Due Date: October 6, 2005

Problem 1: (i) Let $P = \langle p_0, \ldots, p_{n-1} \rangle$ and $Q = \langle q_0, \ldots, q_{n-1} \rangle$ be two nonintersecting convex polygons in $\mathbb{R}^2$. Show that the common tangent, both inner and outer, can be computed in $O(\log n)$ time. You can assume that the sequence of vertices of P (and Q) is stored in an array.

(ii) Let P and Q be two (possibly intersecting) convex polygons with n vertices each. Describe an $O(\log n)$ time algorithm for computing the minimum distance between P and Q. If P and Q intersect, then the minimum distance between them is zero.

Problem 2: (i) Given a collection $\mathcal{R}$ of "red" nonintersecting line segments and another collection $\mathcal{B}$ of "blue" nonintersecting segments in $\mathbb{R}^2$, show that all red-blue intersections (intersections between a red segment and a blue segment) can be counted in time $O(n \log^2 n)$, where $n = |\mathcal{R}| + |\mathcal{B}|$. What is the space complexity of your algorithm? Improve the space complexity to $O(n)$.
(**Hint:** Use a segment tree on x-projections of segments and, at each node v, count intersections among the segments stored at v. You need to store some additional information at each node of the segment tree. Make sure that each intersection is counted exactly once.)

Extra credit: Improve the time complexity to $O(n \log n)$.

Problem 3: Let S be a set of n points in the plane. A point $p \in S$ is called *maximal* if there is no point $q \neq p \in S$ such that $x(p) \leq x(q)$ and $y(p) \leq y(q)$. Describe an $O(n \log h)$ time algorithm to compute the maximal points of S, where h is the output size.

Extra credit: Describe an $O(n \log h)$ time algorithm to compute the maxima of a set of n points in $\mathbb{R}^3$; h is again the output size.

Problem 4: Let P be a set of n points in $\mathbb{R}^3$. We define a map $N : \mathbb{S}^2 \to P$, where $N(u) = \arg\max_{p \in P} \langle p, u \rangle$. N induces a subdivision P^* of $\mathbb{S}^2$. What are the vertices, edges, and faces of P^*, and how fast can P^* be computed?

We call a pair of vertices $p, q \in P$ *antipodal* if there are two parallel planes h_p, h_q passing through p and q, respectively, so that P lies between them. Describe an $O((n + k) \log n)$-time algorithm to compute the set of antipodal pairs, where k is the number of such pairs.