Introduction to Computational Geometry

Homework Assignment 1

Give written (and legible) or typeset answers to the following questions. Please give succinct and mathematically precise answers. From now on, whenever you are asked to give an algorithm for a problem, you should present the following: the algorithm, a justification of its correctness, and a derivation of its running time. Proving an algorithm correct can get messy for geometric problems (in part because of many special cases), so using intuition and illustrations is acceptable. However, be warned that “proof by picture” is a common pitfall in computational geometry (because it is easy to miss the aforementioned special cases!). There is no need to give detailed C++/Java/Python code, but do write clear and convincing pseudo-code for your algorithms. All work must be done independently.

Question 1. (10 points) Prove that the intersection of two convex sets is convex.

Question 2. (10 points) Let $E$ be an unsorted set of $n$ segments that are the edges of a convex polygon. (Assume that the edges are given as pairs of vertices, and the coordinates of the vertices are also given.) Describe an algorithm that computes from $E$ a list containing all vertices of the polygon, sorted in clockwise order. Your algorithm should run in $O(n \log n)$ time.

Question 3. (15 points) 1. Give pseudo-code to determine if a point $t$ lies inside a triangle with vertices $p$, $q$, and $r$ (in clockwise order, say). A value of true should be returned if $t$ lies on the boundary or inside the triangle, and false otherwise. See Figure 1.

![Figure 1: (a) $t$ lies outside the triangle (b) $t$ lies inside the triangle.](image)

2. Assume that the vertices $p_1, p_2, \ldots, p_n$ of an $n$-sided convex polygon $P$ are given in clockwise order. Give an algorithm that tests if a query point $t$ lies inside the given convex polygon (i.e., it should return true if $t$ lies inside the polygon or on the boundary, and false otherwise). Your algorithm should run in $O(\log n)$ time.

   Hint: The pseudo-code from part 1 might come in useful here.

Question 4. (20 points) Exercise 1.7 from the textbook. (A scanned copy of the problem is available on Sakai.)
Question 5. (25 points) There are many situations in which one needs to compute the convex hull of objects other than points. Figure 2(a) shows the convex hull of a set of line segments, and Figure 2(b) shows the convex hull of a simple polygon. The hull is drawn in dashed lines.

Figure 2: (a) Convex hull of a set of line segments. (b) Convex hull of a simple polygon. (c) An x-monotone polygon.

1. Let $S$ be a set of $n$ line segments in the plane. Prove that the convex hull of $S$ is exactly the same as the convex hull of the $2n$ endpoints of the segments.

2. A polygonal chain is $x$-monotone if any vertical line intersects it at most once. A simple polygon is $x$-monotone if its boundary can be partitioned into two chains that are $x$-monotone. See Figure 2(c). Note that the polygon in Figure 2(b) is not $x$-monotone. The convex hull of a simple polygon is simply the convex hull of its vertices. Show that the convex hull of an $x$-monotone polygon can be computed in linear time. Assume that the vertices of the polygon are given in order (clockwise, say).

3. Now consider an arbitrary simple $n$-gon $P$. Name the vertices of $P$ in counter-clockwise order about the boundary, starting at the rightmost vertex, as $p_0, p_1, p_2, \ldots, p_{n-1}$. Recall that as we walk around the convex hull in counter-clockwise order, we make only left turns. Consider the following algorithm that claims to find the convex hull of $P$: Let first, second, and third be three consecutive vertices of $P$ in counter-clockwise order, with first initialized to $p_0$. There are two cases (refer to Figure 3):

Figure 3: Two cases ((a) and (b)) for Sklansky’s algorithm.

(a) first, second, and third form a left turn, or are collinear. In this case we advance first, second, and third by one vertex along the boundary (so that the old second is now first, the old third is now second, and the vertex following the old third is the new third.

(b) first, second, and third form a right turn. In this case, we know that second cannot be on the convex hull, so we remove it from further consideration. Then we
“backtrack”, that is the vertex before first becomes the new first, the old first
is now second, and the old third continues to be third.

Do the above step continually until third is p0 and first, second, and third form a
left turn. The claim is that when the algorithm terminates, the remaining set of edges
form the convex hull of the simple polygon. This is a linear time algorithm, which was
given by Sklansky in 1972.

As it turns out, the algorithm is incorrect. It fails to produce the convex hull in some
instances. While there are classes of polygons for which this algorithm correctly produces
the convex hull, it does not do so for arbitrary simple n-gons. In this question, you are
required to give a counter-example to the above algorithm. In other words, draw a
polygon on which the above algorithm fails.

In fact, there are linear time algorithms to find the convex hull of a simple polygon. However, you are not required to come up with one here.

Question 6. (20 points) The convex hull is a somewhat non-robust shape descriptor, since a
few distant outlying points can dominate the shape of the hull. A more robust method to
describe the shape of a collection of points is by constructing nested layers of convex hulls, as described here: Given a planar point set \( P \) in general position (see Fig. 1(a)), let \( H_1 \) be
the convex hull of \( P \). Remove the vertices of \( H_1 \) from \( P \) and compute the convex hull of the
remaining points, call it \( H_2 \). Repeat this until no more points remain, letting \( H_1, \ldots, H_k \)
denote the resulting hulls (see Fig. 1(b)). More formally, \( H_i = \text{conv}(P \setminus (\bigcup_{j=1}^{i-1} \text{vert}(H_j))) \).
The final result is a collection of nested convex polygons, where the last one may degenerate
to a single line segment or a single point.

Figure 4: Nested hulls

1. Assuming that the points are in general position in \( \mathbb{R}^2 \), as a function of \( n \), what is the
maximum number of hulls that can be generated by this process? (I am looking for an
exact formula, not an asymptotic one. For every \( n \), there should exist a point set that
exactly achieves your bound.) Briefly justify your answer.

2. Given a set \( P \) of \( n \) points in the plane, devise an \( O(n^2) \) time algorithm to compute this
iterated sequence of hulls. (FYI: \( O(n \log n) \) is possible, but complicated.)

3. Prove the following lemma: Given a planar point set \( P \) in general position, let \( k \) denote
the number of hulls generated by the repeated hull process. There exists a point \( q \) in
the plane (it need not be in \( P \) ) such that, every closed halfplane whose bounding line \( l \)
passing through \( q \) contains at least \( k \) points of \( P \). (A closed halfplane is the region of the plane lying on or to one side of a line. An example of such a point for \( k = 4 \) is shown in Fig. 1(c).)

Please submit your assignment (as one pdf file) on Sakai by 11:59PM on February 7, 2018.