22C : 196 Computational Geometry
Homework 2

Some of the problems in this homework are adapted from the text *Computational Geometry: Algorithms and Applications* by de Berg et al., but I have stated such problems to avoid issues that may come up because of using different versions. Each of the following four problems is worth 2.5 points.

1. The first step in our $O(n \log n)$ algorithm for triangulating a simple polygon with n vertices was to use a sweep-line approach to add diagonals going up from each of the split vertices. These non-intersecting diagonals decompose the simple polygon into smaller polygonal regions. Assuming that the subdivision induced by the original simple polygon is given as a doubly conencted edge list (DCEL), and given the set of computed diagonals, describe how we can compute a DCEL for the resulting subdivision. The running time of your algorithm should be $O(n \log n)$. Suggestion: Review the sweep-line method for adding diagonals. Then think of some concrete algorithm for computing the DCEL. Analyze its running time, and if it is too high, ask yourself how it can avoid doing wasteful work.

   In the textbook description of the DCEL, a face remembers one half-edge from each component of its boundary. In our case, the boundary of each face had only one component, so a face needs to remember only one half-edge. In class, we worked with such a simplified DCEL.

2. Suppose that after this first step we have decomposed the original simple polygon (with n vertices) into k sub-polygons, and suppose that the i-th sub-polygon has $m_i \geq 3$ vertices. Show that $\sum_{i=1}^{k} m_i \log m_i = O(n \log n)$.

3. This question concerns our first slow algorithm for computing the convex hull of a set P of n points in the plane. Assume that the points are given as an array $P[1..n]$ where $P[i]$ hold the i-th point. Suppose the algorithm has computed a list consisting of (clockwise-oriented) convex hull edges but it has not figured out the order in which these edges occur on the convex hull. Each computed edge is of the from (i, j), which means that the directed segment from $P[i]$ to $P[j]$ is an edge of the convex hull as we traverse it clockwise. Describe an $O(n)$ algorithm for computing a correct ordering of these edges.

   In the figure, for example, we may be given the list

   $$\langle (5,7), (6,1), (1,5), (7,6) \rangle.$$

   A correct ordering of these edges is

   $$\langle (1,5), (5,7), (7,6), (6,1) \rangle.$$

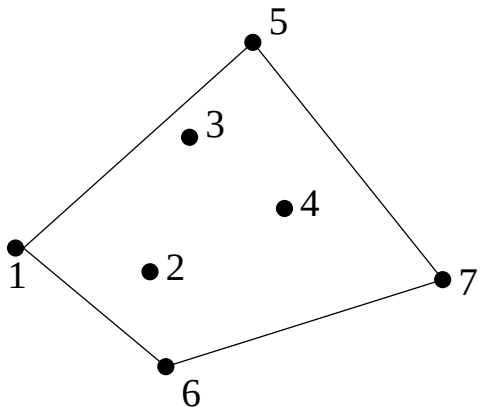

Figure 1: Problem 3

4. Show that the point $r = (r_x, r_y)$ lies to the left of the directed line from $p = (p_x, p_y)$ to $q = (q_x, q_y)$ if and only if the expression

$$(q_x - p_x)(r_y - p_y) - (r_x - p_x)(q_y - p_y)$$

is positive.

You are welcome to solve the following problems as well, but you will not receive any credit for doing this.

1. Suppose that we have a subroutine ConvexHull available for computing the convex hull of a set of points in the plane. Its output is a list of convex hull vertices, sorted in clockwise order. Now let $S = \{x_1, x_2, \ldots, x_n\}$ be a set of n integers. Show that S can be sorted in $O(n)$ time plus the time needed for one call to ConvexHull.

   This reduction shows that computing the convex hull takes $\Omega(n \log n)$ in those models of computation where sorting takes $\Omega(n \log n)$ time.

2. The *stabbing number* of a triangulated simple polygon is the maximum number of diagonals intersected by a line segment interior to the polygon. Give an algorithm that takes as input any n-vertex convex polygon and computes a triangulation that has stabbing number $O(\log n)$.

The homework is to be turned in into the dropbox *Homework2* on ICON. I would prefer if you type in the text, but hand-drawn figures are okay. The homework is due by 11:59 pm on Feb 21st.

On the question of collaboration and seeking help, I recommend thinking about each problem for 30 minutes first (not counting time spent getting familiar with basic material covered in class). You may collaborate with classmates after that, but definitely avoid looking at completely written solutions of others. Explain the final solution in your own words, and do not turn in a solution that you don't understand.