Computational Geometry
Homework 2

Several problems in this homework are adapted from the textbook *Computational Geometry: Algorithms and Applications* by de Berg et al., but I have restated the problems for convenience.

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 compute diagonals going up from each of the split vertices. These non-intersecting diagonals decompose the simple polygon into smaller polygonal regions. Assume that the subdivision induced by the original simple polygon is given as a doubly conencted edge list (DCEL). Given this DCEL and the set of computed diagonals, describe how we can compute a DCEL for the subdivision of the plane obtained by adding the diagonals. The running time of your algorithm should be $O(n \log n)$.

   Suggestion: To understand what I am asking in this question, you first have to read about DCELs from Chapter 2 and see how they can represent planar subdivisions. Then review at a high level the sweep-line method for computing the diagonals from split vertices in the triangulation algorithm. After that, think of some concrete algorithm for computing the new DCEL for the subdivision obtained due to adding diagonals. 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.

2. Suppose that after this first step (mentioned above), 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. The *stabbing number* of a triangulated simple polygon is the maximum number of diagonals intersected by any line. Give an algorithm that takes as input any n-vertex convex polygon and computes a triangulation that has stabbing number $O(\log n)$. (This one is a bit tricky till you see the idea; if you are stuck after spending substantial time, ask for help.)

4. Give an efficient algorithm to determine whether a polygon $\mathcal{P}$ with n vertices is monotone with respect to some line, not necessarily a horizontal or vertical one.

5. Here is a paranoid algorithm to compute the maximum of a set A of n real numbers:

Algorithm 1 PARANOIDMAXIMUM(A)

1: **if** $|A| = 1$ **then**
2: Return the unique element $x \in A$
3: Pick a random element x from A
4: $x' \leftarrow$ PARANOIDMAXIMUM($A \setminus \{x\}$)
5: **if** $x \leq x'$ **then**
6: Return x'
7: **else**
8: Now we suspect that x is the maximum, but to be absolutely sure, we compare x with the $|A| - 1$ other elements of A.
9: Return x

What is the worst case running time of this algorithm? What is the expected running time? Explain. (Solve this problem after we complete material from Chapter 4.)

The homework is due via ICON by 11:59 pm on Tuesday, February 25. I would prefer if you type-set your submissions, but feel free to include hand-drawn figures.

Please pay attention to the policy on collaboration outlined in the syllabus. In particular, you will get the most out of home work if either you work on your own or collaborate with a fellow student or two in a reasonable way. I will evaluate the originality of your writing while reviewing your work. Feel free to stop by during my walk-in hours if you need help.