Planar Convex Hulls

Input: $N=$ Set of n-points in the plane.

Output: Convex Hull.

1. How to Compute?

2. Naive Solutions.

3. Gift Wrapping Algorithms.

4. Two $O(n \log n)$ Algorithms.

5. Lower Bounds.

Naive Algorithms

– Generate All Possible Triangles

– Generate All Possible Edges.

Gift Wrapping Algorithm

– Start with the lowest point.

– Inductively, assume that part of the hull is computed.

– Consider the last ray that is added to the hull. For each remaining point, compute the angle which it makes with respect to this ray.

– The point with the minimum angle is on the hull.

– Stop when return to the starting point.

$$O(n^2)\text{-Time}$$

.

An optimal Incremental Algorithm

Upper and Lower Hulls/Chains

Computation of Upper Chain

Step 1: Compute the points belonging to the Upper Hull.

Step 2: Sort the points with respect to x-coordinate.

Step 3:
- Let N^i, $i \geq 1$, denote the first i points in this order.
- Let $U(N^i)$ denote the upper chain of N^i.
- Let us compute the upper chain incrementally.
- Assume we know $U(N^i)$, i.e. stored as the link list of vertices.
- Consider the addition of the (i+1)st point, $s = s_{i+1}$.

- Let r denote the point in $U(N^i)$ with the highest x-coordinate.
- Traverse $U(N^i)$ backwards, starting at r, till we reach a vertex $q \in U(N^i)$ such that whose left neighbor p lies below the line sq.

$U(N^{i+1}) = \{x_{min}, \cdots, p, q, s\}$.

Analysis of the Incremental Algorithm

Step 1: $O(n)$.

Step 2: $O(n \log n)$.

Step 3: $O(n)$!

At any step, the cost of each addition is $1 +$ the number of discards.

Total number of discards $< n$.

Theorem: The convex hull of n points in the plane can be computed in $O(n \log n)$ time. If the points are given in x-sorted order, then it can be computed in $O(n)$ time.

An Optimal Divide & Conquer Algorithm

Main Steps

1. Sort points w.r.t. x-coordinate.

2. Divide the point into two sets A and B, where A consists of the left $\lceil \frac{n}{2} \rceil$ points and B consists of the right $\lceil \frac{n}{2} \rceil$ points.

3. Recursively, compute $H_A = conv(A)$ and $H_B = conv(B)$.

4. Merge the two hulls into a common convex hull, H and output this hull.

Analysis of the Divide & Conquer Algorithm

1. Time for Sorting: $O(n \log n)$.

2. Recurrence for the running time

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

   where $f(n)$ is the time to merge the two hulls.

It turns out that merging will require linear time.

Thus, Divide and Conquer Algorithm runs in $O(n \log n)$ time.

The Merge Step

Polygons A and B are separated by a vertical line.

Merging is nothing but computing upper and lower tangents.

Procedure for computing Lower Tangent

1. Let a be the rightmost point of H_A.

2. Let b be the leftmost point of H_B.

3. While ab is not a lower tangent for H_A & H_B do
   (i) while ab is not a lower tangent to H_A do $a = a - 1$
   (ii) while ab is not a lower tangent to H_B do $b = b + 1$

4. Return ab.

Merge Step: Figure

Tricky, but each vertex is visited only once, hence linear time.

Lower Bounds : $\Omega(n \log n)$

Reduce the *Sorting Problem* to the *Convex Hull Problem*.

Sorting Problem: Sort the positive numbers $x_1, \cdot, x_n$.

Transformation: Compute the points (x_i, x_i^2), for $1 \leq i \leq n$.
These points could be viewed as the points on the parabola, thus all points will lie on the convex hull.

Compute Hull: Compute the convex hull by any desired method.

Extract sorted order:
- Determine the point with the lowest x-coordinate in linear time.
- Walk ccw around the hull.
- The x-coordinates of the points gives the sorted order of the original points.

