CS 274
Computational Geometry (Spring 2005)
Homework 2

**Homework rules:** All the homework rules from Homework 1 remain in effect. Please reread the first page of Homework 1 if you don't remember the policies. All homeworks in this class are individual assignments and should be treated like take-home exams.

Homework 2 is **due at the start of class (1:10 pm) on Wednesday, March 2, 2005**.

**[1] Few turn vertices** (4 points). Suppose a polygon has $\mathcal{O}(1)$ turn vertices. Show how to modify the polygon triangulation algorithm (Chapter 3 of the Dutch Book) to run in $\mathcal{O}(n)$ time on this polygon, even in the decision-tree model of computation (e.g. without radix sort).

**[2] Security camera** (9 points). Let $X$ be a PSLG with $n$ segments, which represent the walls of a gallery. Some of the walls separate a bounded "inside" region from an unbounded "outside" region where there are no walls. Inside the gallery, there may be walls (non-crossing segments) anywhere, some of which are just there to hang paintings on. (More precisely, there is a simple polygon $P$, not generally convex, such that every edge of $P$ is a segment in $X$, and for every segment $s \in X$, $s \subset P$.)

A fixed, rotating security camera is attached to the ceiling at a point $q$ in the gallery (i.e. $q \in P$). The visibility between two points $q$ and $p$ is *occluded* if there is a segment $s \in X$ such that $s$ intersects the line segment $qp$ and the points $p$ and $q$ lie on opposite sides of $s$. (If $q$ or $p$ is collinear with $s$, then $s$ does not occlude the visibility between them. If the camera is placed on a wall, it can see on both sides of the wall.) A point $p$ is *visible* from $q$ if no segment in $X$ occludes the visibility between them.

- (i) Give (in English) an efficient algorithm to compute the subset of $P$ visible from $q$, and analyze its running time. Your algorithm's output should be a list of segments in counterclockwise order, bounding the visible region. You do not need to prove your algorithm's correctness (though it does need to be correct). Algorithms slower than $\mathcal{O}(n \log n)$ time will only receive partial credit. Hint: there are ways to sweep the plane that don't involve moving a line linearly.
- (ii) The gallery is displaying eight masterpieces by the renowned artist Florent de Microscopie, famed for his paintings the size of points. Each of the eight paintings is hung at a different point inside the gallery. Give an efficient algorithm to determine whether there is a single point $q \in P$ from which a fixed rotating security camera can monitor all eight paintings, and to output one such point if at least one exists. Analyze its worst-case running time in term of $n$. (Don't try to give an output-sensitive bound; the output size is one.) Unduly slow algorithms (or fast algorithms with slow analyses) will only receive partial credit, but you should not expect to achieve $\mathcal{O}(n \log n)$ time. (Note: even if you didn't solve part (i), you can still answer this question by assuming you know a solution to part (i).)

**[3] Cocircularities I** (6 points). Let $V$ be a vertex set in the plane. $V$ may have subsets of four or more cocircular vertices, so $V$ may have many Delaunay triangulations.

- (i) Prove that it is possible to transform a triangulation of a convex polygon to any other triangulation of the same polygon by a sequence of flips.
- (ii) Prove that it is possible to flip from any Delaunay triangulation of $V$ to any other Delaunay triangulation of $V$, such that every intermediate triangulation is also Delaunay.
- (iii) Prove that *every* Delaunay triangulation of $V$ maximizes its minimum angle—there is no triangulation of $V$ whose smallest angle is greater.

**[4] Cocircularities II** (7 points). The randomized incremental insertion algorithm for constructing a Delaunay triangulation with conflict-based point location runs in expected $\mathcal{O}(n \log n)$ time even if the input points do not have a unique Delaunay triangulation. Here we consider one way to prove it. Assume we are using a triangular bounding box.

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

(i) Show that there exists a vertex set $W = \{w_1, w_2, \ldots, w_n\}$ such that, for each choice of four distinct indices $i$, $j$, $k$, and $l$ in the range $[1, n]$,
- if $\text{INCIRCLE}(v_i, v_j, v_k, v_l) > 0$, then $\text{INCIRCLE}(w_i, w_j, w_k, w_l) > 0$;
- if $\text{INCIRCLE}(v_i, v_j, v_k, v_l) < 0$, then $\text{INCIRCLE}(w_i, w_j, w_k, w_l) < 0$; and
- if $\text{INCIRCLE}(v_i, v_j, v_k, v_l) = 0$, then $\text{INCIRCLE}(w_i, w_j, w_k, w_l) \neq 0$ so DT$(W)$ is unique.

Hint: jiggle the vertices of $v$ cleverly. Note that a correct proof will probably use induction.

(ii) Does moving the vertices of DT$(W)$ to the coordinates $V$ yield a Delaunay triangulation of $V$? Why?

(iii) Prove that the number of flips performed by the random incremental insertion algorithm with input $V$ cannot exceed the number of flips performed by the random incremental insertion algorithm with input $W$ (which we know is in $\mathcal{O}(n)$ on average). *Warning:* The actual flips performed in the former case is *not* necessarily a subset of the flips performed in the latter case; you'll need a slightly more subtle argument, but only slightly. Assume the bounding box vertices are far enough away to avoid causing trouble.

This doesn't quite complete the proof; we also need to show that the point location time is no greater for $V$ than for $W$. No need to belabor this homework, though.

**[5] Constrained Delaunay triangulations** (4 points). Sometimes you want to construct a structure like a Delaunay triangulation, but you need to enforce the presence of certain edges in the triangulation—for instance, the boundaries of an object or domain you're triangulating.

Let $X$ be a PSLG. Define the visibility between two points as in problem [2]. A triangle $t$ is *constrained Delaunay* if its vertices are vertices in $X$, the interior of $t$ intersects no segment in $X$, and the circumcircle of $t$ encloses no vertex of $X$ that is visible from any point in the interior of $t$.

Let $t$ be a triangle that satisfies the first two of those three conditions. Let $q$ be a point in the interior of $t$. Prove that if no vertex of $X$ inside $t$'s circumcircle is visible from $q$, then no vertex of $X$ inside $t$'s circumcircle is visible from *any* point in the interior of $t$ (so $t$ is constrained Delaunay).

**[open1] Open problem: Easy linear-time triangulation** (two letter grades; e.g. from an A− to an A+). Several algorithms have been developed for triangulating a polygon in linear time, even restricted to the decision-tree model of computation. However, they're complicated and impractical. If we throw away the decision-tree restriction and allow the use of radix sort and bucketing, is there a linear-time polygon triangulation algorithm that is elegant and relatively simple?

**[open2] Open problem: Shallow Delaunay history DAG** (three letter grades). Devise an algorithm (with polynomial running time in the number of vertices) that chooses the order in which vertices are inserted into a Delaunay triangulation so that the standard Delaunay history DAG has an expected depth of $\mathcal{O}(\log n)$. The algorithm must work for every finite vertex set. (Partial credit is offered for a proof that such an order always exists, or an algorithm that achieves $o(n)$ expected depth. Ask me if you want an example of a vertex set that creates a history DAG with $\Theta(n)$ expected depth with a randomized ordering.)

**Note:** The deadline does not apply to the open problems, nor do the rules against collaboration. The awards are likely to be accompanied by a publication with your name on it. The awards do not apply if I'm wrong about them being open, or somebody else solves them first.