CMSC 754:Spring 2020 Dave Mount

Homework 1: Convexity and Hulls

Handed out Tuesday, Feb 11. Due at the start of class on Tuesday, Feb 18. Late homeworks are not accepted (unless an extension has been prearranged) so please turn in whatever you have completed by the due date. Unless otherwise specified, you may assume that all inputs are given in *general position*.

Problem 1. In this problem, we will consider a few applications of orientation testing. In each case, express the answer in terms of orientation tests in 2- or 3-dimensions.

(a) You are given four points a, b, c, and d in $\mathbb{R}^2$. Determine whether the line segments $\overline{ab}$ and $\overline{cd}$ intersect (see Fig. 1(a)). (You do *not* need to compute the intersection point itself. If they intersect, you may assume that they intersect in a single point in their interiors.)

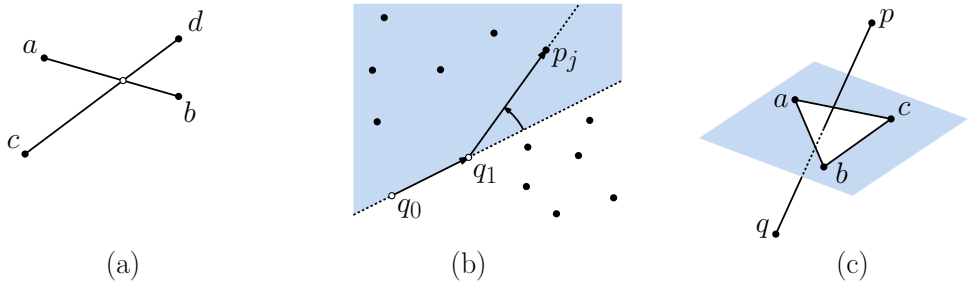

Figure 1: Orientation testing.

(b) You are given a set of points $P = \{p_1, \ldots, p_n\}$ in $\mathbb{R}^2$ and two additional points q_0 and q_1. Among the points of P that lie strictly to the left of the directed line $\overrightarrow{q_0q_1}$, compute the point p_j that minimizes the counterclockwise angle between the rays $\overrightarrow{q_0q_1}$ and $\overrightarrow{q_1p_j}$ (see Fig. 1(b)). If no point of P lies to the left of $\overrightarrow{q_0q_1}$, your algorithm should report this. Your solution should employ $O(n)$ orientation tests. (Note: The notion of being "left" of a directed line means on the left side of the line with respect to an observer facing the line's direction.)

(c) You are given five points a, b, c, p and q in $\mathbb{R}^3$. Determine whether the line segment $\overline{pq}$ intersects the triangle $\triangle abc$ (see Fig. 1(c)). (You may assume that p and q do not lie in the plane spanned by $\triangle abc$ and that if there is an intersection, it occurs in the interior of the triangle.)

Problem 2. Consider a convex polygon P presented as a counterclockwise sequence of vertices $\langle p_1, \ldots, p_n \rangle$, and let q be a point that is external to P (see Fig. 2). Present an $O(\log n)$-time algorithm that computes a vertex p_j that defines a support line passing through q so that P lies to the left of the directed line $\overrightarrow{qp_j}$. For full credit, your algorithm should access points only through orientation tests.

Hint: Indexing and bisecting circular arrays can be a bit messy given issues with wrapping around. Given two vertices p_i and p_k, let $\text{ccw}(i, k)$ denote the subsequence of vertices along the boundary of P between p_i and p_k in counterclockwise order. You may assume that you have access to a function $\text{bisect}(i, k)$ which in $O(1)$ time returns an index m that splits this vertex chain into two subsequences $\text{ccw}(i, m)$ and $\text{ccw}(m, k)$ each having roughly an equal number of vertices.

Problem 3. The objective of this problem is to explore possible variations in the guessing sequence for Chan's convex hull algorithm for an n-element point set P. Let h denote the size of the final convex hull. Recall that $\text{ConditionalHull}(P, h^*)$ returns the convex hull of P if $h^* \geq h$ and "fail" otherwise, and the algorithm iteratively guesses values of h^* through repeated squaring. Here is a equivalent version of the one given in class.

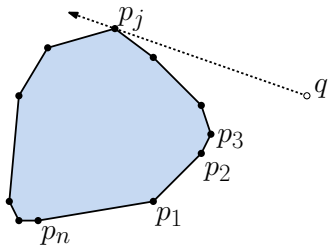

Figure 2: Orientation testing.

