

7 Lifting

We will interpret the Delaunay triangulation in \mathbb{R}^2 by lifting it up to \mathbb{R}^3 . Also we will give a randomized algorithm for constructing D_S with expected running time $O(n \log n)$. Details descriptions can be found in most of computational geometry books including [1–5].

7.1 Lifting Map

We introduce a map $\lambda : \mathbb{R}^2 \rightarrow \mathbb{R}^3$ to bring the point set S in the plane to three dimensions. For every $p = (\phi_1, \phi_2)$, we define $\lambda(p) = (\phi_1, \phi_2, \phi_1^2 + \phi_2^2)$. We can think of placing all the points in S on the x_1x_2 -plane, and project every point up to the paraboloid $\pi : x_3 = x_1^2 + x_2^2$.

Lemma 7.1 For every circle c in \mathbb{R}^2 , there exists a plane $h \in \mathbb{R}^3$ so that $c = \lambda^{-1}(h \cap \pi)$.

PROOF. For any circle c in the form of $(x_1 - a_1)^2 + (x_2 - a_2)^2 = r^2$, the intersection of π and the plane h of $x_3 = 2a_1x_1 + 2a_2x_2 - (a_1^2 + a_2^2 - r^2)$ is the projection of the circle. \square

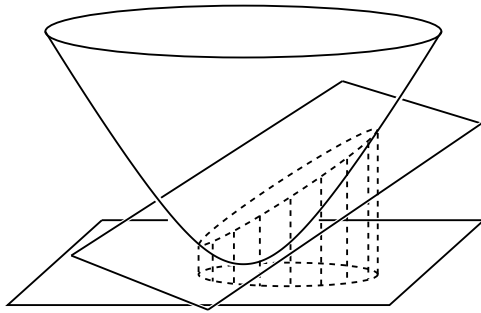


Figure 7.1: The intersection of any plane with π is a circle after projecting to \mathbb{R}^2 .

Figure 7.1 shows such a configuration. The hyperplane h divides the paraboloid into three parts. We name them the space above, on, and below the plane as h^+, h , and h^- respectively. The space in \mathbb{R}^2 is also divided into three parts, namely the region outside, on, and inside the circle. We also name these regions c^+, c and c^- respectively. Every point in c^+ is mapped to a point in the intersection of h^+ and the paraboloid. The fact is also true for

the other two pairs and we have

$$\begin{aligned} x \in c^+ &\Leftrightarrow \lambda(x) \in h^+ \cap \pi, \\ x \in c &\Leftrightarrow \lambda(x) \in h \cap \pi, \text{ and} \\ x \in c^- &\Leftrightarrow \lambda(x) \in h^- \cap \pi. \end{aligned}$$

7.2 Edge Flipping in \mathbb{R}^2

Recall that for two triangles pqr and pqs , we check if the edge pq is locally Delaunay. This is equivalent to check if the point s is outside the circumcircle c of triangle pqr . After the lifting operation, it is equivalent to check if $\lambda(s)$ is above the plane h . Figure 7.2 shows two examples. Imagine seeing the triangulation from below, a non-locally Delaunay edge is concave to the viewer.

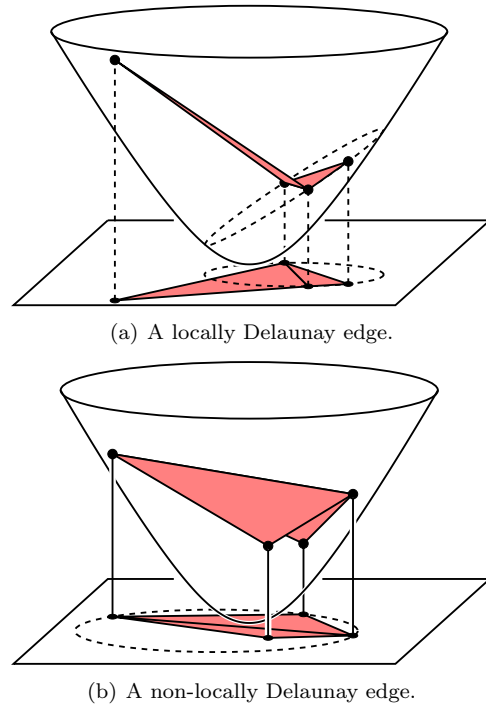


Figure 7.2: Two examples of lifted triangulation.

