COS 451 – Assignment (due March 11, 2003)

1 The Art of the Nitty-Gritty

Implement a primitive CCW(A, B, C), which given three points A, B, C, computes the following information: If $A \neq B$, then the primitive returns 1 (resp. -1), if A, B, C make a right (resp. left) turn, 0 if C lies in the closure of AB, 2 if B lies in the relative interior of AC, -2 if A lies in the relative interior of BC. If $A = B$ then the primitive returns 0 if $A = B = C$ and 2 if $A \neq C$. Use CCW to implement a routine to check whether two (closed) segments intersect.

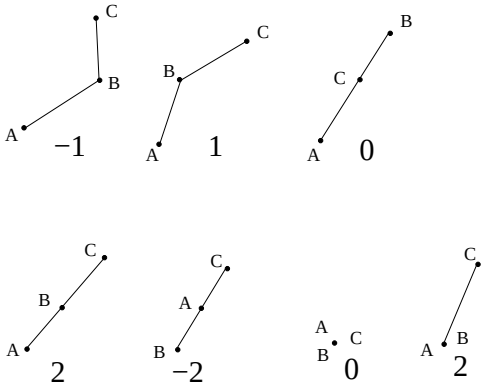

Figure 1: *CCW in its full glory*

2 Another Convex Hull Algorithm

Here is a great new convex hull algorithm to add to our catalog. So fast it is, indeed, that we shall call it *quickhull*. The idea is to identify the leftmost and rightmost points and join them with a line segment ab. Among the remaining points, let c and d be the furthest away from the line passing through ab, with c (resp. d) above (resp. below) the line. Next, we eliminate the points strictly inside the quadrilateral $abcd$, and iterate on this process with respect to each of the four edges (on one side only).

- Implement the algorithm and test it thoroughly.
- Prove its correctness.

- Analyze its running time in the worst case. (Extra credit for any intelligent remarks about what its expected complexity might be, making reasonable assumptions about the point distribution, etc. Justify your answers.)

Does *Quickhull* deserve its name?

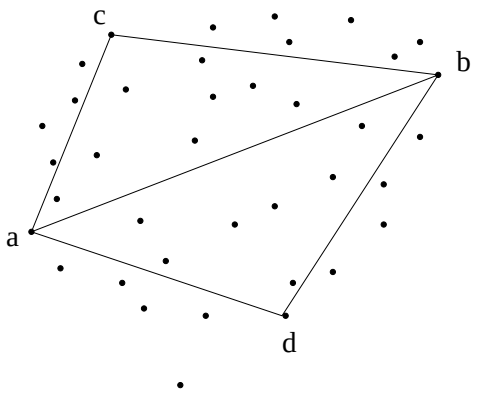

Figure 2: *Quickhull*

3 Testing Point Enclosure

Justify the following statement. Given a convex n-gon, it is possible to build a data structure of size $O(n)$ with the following characteristics: Given an arbitrary point (the query), whether it lies inside the polygon or not can be decided in $O(\log n)$ time.

4 Testing Separation

Justify the following statement. Given n points in the plane, it is possible to build a data structure of size $O(n)$ with the following characteristics: Given an arbitrary line (the query), whether the n points all lie on the same side can be decided in $O(\log n)$ time.

5 Computing Maxima

Design and analyze an efficient algorithm for finding all the maxima of a given set of points in the plane. Can you use your algorithm to build a data structure, which given a new point, allows you to find in logarithmic time whether it is a maximum or not?

6 Algorithmic Voyeurism

Design a linear algorithm, which given n segments parallel to each other, determines if a line can pass through all of them (assume that the order of the segment is available beforehand). How hard is it to compute an implicit description of all the stabbing lines?

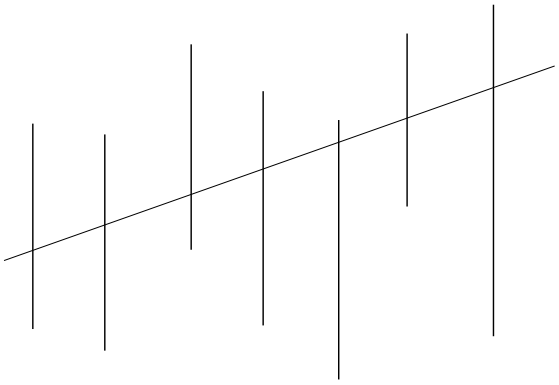

Figure 3: *Stabbing lines*

7 The Complexity of Simplicity

Given n points in the plane, how can you construct a simple polygon having each point as a vertex and none other? Still more interesting is to prove that in the worst case $\Omega(n \log n)$ operations are required. Why is that so?