CS 274
Computational Geometry (Spring 2003)
Homework 1

Homework rules: All homeworks in this class are individual assignments. You may not seek or obtain assistance from anyone but me. You may not help your fellow students, either. Giving or receiving hints or answers is grounds for an immediate F in the course. However, you may consult any books, papers, Web pages, or other inanimate information sources you please. If you use information from such a source, please cite it. I do not believe that the solutions to any of the problems here are available in easy-to-find references, but if you do find one, it's fair game as long as you cite it. (After all, a good literature search should be rewarded.) You may not, however, let your classmates know about your discovery or sources until the deadline has passed.

You are welcome to ask me to clarify anything in the readings, the lectures, or this homework that you don't fully understand. But asking for extra hints is permitted only in class, where everyone can hear the answer.

I prefer typeset solutions, handed in on paper. Handwritten solutions will be accepted with the proviso that I may give a solution an instant zero if there is any word or variable I can't read clearly. Sorry, but the volume of homework is such that I can't spend time trying to backward-engineer whether it's an i or a j. Any such decision is final and irrevocable. By contrast, printed solutions reserve the right to argue with my grading after the fact.

You may give me your solutions directly, or put them in my CS mailbox or under my office door (625 Soda) before the deadline. Emailed solutions will be accepted grudgingly, but please use email only if you're out of town or it's otherwise really inconvenient to come to Soda. Snail-mailed solutions (sent to my office; see my Web page for the address) will be accepted if they are postmarked the day before the deadline. Solutions postmarked on the day of the deadline will be accepted if the postmark is clearly from outside the Bay Area. Late homeworks are not accepted.

Homework 1 is due **at the start of class (3:30 pm) on Thursday, February 6, 2003**.

[1] Convexity (7 points). Let $P = \{p_1, p_2, \ldots, p_n\}$ be a finite set of points in d-dimensional Euclidean space. Notwithstanding the abominable foolishness I sputtered in class, a point q is an *affine combination* of points in P if there is a choice of weights w_i such that

$$q = \sum_{i=1}^{n} w_i p_i, \qquad \text{where } \sum_{i=1}^{n} w_i = 1.$$

The point q is a *convex combination* of points in P if there is a **nonnegative** set of weights for which these equations hold.

(i) "q is an affine combination of points in P if and only if q' is a linear combination of vectors in V." Give a vector q' and a set V of vectors for which this statement is true.

(ii) Prove that if q is an affine combination of points in P, there is a subset $P' \subseteq P$ for which P' contains at most $d + 1$ points and q is an affine combination of points in P'. (If you get stuck, go brush up on *matrix rank* in your undergraduate linear algebra textbook.)

(iii) Prove that if q is a convex combination of points in P, there is a subset $P' \subseteq P$ for which P' contains at most $d + 1$ points and q is a convex combination of points in P'. (Hint: if there are more than $d + 1$ points for which the weight w_i is not zero, you can adjust the weights a bit so that one of them is reduced to zero and the others are still nonnegative but the equalities still hold. If you can't figure out how to prove that, look up "Radon's Theorem" for a clue.)

[2] Binary Search on Convex Hulls (5 points). Let H be a convex polygon whose h vertices $P = \langle p_1, p_2, \ldots, p_h \rangle$ are listed in counterclockwise order around H. Let q be a point outside H. To complete Chan's Shattering algorithm, write pseudocode to find, in $\mathcal{O}(\log h)$ time, the index i of the vertex $p_i \in P$ such that the line qp_i is tangent to H and H is on the left side of the ray directed from q to p_i. If there is more than one such vertex, choose the vertex nearest q.

For example, here is pseudocode to do the job in $\mathcal{O}(h)$ time.

```
SEARCH(q, h, P)
1    i ⇐ 1
2    for j ⇐ 2 to h
3        α ⇐ ORIENT2D(q, p_i, p_j)
4        if α < 0 or (α = 0 and |p_j − q| < |p_i − q|)
5            i ⇐ j
6    return i
```

[3] Overlapping Segments (5 points). Describe (in English, not pseudocode) how to modify the segment intersection algorithm taught in class so that it correctly handles the case where two or more parallel segments overlap (i.e. intersect at an infinite number of points). The modified algorithm should report each pair of parallel overlapping segments as such, as well as reporting the ordinary intersection points. Any number of segments may lie in a single line. Your modified algorithm should still run in $\mathcal{O}((n + I) \log n)$ time.

Does your algorithm require any modifications to the event queue or status tree? Be sure to note *every* situation in which your algorithm varies from the one given in class (and in the Dutch book).

[4] Face topology reconstruction (2 points). Suppose we are restricted to a decision-tree model of computation. Practically, this means we cannot use radix sort, bucketing, or other techniques that make k-way decisions for large k; we are only permitted to make binary decisions (i.e. **if** statements). It's well known, for instance, that no comparison-based sort can have a better worst-case running time than $\Theta(n \log n)$, even though radix sort (which is not comparison-based) runs in $\mathcal{O}(n)$ time.
(See http://www.cs.berkeley.edu/∼jrs/61b/lec/35 if you're not familiar with this kind of lower bound result.)

Suppose we are trying to compute the overlay of two subdivisions, and we have completed the sweepline intersection step. We now have a DCEL with the faces missing, and we want to determine the face topologies next (i.e. match outer boundaries of faces with inner boundaries of their holes). However, we have foolishly not retained any information from the sweepline stage. Explain why the worst-case running time of any algorithm that determines the face topologies is in $\Omega(h \log h)$ (in the decision-tree model), where h is the complexity of the DCEL.

[5] Maximality (6 points). Let P be a set of n points in the plane. A point $p \in P$ is *maximal* if no other point in P is both above and to the right of p.

(i) Give a four-point example P that has both a maximal point not on the convex hull, and a convex hull point that is not maximal.

(ii) For any P, give an $\mathcal{O}(n)$-time algorithm to compute the maximal points of P. Be careful to handle degeneracies correctly (i.e. points with the same x-coordinates or the same y-coordinates).

(iii) Suppose we are restricted to a decision-tree model of computation. Give an $\mathcal{O}(n \log h)$-time algorithm to compute the h maximal points of P. (Adapt Chan's Shattering algorithm.)