We return to the algorithm in Section 6.3 now. By flipping the edge pq to rs in \mathbb{R}^2 , it is equivalent to gluing the tetrahedron $\text{conv}(\{\lambda(p), \lambda(q), \lambda(r), \lambda(s)\})$ in \mathbb{R}^3 . To prove that the algorithm terminates, we can argue that the triangulation in \mathbb{R}^3 will simply run out of concave edges. The gluing also implies the number of flips cannot exceed the number of edges in \mathbb{R}^3 (why?),

and this means the worst running time cannot exceed $O(n^2)$. However, there are cases in which the algorithm takes $\Theta(n^2)$ flips such as Figure 6.8.

Primitive for locally Delaunay test. The *in-circle test* decides whether a point $s \in \mathbb{R}^2$ is inside or outside the circumcircle c_{pqr} of $p, q, r \in \mathbb{R}^2$. The Lemma 7.1 implies that this test is equivalent to check whether $\lambda(s) \in \mathbb{R}^3$ is above or below the plane $h = \text{aff}(\{\lambda(p), \lambda(q), \lambda(r)\})$. We use the notations of $p = (p_1, p_2)$, and also for q, r , and s . Then we can conclude that s is outside c_{pqr} iff

$$\begin{vmatrix} 1 & p_1 & p_2 & p_1^2 + p_2^2 \\ 1 & q_1 & q_2 & q_1^2 + q_2^2 \\ 1 & r_1 & r_2 & r_1^2 + r_2^2 \\ 1 & s_1 & s_2 & s_1^2 + s_2^2 \end{vmatrix} \cdot \begin{vmatrix} 1 & p_1 & p_2 \\ 1 & q_1 & q_2 \\ 1 & r_1 & r_2 \end{vmatrix} > 0.$$

Degenerate cases occur when the product is zero.

MaxMin angle property. The algorithm also helps us to prove one of the nice properties of Delaunay triangulation, namely, maximizing the minimum angle. In the configuration in Figure 7.3, the edge pq is flipped to rs . Before flipping, we have the six angles of the two triangles, namely, $\alpha_1, \delta_1 + \delta_2, \gamma_1, \gamma_2, \beta_1 + \beta_2$, and α_2 . We claim that for each angle after the flip, there is an angle in the old configuration which is smaller. It is simple that the two new combined angles $\alpha_1 + \alpha_2$ and $\gamma_1 + \gamma_2$ satisfy the claim. For the rest, we can see that $\delta_2 > \gamma_2, \beta_2 > \gamma_1, \delta_1 > \alpha_2$, and $\beta_1 > \alpha_1$. For example, the marked angle in Figure 7.3 is equal to δ_2 and larger than γ_2 .

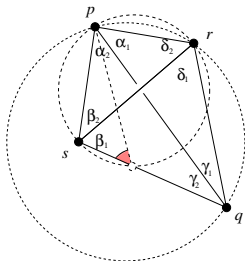


Figure 7.3: By flipping pq to rs , the minimum angle increases.

7.3 Incremental Algorithm

We will present a randomized incremental algorithm running in expected time $O(n \log n)$. After describing the algorithm and its data structure, we will give an analysis of its expected running time.

Algorithm. For a set of points in the plane $S = \{p_i \in \mathbb{R}^2 \mid i = 1..n\}$, we first find three points x, y, z in the plane such that they form a triangle which contains S . We add the point from S into this triangle one at a time and form the partial Delaunay triangulation. For step i , denote $S_i = \{x, y, z, p_1, p_2, \dots, p_i\}$, and let D_i be the Delaunay triangulation of S_i . For each step, we first locate the triangle σ that contains p_i and add p_i by splitting σ into three. After adding a point, flip the edges in the link of p_i to obtain D_i before adding the next point.

Data structure. There are two operations which change the structure of the triangulation, namely, adding a point and flip and edge. Instead of deleting the triangles, we add the new triangles formed as the children of the ‘deleted’ triangles. These operations form a directed acyclic graph (DAG) structure, see Figure 7.4.

