CS 274
Computational Geometry (Autumn 2006)
Homework 4

Homework 4 is due **at the start of class (1:10 pm) on Wednesday, November 15, 2006**.

[1] Line-polygon intersection and linear programming with preprocessing (6 points).

(i) You are given a convex polygon P, described as an array of n vertices in counterclockwise order. Suppose you know the indices of the lexicographically least and greatest vertices. Give an algorithm for testing in $\mathcal{O}(\log n)$ time whether an arbitrary line ℓ intersects P, with *no preprocessing*. (Hint: duality might help you better understand the problem.)

(ii) You are given a two-variable linear program. The feasible region is two-dimensional (i.e. a polygon), possibly unbounded. Explain how to preprocess the linear constraints so that, given any objective vector c, you can solve the linear program in $\mathcal{O}(\log n)$ time (not counting the preprocessing time).

(iii) Briefly explain the relationship between problems (i) and (ii), and in particular, between ℓ and c.

[2] Cocircularities and symbolic weight perturbations (8 points). If you believe that every point set has a convex hull, then the parabolic lifting map shows that every d-dimensional point set has a Delaunay triangulation ... unless the point set includes $d + 2$ points lying on a common hypersphere, in which case the underside of the convex hull of the lifted points might not be simplicial. Let's fix this with symbolic perturbations. We'll work in two dimensions, but the ideas generalize easily to any dimensionality.

Let $V = \{v_1, v_2, \ldots, v_n\}$ be a set of vertices in the plane. V may have subsets of four or more cocircular vertices, so V may have many Delaunay triangulations.

(i) For any vertex v with a real *weight* w, define the lifting map $\langle v, w\rangle \mapsto v^+ = (v_x, v_y, v_x^2 + v_y^2 - w) \in E^3$. Show that there exist weights $w_1, w_2, \ldots, w_n$ (one for each vertex in V) such that
- if $\textsc{InCircle}(v_i, v_j, v_k, v_l) > 0$, then $\textsc{Orient3D}(v_i^+, v_j^+, v_k^+, v_l^+) > 0$;
- if $\textsc{InCircle}(v_i, v_j, v_k, v_l) < 0$, then $\textsc{Orient3D}(v_i^+, v_j^+, v_k^+, v_l^+) < 0$; and
- if $\textsc{InCircle}(v_i, v_j, v_k, v_l) = 0$, and the four vertices are distinct and *not* collinear, then $\textsc{Orient3D}(v_i^+, v_j^+, v_k^+, v_l^+) \neq 0$. Therefore, the non-vertical faces (those not parallel to the z-axis) of the convex hull of $V^+ = \{v_1^+, v_2^+, \ldots, v_n^+\}$ are all triangles.

Hints: choose small weights. Think about the relationship between a lifted point and a plane through a triple of lifted points when you perturb the weight of one of those four points. Then think about all the points and planes together when you perturb a weight. A correct proof will probably use induction.

(ii) Let T be the projection of the faces on the underside of $\text{conv}(V^+)$ down to E^2. T is called a *weighted Delaunay triangulation*. Prove that for your choice of weights, T is a Delaunay triangulation of V.

(iii) We can modify the randomized incremental Delaunay triangulation algorithm by replacing each $\textsc{InCircle}$ test with an $\textsc{Orient3D}$ test on the lifted points, and thus guarantee that we always compute the same canonical triangulation regardless of the point insertion order. (For this algorithm, the backward analysis of running time extends to points not in general position, because every Delaunay triangulation is now unique.) However, we want to use symbolic weights (like in our point-in-polyhedron algorithm) instead of explicitly computed weights—explicit weights satisfying the conditions in part (i) are so small that numerical roundoff makes computation with them impractical. So we begin by computing $\textsc{Orient3D}$ on the *un*perturbed points (i.e. with the weights set to zero). Part (i) tell us we only need to take the weights into account if that result is zero, in which case we must perform one or more additional tests to disambiguate the sign of $\textsc{Orient3D}$ on the perturbed points. Describe those additional tests. Hint: observe that $\textsc{Orient3D}$ is a *linear* function of the weights, so it's easy to determine how changing a weight will change the function. (Even if $\textsc{Orient3D}$ were a black box).

