalization step that puts a huge triangle T_h around T and triangulates the region in between T_h and T (in linear time—how?).

The main idea for the data structure is to construct a hierarchy $T_0, \ldots, T_h$ of triangulations, such that

- $T_0 = T$,
- the vertices of T_i are a subset of the vertices of T_{i-1}, for $i = 1, \ldots, h$, and
- T_h is a single triangle only.

Search. For a query point x we can find a triangle from T that contains x as follows.

```
Search($x \in \mathbb{R}^2$)
1. For $i = h, h-1, \ldots, 0$: Find a triangle $t_i$ from $T_i$ that contains $x$.
2. return $t_0$.
```

This search is efficient under the following conditions.

[3]We even know how to decide in $O(\log n)$ time whether or not a given point lies within $\text{conv}(P)$, see Exercise 4.22.

(C1) Every triangle from T_i intersects only few ($\leqslant c$) triangles from T_{i-1}. (These will then be connected via the data structure.)

(C2) h is small ($\leqslant d \log n$).

Proposition 8.19 *The search procedure described above needs $\leqslant 3cd \log n = O(\log n)$ orientation tests.*

Proof. For every T_i, $0 \leqslant i < h$, at most c triangles are tested as to whether or not they contain x. Using three orientation tests one can determine whether or not a triangle contains a given point. □

Thinning. Removing a vertex v and all its incident edges from a triangulation creates a non-triangulated hole that forms a star-shaped polygon since all points are visible from v (the star-point). Here we remove vertices of constant degree only and therefore these polygons case are of constant size. But even if they were not, it is not hard to triangulate a star-shaped polygon in linear time.

Lemma 8.20 *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 8.21 *Prove Lemma 8.20.*

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 using linear programming.

Our working plan is to obtain T_i from T_{i-1} by removing several *independent* (pairwise non-adjacent) vertices and re-triangulating. These vertices should

a) have small degree (otherwise the degree within the hierarchy gets too large, that is, we need to test too many triangles on the next level) and

b) be many (otherwise the height h of the hierarchy gets too large).

The following lemma asserts the existence of a sufficiently large set of independent small-degree vertices in every triangulation.

Lemma 8.22 *In every triangulation of n points in $\mathbb{R}^2$ there exists an independent set of at least $\lceil n/18 \rceil$ vertices of maximum degree 8. Moreover, such a set can be found in $O(n)$ time.*

Proof. Let $T = (V, E)$ denote the graph of the triangulation, which we consider as an abstract graph in the following. We may suppose that T is maximal planar, that is, the outer face is a triangle. (Otherwise combinatorially triangulate it arbitrarily. An independent set in the resulting graph is also independent in T.) For $n = 3$ the statement is true. Let $n \geqslant 4$.

By the Euler formula we have $|E| = 3n - 6$, that is,

$$\sum_{v \in V} \deg_T(v) = 2|E| = 6n - 12 < 6n.$$

Let $W \subseteq V$ denote the set of vertices of degree at most 8. Claim: $|W| > n/2$. Suppose $|W| \leqslant n/2$. By Theorem 2.26 we know that T is 3-connected and so every vertex has degree at least three. Therefore

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

in contradiction to the above.

Construct an independent set U in T as follows (greedily): As long as $W \neq \emptyset$, add an arbitrary vertex $v \in W$ to U and remove v and all its neighbors from W.

Obviously U is independent and all vertices in U have degree at most 8. At each selection step at most 9 vertices are removed from W. Therefore $|U| \geqslant \lceil (n/2)/9 \rceil = \lceil n/18 \rceil$. □

Proof. (of Theorem 8.17)
Construct the hierarchy $T_0, \dots T_h$ with $T_0 = T$ as follows. Obtain T_i from T_{i-1} by removing an independent set U as in Lemma 8.22 and re-triangulating the resulting holes. By Lemma 8.20 and Lemma 8.22 every step is linear in the number $|T_i|$ of vertices in T_i. The total cost for building the data structure is thus

$$\sum_{i=0}^{h} \alpha|T_i| \leqslant \sum_{i=0}^{h} \alpha n (17/18)^i < \alpha n \sum_{i=0}^{\infty} (17/18)^i = 18\alpha n \in O(n),$$

for some constant α. Similarly the space consumption is linear.