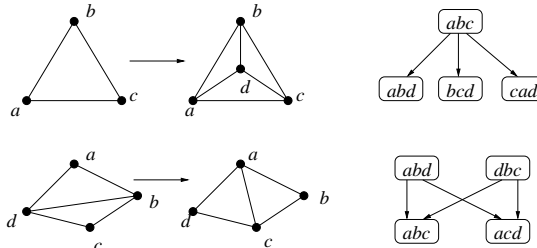


Figure 7.4: Splitting a triangle generates three children. Flipping an edge generates two children.

Edge flipping. We do not need to check too many edges after each point insertion. The only edges we that may have ‘problems’ are in the link of p_i . First, we push the three edges of $\text{lk}(p_i)$ onto a stack. Pop an edge ab and check for if it is locally Delaunay. One of the triangles sharing the edge is abp_i and let the other be abd . After flipping the edge ab we will push the edges ad and bd onto the stack. After the stack is cleared, all triangles

will be locally Delaunay. Observe that the star of p_i grows during the process, see Figure 7.5. The expected number of flip for each step is at most 3, thus, $O(1)$. □

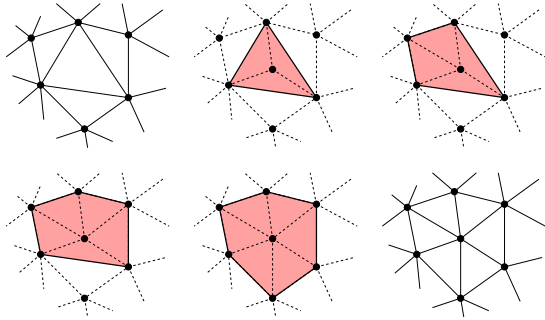


Figure 7.5: The star of p_i is growing during the process of edge flipping.

Locating the triangle. For each insertion at step i , we locate the triangle σ_i that contains p_i . We claim that for each insertion, we can locate the triangle σ_i in expected time $O(\log n)$. Hence, the overall expected time of the algorithm is $O(n \log n)$.

For the time of searching σ_i in step i , we need to traverse the DAG. Thus, the running time is proportional to the depth of the search. Lemma 7.2.

Lemma 7.2 For each $h < i$, let d_h denote the expected number of triangles in $D_h - D_{h-1}$ that are in conflict with p_i . Then

$$\sum_{h=1}^{i-1} d_h = O(\log i).$$

PROOF. A triangle in D_h whose circumcircles contain the new point p_h , are said to be *in conflict* with p_h . Let C denote the set of triangles of D_h that are in conflict with p_i . A triangle $\tau \in C$ happens if p_h is one of the vertex of τ with probability $3/h$. Thus the expected number of triangles in $C - D_{h-1}$ equals $3 \cdot c/h$ where $c = \text{card}(C)$. Since the expected value for c is less than 6 we have $d_h < 18/h$, hence

$$\begin{aligned} \sum_{h=1}^{i-1} d_h &< 18 \sum_{h=1}^{i-1} 1/h \\ &= \Theta(\log i). \end{aligned}$$

Homework Lite

1. What is the expected size of the DAG mentioned in Section 7.3 (5pts)?
2. Given four points in \mathbb{R}^2 , $p_1 : (1, 1)$, $p_2 : (0, 4.5)$, $p_3 : (5, 5)$ and $p_4 : (-5, 5)$, construct their Delaunay triangulation by using the bounding triangle abc with vertices $a : (-10, 0)$, $b : (10, 0)$ and $c : (0, 11)$. (Remember to randomize the points order.) Show every step of the construction and stack contents for each point addition. Finally, give comments on the Delaunay triangulation you have constructed (10pts).

References

- [1] DE BERG, M., VAN KREVELD, M., OVERMARS, M., AND SCHWARZKOPF, O. *Computational Geometry, Algorithms and Applications*. Springer-Verlag, 1997.
- [2] EDELSBRUNNER, H. *Algorithms in Computational Geometry*. Springer-Verlag, 1987.
- [3] EDELSBRUNNER, H. *Geometry and Topology for Mesh Generation*. Springer-Verlag, 2001.
- [4] PREPARATA, F. P., AND SHAMOS, M. I. *Computational Geometry, an Introduction*. Springer-Verlag, 1985.
- [5] SACK, J.-R., AND URRUTIA, J., Eds. *Handbook of Computational Geometry*. North-Holland, 2000.