6.838: Geometric Computing

Fall 2003

Problem Set 1

Mandatory Part

Problem 1. Application of convex hulls: diameter

Let P be a set of points on the plane. The diameter of P is defined as $\max_{p,q\in P} \|p-q\|_2$, where $\|\cdot\|$ is the Euclidean norm.

Design an $O(n\log n)$-time algorithm for computing the diameter of P.

Hint: Compute the convex hull of P first.

Problem 2. Vertical segment intersection.

In the class (Lecture 2) we have seen an algorithm for reporting all segment intersections when all segments are either horizontal or vertical. The description given in the class focused on detecting intersections between horizontal and vertical segments, leaving open the problem of detecting the intersections between vertical and vertical (or horizontal and horizontal) segments. To fill this gap, construct an efficient algorithm, which given a set of vertical (only) segments, reports all pairs of segments having nonempty intersection.

For full credit, your algorithm must run in time $O(P+n\log n)$. Recall that n is the number of segments and P is the number of intersections.

Problem 3. Textbook, exercise 3.3, p. 60.

A *rectilinear polygon* is a simple polygon of which all edges are either horizontal or vertical. Give an example to show that $\lfloor n/4 \rfloor$ cameras are sometimes necessary to guard a rectilinear polygon with n vertices.

Problem 4. Textbook, exercise 4.15, p. 93.

A simple polygon P is called *star-shaped* if it contains a point q such that for any $q \in P$ the line segment $p-q$ is contained in P.

Give an algorithm to decide whether a given simple polygon P is star-shaped. Your algorithm can be randomized; the expected running time of your algorithm should be $O(n)$. You can assume that P is given as a sequence of consecutive vertices $v_1, \ldots v_n$, such that the edges $v_1 - v_2, v_2 - v_3, \ldots v_n - v_1$ form the boundary of P.

Optional Theoretical Part

Problem A. Textbook, exercise 2.12, p. 43.

Let S be a set of n triangles in the plane. The boundaries of the triangles are disjoint, but it is possible that a triangle lies entirely inside another triangle. Let P be a set of n points in the plane. Give an $O(n \log n)$-time algorithm that reports each point in P lying outside all triangles.

If you want, you can assume that all x-coordinates of points in P as well as of vertices of triangles in T are distinct.

Problem B. Area of the union.

Let S be a sequence of n two-dimensional orthogonal rectangles (i.e., whose sides are parallel to either x or y axis).

Give an $O(n \log n)$-time algorithm that computes the area of the union of the rectangles in S. Note: the rectangles can intersect. As before, you can assume that all x-coordinates of the vertical sides of the rectangles are distinct.

Optional Programming Part

Implement a Java applet that illustrates the $O(n)$-expected time algorithm for linear programming in 2D. Your applet should:

- Provide a way for the user to give the input, i.e., the n half-planes. It is strongly preferable that the input could be specified graphically (e.g., the user could define a new half-plane by clicking at two points).
- Illustrate the behavior of the algorithm. In particular, it should emphasize the new constraint currently added, indicate if the old solution is feasible or not, and if not, demonstrate how a new solution is found.

Of course, you are welcome to add as many bells and whistles as you'd like.
A few suggestions:

- You can assume that the boundary of the screen (or the input box) defines additional constraints on the solution. In this way, you can ensure that the linear program is always bounded.
- You can assume that the optimization direction is "down", so that it does not have to be specified by the user.