(1) $i \leftarrow 1$; status $\leftarrow$ fail
(2) while (status == fail)
  (a) $h^* \leftarrow \min(2^{2^i}, n)$
  (b) status $\leftarrow$ ConditionalHull(P, h^*)
  (c) $i \leftarrow i + 1$
(3) return status

What if we do not use repeated squaring? In each of the following cases, indicate what the running time of Chan's algorithm *would be* had we replaced the expression in line (2a). (Express your answer using O-notation as a function of n and h.) In each case, explain how you derived your answer.

(a) $h^* \leftarrow \min(i^2, n)$ (quadratic progression)
(b) $h^* \leftarrow \min(2^i, n)$ (repeated doubling)
(c) $h^* \leftarrow \min(\sqrt{2}^{\sqrt{2}^i}, n)$

You may assume any standard results on summations, e.g., the Wikipedia page on summations.

Problem 4. A *polygonal chain* in the plane is a sequence of vertices $C = \langle p_1, \ldots, p_n \rangle$, where each consecutive pair (p_i, p_{i+1}) is connected by a line segment, called an *edge*. Such a chain is said to be *strictly horizontally monotone* if any vertical line intersects the chain in a single point. A collection of chains is said to be *independent* if no two intersect each other (see Fig. 3).

Figure 3: A set of independent monotone polygonal chains.

Present an algorithm which, given a set of k strictly monotone polygonal chains $\mathcal{C} = \{C_1, \ldots, C_k\}$, determines whether they are independent. Your algorithm does not need to report the intersections, it just needs indicate whether any intersection exists. Let n_i denote the number of vertices in the ith chain, and let $n = \sum_{i=1}^{k} n_i$ be the total size of all the chains. Your algorithm should run in time $O(n \log k)$. (**Hint:** Use plane sweep, but do it efficiently.)

Challenge Problem. (Challenge problems count for extra credit points. These additional points are factored in only after the final cutoffs have been set, and can only increase your final grade.)

Given a set $P = \{p_1, \ldots, p_n\}$ of n points in the plane, the *minimum horizontal trapezoid* is a trapezoid of minimum area that contains P such that two of its sides are horizontal (see Fig. 4(b)). If there are multiple trapezoids of the same area, any one of them may be chosen.

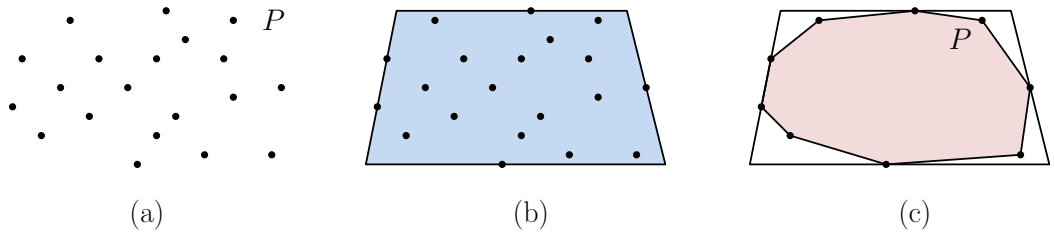

Figure 4: Challenge Problem: Minimum horizontal trapezoid

(a) Clearly, the horizontal sides of the trapezoid pass through topmost and bottommost points of P. What properties should the left and right sides have to yield the minimum area?

(b) Based on your answer to (a), present an $O(n \log n)$ time algorithm for computing the minimum horizontal trapezoid of P.

(c) Suppose that P is given as a convex polygon with n vertices (see Fig. 4(c)). Explain how to compute the minimum horizontal trapezoid in $O(\log n)$ time.

Some tips about writing algorithms: Throughout the semester, whenever you are asked to present an "algorithm," you should present three things: the algorithm, an informal proof of its correctness, and a derivation of its running time. Remember that your description is intended to be read by a human, not a compiler, so conciseness and clarity are preferred over technical details. Unless otherwise stated, you may use any results from class, or results from any standard textbook on algorithms and data structures. Also, you may use results from geometry that: (1) have been mentioned in class, (2) would be known to someone who knows basic geometry or linear algebra, or (3) is intuitively obvious. If you are unsure, please feel free to check with me.

Giving careful and rigorous proofs can be quite cumbersome in geometry, and so you are encouraged to use intuition and give illustrations whenever appropriate. Beware, however, that a poorly drawn figure can make certain erroneous hypotheses appear to be "obviously correct."

Throughout the semester, unless otherwise stated, you may assume that input objects are in *general position*. For example, you may assume that no two points have the same x-coordinate, no three points are collinear, no four points are cocircular. Also, unless otherwise stated, you may assume that any geometric primitive involving a constant number of objects each of constant complexity can be computed in $O(1)$ time.