6.838: Geometric Computing

Spring 2005
Problem Set 2
Due: Monday, March 14*

MANDATORY PART

Problem 1. Intersection check

Given a set S of n vertical segments $s_1 \dots s_n$ in the plane, construct a data structure that uses $O(n \log n)$ space, that given a *horizontal* segment s, checks if s intersects any of segments in S in $O(\log^{O(1)} n)$ time.

Problem 2. Convex point location

Consider a convex polygon P with n vertices. Assume that the vertices of P are given in an array in a sorted order along the boundary.

Show that, given a query point q, one can detect if $q \in P$ in time $O(\log n)$.

Problem 3. Arrangement of circles

Consider the arrangement of n circles $C_1 \dots C_n$ in the plane. What is the maximum number of faces in this arrangement ?

(Recall that a face is a maximal planar region delimited by the circle boundaries.)

Problem 4. Textbook, exercise 9.11, p. 209

A Euclidean Minimum Spanning Tree (EMST) of a set P of n points in the plane is a tree over the vertex set P of a minimum total edge length. Show that the set of edges of a Delaunay triangulation of P contains EMST of P. Does this imply an $O(n \log n)$-time algorithm for computing EMST ?

OPTIONAL THEORETICAL PART

Problem A. Exercise 8.4, p. 181

Let L be a set of n lines in the plane. Give a $O(n \log n)$-time algorithm to compute an axis parallel rectangle that contains all the vertices of the arrangement of L.

*To Chuck Wright, Piotr's administrative assistant, next to G 642.

Problem B. Exercise 8.15, p. 182

Let S be a set of n segments in the plane. A line l that intersects all segments in S is called a *stabber* for S.

Give an $O(n^2)$-time algorithm to decide if there exists a stabber for S.

Optional Programming Part

Implement a Java applet that *simulates visually* the algorithm of Lecture 6 for creating the data structure of point location (note that you are not required to actually implement the algorithm).

Your applet should first enable the users to provide a planar subdivision, e.g., by specifying the segments. Then your algorithm should enumerate all segments, and for each segment, update the trapezoid decomposition on the screen.

Feel free to add any additional bells and whistles.