2D Randomized Incremental Convex Hull

Gary L. Miller

February 5, 1999

Today we will give an analysis one of the most natural algorithm for computing the convex hull of a set of points in 2D. The algorithm is randomized. We shall analyze the running time using a method called backwards analysis.
INPUT: A set of points $P = \{p_1, \ldots, p_n\} \subseteq \mathbb{R}^2$ Let P be a random permutation of the input points.

INIT:

1. Randomly extract 3(i.e. p_1, p_2, p_3) points from P and form a triangle T.
   (a) Let c be some fixed point interior to T.
2. "Draw" an Arc from c to each point in P
3. Partition P by which edge of T its arc crosses, if its crosses a point assign the point it either of the two edges.

{**Inductive Assumption:** We have a convex polygon and a set of points above each edges. We also mark a point as dead if it is in the interior of the polygon, otherwise, it is alive.}

4. **While** there exist a point above some edge **do**
   (a) Pick at random a point p(i.e., the next alive point from P) above, say, edge e.
   (b) $BuildTent(p, e, Poly)$

BuildTent(p:point,e:edge,P: polygon with points)

1. Going left and right from e find all edges visible to p.

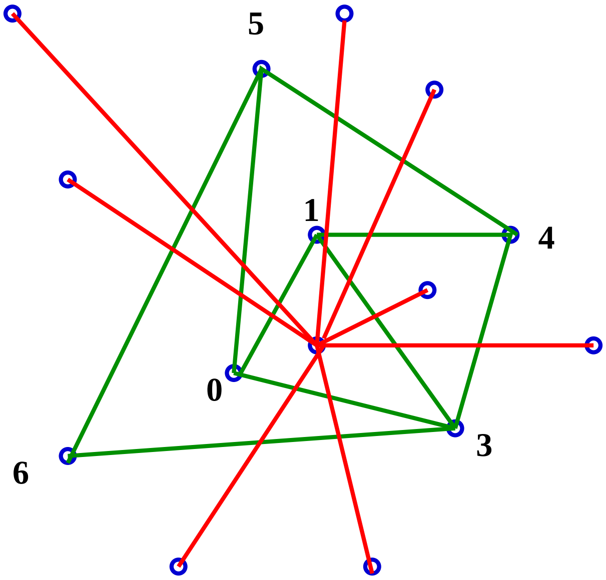

Figure 1: An Example of the Inremental Hull Algorithm after Iserting 6 Points.

2. Replace visible edges with a new pair of edges, a tent.

3. Move the point above the visible edges to the new edges.

Timing: The cost is dominated by the calls to BuildTent and the line-side tests in BuildTent.

We first bound the line-side tests performed by Steps 1) and 2): Danny's proof:

Charge each visible edge for the line-side test with p. Charge p for the two line-side tests which failed. Since the number of edges is at most twice the number of nodes we get a cost of $4n$.

We next bound the line-side test performed by Step 3): We will use backwards analysis to bound the expected number of line-side tests.

In the figure the points are added in order of their number, the green edges are the edges of the convex hull and the red edges are the arc. We claim all we need to do is count the expected number of red-green intersections.

Lets us keep all the edges ever constructed by BuildTent. These edge will form a planar graph, say, G. These are the green edge in the figure. We extend the ray to each point out to infinity. We claim that the number of line-side tests is bounded by the number of intersections of a ray and an edge in G. We run the incremental hull algorithm in reverse.

That is, we P being all the point and $CH(P)$ being the convex hull. In reverse order we delete the point p from P. If p in interior to $CH(P)$ we do not charge anything. If p in on the hull we delete p and the edge e_L to the left and e_R to the right of p from G and we charge of all the rays which intersect either e_L or e_R. If we assume that the size of P is k then the sum of the cost to delete any $p \in P$ will be $2(n-k)$. To see this, we note that every crossing between an arc and a boundary edge will be counted exactly twice, once for each point at either end of the edge.

Thus the expected cost to delete a point will be at most $2n/k$.

Thus the expected cost of step 3. is at most

$$\sum_{k=1}^{n} n/k = n(\gamma + n \ln n + o(1))$$