Postscript: the Dutch Book's trick for handling the vertices of the triangular bounding box works by assigning them weights of $-C$, $-C^2$, and $-C^3$ as $C \to \infty$, and modifying the INCIRCLE test accordingly.

[3] Counting problems in orthogonal range search (5 points). The orthogonal range search problem we have studied is the *reporting* problem, where we want to list all the points in a query box. Here we are interested in the *counting* problem, where we merely want to state how many points are in a query box. Call this number k. Since we are not listing the points, there is no $\Omega(k)$ lower bound on query time.

Suppose you have a two-dimensional layered range tree (with fractional cascading) representing a set P of n points in the plane. Without modifying the data structure, explain how you can report the number of points of P in a query box $B = [x, x'] \times [y, y']$ in $\mathcal{O}(\log n)$ time.

(If you can't solve this, then for partial credit explain how to report the number of points in $B = [y, y']$ from a 1-dimensional range tree in $\mathcal{O}(\log n)$ time.)

[4] Adding points to a convex hull (6 points). Let S and T be two sets of points, having s points and t points respectively. Suppose we are given the convex hull $\operatorname{conv}(S)$ expressed as a facet graph, as produced by the Clarkson–Shor incremental convex hull construction algorithm, and our task is to incrementally insert the points in T and thus construct $\operatorname{conv}(S \cup T)$.

We use the following algorithm. First, for each point p in T, find a facet of $\operatorname{conv}(S)$ visible from p (or verify that none exists) by brute force (checking each point against each facet), and thereby build a conflict graph in $\mathcal{O}(ts^{\lfloor d/2 \rfloor})$ time. Second, incrementally insert the vertices of T in random order (with each permutation being equally likely) using the Clarkson–Shor algorithm.

If we were constructing $\operatorname{conv}(S \cup T)$ from scratch, with the insertion order wholly randomized, it would take at worst expected $\mathcal{O}((s+t)^{\lfloor d/2 \rfloor} + (s+t)\log(s+t)) = \mathcal{O}(s^{\lfloor d/2 \rfloor} + t^{\lfloor d/2 \rfloor} + s \log s + t \log t)$ time. However, because S is *not* a random subset of $S \cup T$, the expected running time for this algorithm can be worse. An adversary could choose S in such a way that inserting the vertices of T is slow.

Prove that the expected time to incrementally insert the vertices of T is in $\mathcal{O}(t^{\lfloor d/2 \rfloor} + t \log t + ts^{\lfloor d/2 \rfloor})$. (For asymptotic purposes, treat the dimension d as a small fixed constant.)

Some hints.

- You may safely assume that the time spent making structural changes does not exceed the time spent updating the visibilities (conflict graph), and analyze only the latter.
- The technique presented in class for analyzing the expected running time works here as well, with several differences. First, define a (j, m)-facet to be a facet with j stoppers in T, m triggers in T, and $d - m$ triggers in S. (Facets with stoppers in S will never appear, so we can ignore them.) Define a $(\leq j, m)$-facet likewise, having $\leq j$ stoppers. Adapt the random sampling technique used in class to show that the number of $(\leq j, m)$-facets is
$$F_{j,m} \in \mathcal{O}\left(j^m s^{\lfloor d/2 \rfloor} + j^{m - \lfloor d/2 \rfloor} t^{\lfloor d/2 \rfloor}\right)$$
for $j \geq 1$. (Obviously, $F_{0,m} \in \mathcal{O}((s+t)^{\lfloor d/2 \rfloor})$.) Figure out the probability that a $(\leq j, m)$-facet will appear at some time during the algorithm. Find the sum of all visibilities created.
- Alternatively, you could adapt the proof in Section 9 of Seidel's paper. The changes to the analysis arise because the sets S and T might be chosen (by an adversary) so the uninserted vertices of T can see a larger-than-usual number of facets, and the inserted vertices of T have much higher average degree than the vertices of S. This is why the time complexity can exceed $\mathcal{O}((s+t)^{\lfloor d/2 \rfloor})$.
- $\sum_{i=1}^{n} \frac{1}{i} \leq 1 + \ln n$.
- $\sum_{i=1}^{\infty} \frac{1}{i^2} = \frac{\pi^2}{6}$. Likewise, $\sum_{i=1}^{\infty} \frac{1}{i^p}$ converges for any $p \geq 2$.