# 8. Point Location

Lecture on Thursday $15^{th}$ October, 2009 by Michael Hoffmann <hoffmann@inf.ethz.ch>

Our goal in the following is to provide the missing bit needed to solve the post office problem optimally.

**Theorem 8.1** *Given a triangulation* $T$ *for a set* $P \subset \mathbb{R}^2$ *of* $n$ *points, one can build in* $\mathcal{O}(n)$ *time an* $\mathcal{O}(n)$ *size data structure that allows for any query point* $q \in \text{conv}(P)$ *to find in* $\mathcal{O}(\log n)$ *time the triangle from* $T$ *containing* $q$.

**Corollary 8.2 (Nearest Neighbor Search)** *Given a set* $P \subset \mathbb{R}^2$ *of* $n$ *points, one can build in* $\mathcal{O}(n \log n)$ *time an* $\mathcal{O}(n)$ *size data structure that allows for any query point* $q \in \text{conv}(P)$ *to find in* $\mathcal{O}(\log n)$ *time the nearest neighbor of* $q$ *among the points from* $P$.

**Proof.** First construct the Voronoi Diagram V of P in $O(n \log n)$ time. It has exactly $n$ faces, all of which are convex polygons. 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 8.1. □

## 8.1 Kirkpatrick's Hierarchy

Idea: 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
- $T_h$ comprises a single triangle only.

**Search.** For a query point $x$ the triangle from T containing $x$ can be found as follows.

**Search**$(x \in \mathbb{R}^2)$

1. For $i = h \ldots 0$: Find a triangle $t_i$ from $T_i$ that contains $x$.
2. return $t_0$.

This search is efficient under the following conditions.

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

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

**Proposition 8.3** *The search procedure described above needs* $\leq 3cd \log n = \mathcal{O}(\log n)$ *orientation tests.*

**Proof.** For every $T_i$, $0 \leq i < h$, at most $c$ triangles are tested as to whether or not they contain $x$. □

**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 $v$.

**Lemma 8.4** *A starshaped polygon, given as a sequence of* $n \geq 3$ *vertices and a star-point can be triangulated in* $\mathcal{O}(n)$ *time.*

**Proof.** Exercise.

As a side remark, the *kernel* of a simple polygon, that is, the set of all star-points, can be constructed in linear time as well. □

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 re-triangulating the hole is too expensive) 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 any triangulation.

**Lemma 8.5** *In every triangulation of* $n$ *points in* $\mathbb{R}^2$ *there exists an independent set of at least* $n/18$ *vertices of maximum degree 8. Moreover, such a set can be found in* $\mathcal{O}(n)$ *time.*

**Proof.** Let $T = (V, E)$ denote the graph of the triangulation. We may suppose that $T$ is maximally planar, that is, all faces are triangles. (Otherwise triangulate the exterior face arbitrarily. An independent set in the resulting graph is also independent in $T$.) For $n = 3$ the statement is true. Let $n \geq 4$.

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: $|W| \geq n/2$. Suppose $|W| < n/2$. As every vertex has degree at least three, we have

$$\sum_{v \in V} \deg_T(v) = \sum_{v \in W} \deg_T(v) + \sum_{v \in V \setminus W} \deg_T(v) \geq 3|W| + 9|V \setminus W| \geq 9n/2 + 3n/2 = 6n,$$

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 $|U| \geq (n/2)/9 = n/18$. □

**Proof.** (of Theorem 8.1)

Construct the hierarchy $T_0, \ldots T_h$ with $T_0 = T$ as follows. Obtain $T_i$ from $T_{i-1}$ by removing an independent set $U$ as in Lemma 8.5 and re-triangulating the resulting holes. By Lemma 8.4 and Lemma 8.5 every step is linear in the number $|T_i|$ of vertices in $T_i$. The total cost for building the data structure is thus

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

for some constant $\alpha$. Similarly the space consumption is linear.

The number of levels amounts to $h = \log_{18/17} n < 12.2 \log n$. Thus by Proposition 8.3 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 [4]. 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 [5]: $4 \log n$.
- Edelsbrunner, Guibas, and Stolfi [2]: $3 \log n$.
- Goodrich, Orletsky, and Ramaier [3]: $2 \log n$.
- Adamy and Seidel [1]: $1 \log n + 2\sqrt{\log n} + O(\sqrt[4]{\log n})$.

## Questions

22. *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.

23. *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 8.4, Lemma 8.5, and Theorem 8.1.

## 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] H. Edelsbrunner, Leonidas J. Guibas, and J. Stolfi, Optimal point location in a monotone subdivision, *SIAM J. Comput.* **15**, 2 (1986), 317–340.

[3] M. T. Goodrich, M. Orletsky, and K. Ramaiyer, Methods for Achieving Fast Query Times in Point Location Data Structures, in: *Proc. 8th ACM-SIAM Sympos. Discrete Algorithms*, 1997, 757–766.

[4] D. G. Kirkpatrick, Optimal search in planar subdivisions, *SIAM J. Comput.* **12**, 1 (1983), 28–35.

[5] N. Sarnak and R. E. Tarjan, Planar point location using persistent search trees, *Commun. ACM* **29**, 7 (1986), 669–679.