The number of levels amounts to $h = \log_{18/17} n < 12.2 \log n$. Thus by Proposition 8.19 the search needs at most $3 \cdot 8 \cdot \log_{18/17} n < 292 \log n$ orientation tests. □

Improvements. As the name suggests, the hierarchical approach discussed above is due to David Kirkpatrick [5]. The constant 292 that appears in the search time is somewhat large. There has been a whole line of research trying to improve it using different techniques.

- Sarnak and Tarjan [6]: $4 \log n$.
- Edelsbrunner, Guibas, and Stolfi [3]: $3 \log n$.
- Goodrich, Orletsky, and Ramaiyer [4]: $2 \log n$.
- Adamy and Seidel [1]: $1 \log n + 2\sqrt{\log n} + O(\sqrt[4]{\log n})$.

Exercise 8.23 *Let $\{p_1, p_2, \dots, p_n\}$ be a set of points in the plane, which we call* obstacles. *Imagine there is a disk of radius r centered at the origin which can be moved around the obstacles but is not allowed to intersect them (touching the boundary is ok). Is it possible to move the disk out of these obstacles? See the example depicted in Figure 8.6 below.*

More formally, the question is whether there is a (continuous) path $\gamma : [0, 1] \longrightarrow \mathbb{R}^2$ with $\gamma(0) = (0, 0)$ and $\|\gamma(1)\| \geqslant \max\{\|p_1\|, \dots, \|p_n\|\}$, 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 $\{p_1, p_2, \dots, p_n\}$ and a radius $r > 0$. Argue why your algorithm is correct and analyze its running time.

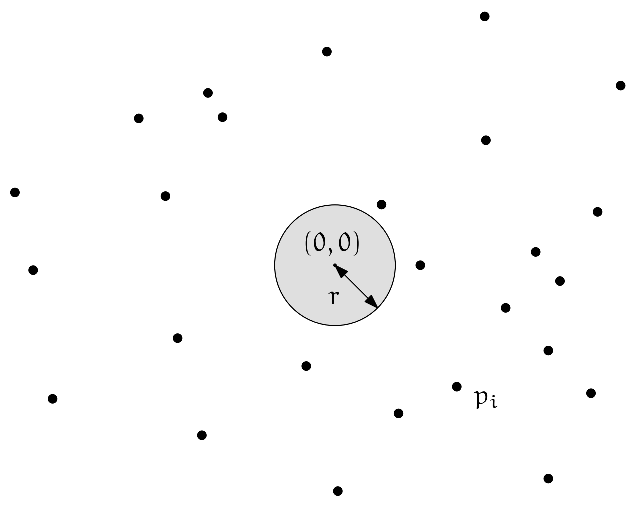

Figure 8.6: *Motion planning: Illustration for Exercise 8.23.*

Exercise 8.24 *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 8.25 *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 8.26 *Show that the bounds given in Theorem 8.17 are optimal in the algebraic computation tree model.*

Questions

33. *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 8.7, Lemma 8.9.

34. *What is the correspondence between the Voronoi diagram and the Delaunay triangulation for a set of points in $\mathbb{R}^2$?* Prove duality (Theorem 8.10) and explain where general position is needed.

35. *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.

36. *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.

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 8.22 and Theorem 8.17.

References

[1] Udo Adamy and Raimund Seidel, On the exaxt worst case query complexity of planar point location. *J. Algorithms*, **37**, (2000), 189–217, URL http://dx.doi.org/10.1006/jagm.2000.1101.

[2] Boris Delaunay, Sur la sphère vide. A la memoire de Georges Voronoi. *Izv. Akad. Nauk SSSR, Otdelenie Matematicheskih i Estestvennyh Nauk*, **7**, (1934), 793–800.

[3] Herbert Edelsbrunner, Leonidas J. Guibas, and Jorge Stolfi, Optimal point location in a monotone subdivision. *SIAM J. Comput.*, **15**, 2, (1986), 317–340, URL http://dx.doi.org/10.1137/0215023.

[4] 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, URL http://doi.acm.org/10.1145/314161.314438.

[5] David G. Kirkpatrick, Optimal search in planar subdivisions. *SIAM J. Comput.*, **12**, 1, (1983), 28–35, URL http://dx.doi.org/10.1137/0212002.

[6] Neil Sarnak and Robert E. Tarjan, Planar point location using persistent search trees. *Commun. ACM*, **29**, 7, (1986), 669–679, URL http://dx.doi.org/10.1145/6138.6151.