Assignment 4 - Computational Geometry (0368-4211)

Due: May 31, 2011

Problem 1

(a) Let $e_1, \ldots, e_n$ be n vertical segments in three dimensions. Find, in linear time, a plane that intersects all the segments, or determine that there is no such plane. (**Hint:** Express this property by linear inequalities in the coefficients of the plane.)

(b) Describe in dual space the set K of all points dual to planes that satisfy the property in (a). What is the shape of K? What is its complexity? And how fast can one compute K?

Problem 2

Let P be a given set of n points in d dimensions, and let C denote the convex hull of P. Design an $O(n^2)$-time algorithm that determines all the vertices of C. (**Hint:** Solve n linear-programming problems, one for each point in P, where the variables are the coefficients of an appropriate hyperplane.) (**Note:** This is a weaker problem than the one of computing the complete structure of C, which, in higher dimensions, can have a much larger complexity.)

Problem 3

Let T be a set of n triangles (in general position, but possibly intersecting) in the plane. Using duality, express the property that a line ℓ intersects a triangle $\Delta = abc$ in T. Use this representation to find a line that intersects the largest number of triangles. (**Hint:** After the duality, use line sweeping to find the face in the map formed by the (intersecting) dual regions, that is contained in the largest number of regions.) What is the running time of the algorithm?

Problem 4

Given n inequalities $a_i x + b_i y \geq 1$, for $i = 1, \ldots, n$, find a point (x, y) that satisfies all these inequalities and is nearest to the origin (if such a point exists). Adapt

Megiddo's algorithm or the randomized incremental algorithm to obtain a linear-time (or expected linear-time) solution.

Problem 5

Let $D_1, \ldots, D_n$ be n disks in the plane, and let K denote their intersection.

(a) Show that K has linear complexity. (**Hint:** Analyze how many arcs can the largest disk contribute to the boundary of K, and use induction.)

(b) Give an $O(n \log n)$ algorithm for computing K. (Use divide-and-conquer, for example.)

(c) Give a linear-time algorithm for finding the lowest point in K (without computing K explicitly), or determining that K is empty. (There are several ways of doing this. One is to extend Megiddo's algorithm.)