Transformation and extraction is linear time, therefore convex hull must take $\Omega(n \log n)$*-time*

Ultimate Convex Hull Algorithm : $\Omega(n \log h)$

(Kirkpatrick and Seidel, SIAM Jl. Comp., 1986)

Let S denote the current set of points whose upper hull we wish to compute.

Procedure Connect(S)

Basis: If $|S|=1$, output this point and return.

Divide: Find a vertical line $x = a$ that divides splits S into two subsets of equal size.

Find the Bridge: Compute the convex hull edge that crosses the dividing line $x = a$. Let x_L, x_R be the x-coordinate of the left and right endpoints of this bridge, respectively.

Recurse: Divide the points of S in three groups.
-S_L lie to the left of x_L.
-S_R lie to the right of x_R.
-Points that lie underneath the bridge.
Recursively call **Connect(**S_L**)** and **Connect(**S_R**)**.

Theorem

The convex hull of a planar n- point set can be computed in $O(n \log h)$ time, where h is the number of vertices on the convex hull.

Arrangement of Lines

Input: N= Set of lines in the plane.

Output: $H(N)$= Arrangement, or Geometric Complex (i.e. 0-faces, 1-faces, 2-faces). A suitable representation of Planar Graphs, e.g. DCEL Data Structure.

An Incremental Algorithm

- Let N^i be the set of first i-added lines.

- Let $H(N^i)$ represent the arrangement of N^i.

- Consider the addition of (i+1)st line, $S = S_{i+1}$.

Main Steps in the incremental construction

1. Pick any line $L \in N^i$ and compute the intersection point, u, of L and S.

2. Search all edges of L in $H(N^i)$ to locate the edge containing u.

3. Travel along S (both in left and right direction) starting at u and splitting the successive faces R_0, R_1, ...

   i

What is Splitting a Face?

- We know edge containing $u \in R_0$.

- Access the node corresponding to the facial lattice of R_0.

- Adjacency information tells us all vertices and edges of R_0.

- Traverse them to find where S leaves R_0. This leads to splitting R_0 and creating two new faces.

- Let the point where S leaves R_0 be v. Now v plays the role of u.

Analysis

1. Computation of intersection of L and S and locating the edge containing u of $H(N^i)$ takes $O(i)$ time.

2. Cost of traversing and splitting along S is proportional to the total face-length of all faces of $H(N^i)$ intersecting S.

An important definition!

Zone: The set of faces of $H(N^i)$ intersecting S, together with the edges and vertices adjacent to them, is called the *zone* of S in $H(N^i)$.

ZONE THEOREM

Theorem: Let M be any set of m lines in the plane. For any line $S \notin M$, the size of the zone of S in $H(M)$ is $O(m)$ (actually $\leq 6m$).

Proof:

- W.l.o.g let S be horizontal.

- S intersect each line of M.

- First worry about only the left edges in the zone.

- Place a pebble on each left edge of each face of the zone.

- Pebbles are moved along the following rules.

Rule I: If left edge intersects S then pebble is moved to the intersection point. (Note :: *Every intersection point is assigned at most one pebble*)

Rule II: If left edge does not intersect S then let L' be the line carrying this edge and let L'' be the line whose intersection with L' forms the left vertex of this edge. Assign the pebble for this edge to the intersection point of L'' and S. (Note :: *At most two pebbles can be assigned to any intersection point*)

- Thus each of the m intersections can be assigned at most 3 pebbles.

- Since there are n-intersections, the total number of pebbles and hence the total number of left edges in the zone are at most $3m$.

Analysis Continued

Hence, the $i + 1$st step will require $O(i)$ time in all, therefore the arrangement can be computed in $O(\sum_{i=1}^{i=n} i) = O(n^2)$ time.

Theorem: Arrangement of n-lines in the plane can be build in $O(n^2)$ time.

Zone theorem in higher dimensions!