CS 266 Homework 8

Zachary DeStefano, PhD Student, 15247592

Due Date: June 5, 2014

Problem 11.2

We will always be going through each of the n points.
In the worst case, when we go through a point, almost every current facet except a constant number has to be deleted.
A new conflict graph is then made and for each face,
all the remaining points have to be checked to reconstruct the graph.

Deleting the current facets will take O(n) time.
Making a new conflict graph will be O(n) time for each facet and there are O(n) facets, so for each point there could be O(n^2) operations to reconstruct the conflict graph.
This has to be done for each point so in the worst case, the running time is $O(n^3)$.

An example:
Let's say we have a tower-like structure, where there is a triangular face at the bottom and then perpendicular to that face are n points that are almost collinear above it. If we pick the bottom face first and then pick the points in ascending order by distance from the bottom, then at each point, we have to completely delete and rebuild the conflict graph and the hull as all but the bottom facet would be changed. This is a case that needs $\Theta(n^3)$ time. It is illustrated below. In the figure below, if the bottom triangle and 1 makes your starting tetrahedron, then you process point 2 and then point 3, the running time will be as bad as possible.

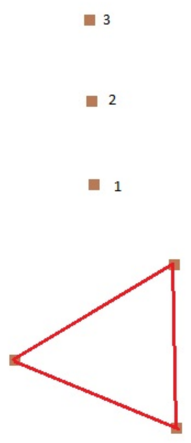

Figure 1: Bad Convex Hull points

Problem 11.4

In many applications, only a small percentage of the points in a given set P of n points are extreme. In such a case, the convex hull of P has less than n vertices. This can actually make our algorithm CONVEXHULL run faster than (nlogn).

Assume, for instance, that the expected number of extreme points in a random sample of P of size r is $O(r^{\alpha})$, for some constant $\alpha < 1$. (This is true when the set P has been created by picking points uniformly at random in a ball.) Prove that under this condition, the running time of the algorithm is O(n).

Problem 11.8

Describe a randomized incremental algorithm to compute the intersection of half-planes, and analyze its expected running time. Your algorithm should maintain the intersection of the current set of half-planes. To figure out where to insert a new half-plane, maintain a conflict graph between the vertices of the current intersection and the half-planes that are still to be inserted.