Computational Geometry. Lecture 5 and 6

Scribe Gaurav Tyagi

1 Convex Hull (CH)

1.1 Problem Defnition

Given a set P of n points in the plane,we want to compute the smallest *convex polygon* containing the points.

A polygon is convex if for any two given points a,b inside the polygon,the line segment ab is completely inside the polygon.

1.2 Graham's Scan

Using this algorithm each point in the set P is first sorted using their x-coordinate in *O(nlogn)* time and then applies a linear-time scan to finish building the hull.

For building the hull, we divide the points by a diagonal, so that we have upper hull and lower hull. We also rotate the hull so that the diagonal is parallel to x-axis. Now we need to apply a linear scan to compute the upper hull and lower hull of the polygon.

While building the hull, we will need to test whether three points (x0,y0),(x1,y1), and (x2,y2) form an indentation. Since the x-coordinates of all the points are ordered, all we need to do is test whether the middle point is above or below the line segment formed by the other two.

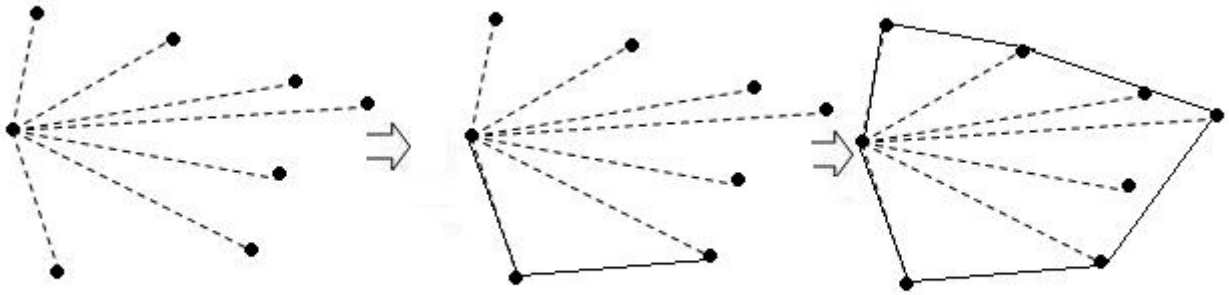
The triple points (p0,p1,p2) are said to form a right turn iff the determinant

$$\begin{vmatrix} x0 & y0 & 1 \\ x1 & y1 & 1 \\ x2 & y2 & 1 \end{vmatrix} < 0$$

where (x1,y1) are the co-ordinates of p1. If the determinant is positive, then the triple points form a left turn. If the determinant is 0, the points are collinear.

1.3 Jarvis March

Another algorithm for computing convex hulls which simply simulates package wrapping *(or gift wrapping)*. The algorithm runs in *O(nh)* time where h is the number of vertices of CH. When h is *O(logn)*, Jarvis march is asymptotically faster than Graham's scan.



The execution of Jarvis March

Jarvis march starts by computing the leftmost point l i.e. the point whose y-coordinate is smallest which takes linear time. The next convex hull vertex is picked by choosing the least polar angle wrt the positive horizontal ray from the first vertex.

1.4 Chan's Algorithm

This algorithm is a combination of divide-and-conquer and gift-wrapping. Here,we assume that we are given the value of h *(The number of vertices in the convex hull).*

1. We divide the axis into n/h equal size sets. (n/h sets of h points each)
2. Construct the Convex hulls $C_1, C_2, ...C_{n/h}$. using Graham's scan.
    - Starting with $p = l$, where l is the leftmost input point, we successively find the convex hull vertex that follows p. The vertex that follows p is the point that appears to be the furthest to the right from p. This means that the successor of p must lie on a *right tangent line* between p and one of the subhulls. We can find the right tangent line between p and any subhull in *O(log h)* time using a variant of binary search.
3. For each of the convex hull, we find the supporting line.
4. Find the edges of the convex hull by selecting appropriate supporting line. *(from n/h lines)*

1.4.1 Analysis of Chan's Algorithm

1. In order to divide into n/h sets, we do a simple division of n points into n/h sets. This will take $O(n)$
2. Cost of Graham scan. (n/h) sets. Each set takes $O(hlogh)$. Total time is $O(nlogh)$
3. Finding the supporting line for n/h sets takes $O((n/h)logh)$. For h vertices, it becomes $O(nlogh)$.

In order to find h we start by guessing its value. Initially, say $h = 2$. If by using this, we do not reach the end of the vertices, so we know that our guess of h is too small. Hence we *square* h and try again. Let us denote the guess by h^*. Then we divide the points into n/h^* subsets of size h^*, compute their subhulls, and then find the first h^* edges of the global hull.

The last iteration takes $O(nlogh)$
The one before that is $O(nlog\sqrt{h^*})$

Hence the total time
$= nlogh^* + nlog\sqrt{h^*} + nlog\sqrt{\sqrt{h^*}} + \dots$
$= (nlogh^*)\left(1 + \frac{1}{2} + \frac{1}{4} + \dots\right)$
$= (nlogh^*)\left(\frac{1}{1-\frac{1}{2}}\right)$
$= 2nlogh^* = O(nlogh)$

So Chan's algorithm runs in $O(nlogh)$ time.

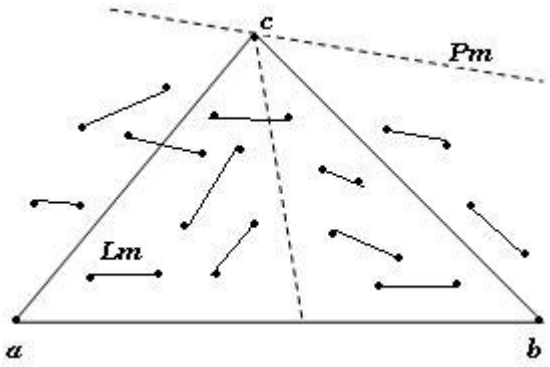
1.5 Quick-Hull

This is a divide-and-conquer algorithm that is based on the observation that we discard some of the points in the given set as interior points and concentrate on remaining points. For this we use the following approach.

1. Pair the points arbitrarily. Denote the lines through these points by $l_1, l_2, \dots, l_{n/2}$.
2. Choose the line with median slope, say l_m and find the extreme point, say p_m through which a supporting line passes parallel to l_m.
3. Draw a vertical line through p_m to partition the point set.
4. Prune some of the points using the right turn test.
5. Call recursively on the remaining points (on each side of the vertical line).

1.5.1 Observation

On the left subproblem for any pair of points that has slope (corresponding to the line through these points) smaller/larger than l_m, we can eliminate one point on the basis of the "turn test".



Lemma 1 The running time of the algorithm T(n,h) = *O(n logh)*, where n is the number of input points and h is the output points.

1.6 MergeHull

Another algorithm based on divide-and-conquer paradigm which works by arbitrary partitioning is called merge hull. After creating arbitrary partition, we construct convex hulls of each partition. Finally, merge the two partitions in *O(n)* steps.

$$T(n) = 2T(\frac{n}{2}) + O(n)$$

The key step here is to merge the two upper hulls. These two upper hulls are separated by a vertical line L. The merge step computes the common tangent, called bridge over line L, of these two upper hulls.