Polyhedral Computation	2010/10/1
Lecture 1	
Professor: David Avis	*Scribe:Yuichi Yoshida*

In this course, we study geometric objects such as points, lines, planes, hyperplanes, polyhedra, and triangulations. *Polyhedral computation* studies algorithms working on those objects. The dimension is not usually considered fixed. When the dimension is fixed to two or three, this area is often called *computational geometry*.

In this first lecture, we study some examples for the two-dimensional case. Throughout this lecture, we assume that there are no three points on a line.

1 Convex Hull

The first problem we consider is the convex hull problem. A polygon P is called *convex* if every line segment between two vertices in P is also in P (Here, P includes both the boundary and the interior of the polygon). For a set S of n points $p_1, \dots, p_n \in \mathbb{R}^2$, we define the *convex hull* of S as the smallest convex polygon containing S, more precisely

$$\mathrm{CH}(S) = \bigcap_{\substack{P:\text{ convex polygon,}\\ P \supseteq S}} P.$$

See Fig. 1 for an example.

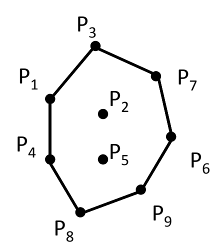

Figure 1: The convex hull of a point set $S = \{p_1, \dots, p_9\}$

The convex hull problem asks to compute $CH(S)$ from a set of points S. The output could be given in two ways.

1. The extreme points of $CH(S)$.
2. The edges of $CH(S)$.

Let S be the point set depicted in Fig. 1. Then, $\{p_1, p_3, p_4, p_6, p_7, p_8, p_9\}$ is one of the possible answers for the first case, and $\{p_1p_3, p_6p_7, p_3p_7, p_6p_9, p_1p_4, p_4p_8, p_8p_9\}$ is one of the possible answers for the second case.

It is clear that, if we are given the edges of $\mathrm{CH}(S)$, we can compute the extreme points of $\mathrm{CH}(S)$ in linear time. However, to get the edges from the unsorted vertex list requires

$\Omega(n \log n)$ time. We prove this by reducing the sorting problem for integers to this problem as follows.

Let $X = \{x_1, \ldots, x_n\}$ be a set of unsorted integers. Then, we generate a set of points $S = \{(x_i, x_i^2) \mid i = 1, \ldots, n\}$. Consider CH($S$). From the convexity of the function $y = x^2$, all points of S are extreme points of the convex hull. So problem (1) above is trivial, just return S. Note that the edges in order around the polygon give a sorting of the set X. Furthermore, given an unsorted edge list it is easy to obtain the edges of the convex hull in order, by using a double linked list. Therefore getting an unsorted edge list of CH(S) from S is at least as hard as sorting X. See Fig. 2 for an example for the case $X = \{3, 1, 4, 2, 6\}$ and $S = \{(3, 9), (1, 1), (4, 16), (2, 4), (6, 36)\}$.

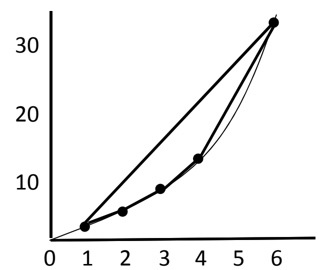

Figure 2: Sorting via a convex hull.

Since we need $\Omega(n \log n)$ time to sort n integers, we have the same lower bound to compute the convex hull of n points.

2 Delaunay Triangulation

Next, we consider Delaunay triangulations. Let S be a set of points. A *triangulation* of S is a subdivision of the convex hull of S into triangles in such a way that no two triangles intersect and the set of vertices of the triangles coincides with S. A triangulation is called satisfying the *empty circle condition* if, for every triangle in the triangulation, the circumcircle of the triangle does not contain any other point of S in its interior (see Fig. 3 for an example). A triangulation satisfying the empty circle condition is called a *Delaunay triangulation (DT)*. Note that if we take $n > 3$ points on a circle, then any triangulation of the n points is a DT, and so it is not unique. We can use lexicography to define a unique DT, but for simplicity let us just assume for now that no four input points lie on a circle.

It is known that for every set of points S, there exists a Delaunay triangulation. (why?) We can output CH(S) if we have the Delaunay triangulation of S since the Delaunay triangulation contains the edge set of CH(S).

3 Voronoi Diagram

Finally, we consider Voronoi diagrams. Let $S = \{p_1, \ldots, p_n\}$ be a set of n points in the plane. We define the distance between two points $p = (x_p, y_p)$ and $q = (x_q, y_q)$

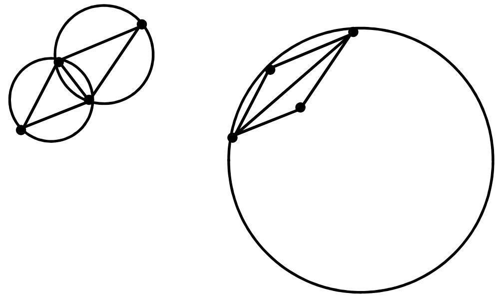

Figure 3: Left: A triangulation satisfying the empty circle condition. Right: A triangulation not satisfying the empty circle condition.

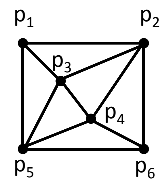

Figure 4: The Delaunay triangulation for a point set $S = \{p_1, \ldots, p_6\}$.

as $\sqrt{(x_p - x_q)^2 + (y_p - y_q)^2}$. Then, for each p_i, we define a region $V(p_i) = \{x \in \mathbb{R}^2 \mid d(x, p_i) \leq d(x, p_j),\ j = 1, ..., n\}$. We note that the definition of $V(p_i)$ does not change if we remove the square root from the distance function. We do this in practice as it makes the calculations more numerically stable. In words, a point x in $\mathbb{R}^2$ is contained in $V(p_i)$ if p_i is a closest point to x among the points in S. The decomposition of a plane into $V(p_1), \ldots, V(p_n)$ is called a *Voronoi diagram* (see Fig. 5 for an example). Each region $V(p_i)$ is a (possibly unbounded) convex polygon. Note that the unbounded regions correspond to points on the boundary of the convex hull. (why?) Also, the boundary between $V(P_i)$ and $V(P_j)$ is a part of the bisector between p_i and p_j.

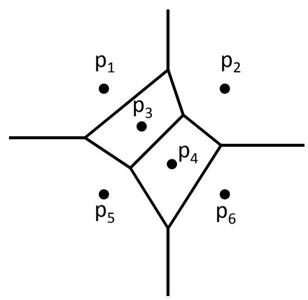

Figure 5: The Voronoi diagram for a point set $S = \{p_1, \ldots, p_6\}$.

Consider the triangulation obtained by joining points p_i and p_j iff $V(p_i)$ and $V(p_j)$ share an edge in common. We can see that the resulting triangulation coincides with the Delaunay triangulation of S. (why?) Using terms of graph theory, the graph constructed from boundaries of a Voronoi diagram is the dual of the graph constructed from a Delaunay triangulation. Thus, computing Voronoi diagrams and Delaunay triangulations are equivalent. In particular, we can compute $\mathrm{CH}(S)$ if we have the Voronoi diagram of S.

In general, we cannot directly construct the Voronoi diagram of S from $\mathrm{CH}(S)$. However, there is a way to construct the Voronoi diagram from the convex hull of a 3-dimensional polyhedron associated with S. This relation between convex hulls and Voronoi diagrams can be extended to higher dimensions and we will study this in a later lecture.