AMS 345/CSE 355 (Fall, 2022) Joe Mitchell

COMPUTATIONAL GEOMETRY
Homework Set # 2

Due (upload to Blackboard) on Tuesday, September 20, 2022.

Relevant Lecture Modules: 9–14.

Required Reading: Devadoss-O'Rourke, Chapter 2 (sections 2.1–2.6), and the Appendix (on computational complexity); O'Rourke, Chapter 3 (sections 3.1–3.8).

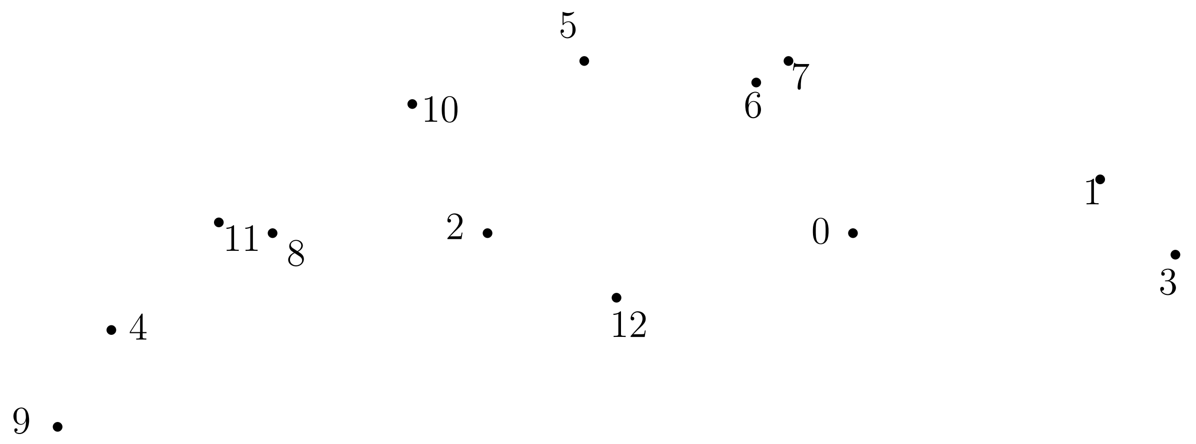
In all of the exercises, be sure to give at least a brief explanation or justification for each claim that you make.

PROBLEMS TO TURN IN:

(1). [5 points] (a). Let S be the set of points {(3,1), (4,4), (5,-1), (2,-1), (2,2), (2,1), (4,-1), (6,1), (8,1), (8,5), (7,4), (3,3)}. (Specifically, consider the points in S to be $p_0 = (3,1)$, $p_1 = (4,4)$, $p_2 = (5,-1), \ldots, p_{11} = (3,3)$, so the 12 points are indexed 0-11, in the order given.) When the Graham Scan code of Section 3.5 is run on this data, what is the sorting (as in Table 3.1 of [O'Rourke]) and what is the history of the stack (as in the example on page 86 of [O'Rourke])? Plot the points and the resulting convex hull.

(b). In Graham's convex hull algorithm, as implemented in O'Rourke, we sort the n input points by angle about a bottommost point, and then, scanning through the sorted list of points, maintain a stack, representing a left-turning chain of (input) points that are the vertices of the convex hull so far in the scan. Suppose now that we skip the sorting step; specifically, assume that we are given an input sequence of points, $(p_0, p_1, p_2, \ldots, p_{n-1})$, and we know that p_0 is the (unique) bottommost point and that (p_0, p_1) is one edge of the convex hull of the points (i.e., all points $p_2, \ldots, p_{n-1}$ lie to the left of the oriented line through p_0 and p_1). Nothing further is assumed about the ordering of the input points. (Note that we could easily find two points that form an edge of the convex hull, like p_0p_1, in $O(n)$ time, by doing one step of gift-wrapping.) Now, consider performing the Graham scan algorithm to this (unsorted) list of points, maintaining a stack representing a left-turning chain, pushing a new point (in the order given by the input) if it is to the left of the oriented line defined by the top two elements of the stack, and, if not, popping the stack, until we can push it on the stack while keeping the left-turning property. (That is, we exactly do the Graham scan algorithm, but we just use the order of the input points, without sorting them in any way, other than knowing that p_0p_1 is an edge of the hull.)
(i). This algorithm runs in time $O(n)$. Give an explanation why this is true, as precisely and succinctly and clearly as you can.
(ii). Give an example (having no degeneracies – no two points have the same x- or the same y-coordinate, and no three points are collinear) showing, though, that it can give an incorrect answer. Explicitly list the points in your example: what are the coordinates of p_0, p_1, p_2, etc? Also show a picture. Explain briefly what goes wrong.

(2). [5 points] (a). Consider running the QuickHull algorithm to compute the upper convex hull of the set $S = \{p_0, p_1, \ldots, p_{12}\}$ of $n = 13$ points shown below. The first call is to QuickHull(p_9, p_3, $\{p_0, p_1, p_2, p_4, p_5, p_6, p_7, p_8, p_{10}, p_{11}, p_{12}\}$), which is to determine the upper convex hull from $a = p_9$ to $b = p_3$, with respect to the set $\{p_0, p_1, p_2, p_4, p_5, p_6, p_7, p_8, p_{10}, p_{11}, p_{12}\}$. Now, QuickHull recursively will make several calls, to complete the computation. Explicitly give those recursive calls (for each call to QuickHull(a, b, S), give explicitly the values of the points a, b, and the set S).



(b). In our description of the QuickHull algorithm to compute the upper convex hull of a set S of points from leftmost point a to rightmost point b, we computed a "split" point, c, based on selecting c to be the point of S above segment ab that maximizes the distance from c to the line through a and b (equivalently, we maximized the vertical distance from c to the line through a and b).

Consider alternative ways of selecting c below. For each, state whether or not it always works (yields a correct algorithm for the upper hull); justify briefly – if it works, explain briefly why it works; if it does not always work, give an explicit example for which it fails, and explain your example (why it fails).

You may assume that the points S are in general position (no three are collinear) and are all above the line through a and b (points that are below the line will never be vertices of the upper convex hull of S).

(i). Pick c to maximize the angle $\angle abc$

(ii). Pick c to minimize the angle $\angle abc$

(iii). Pick c to maximize $c_y - a_y$, where c_y is the y-coordinate of c and a_y is the y-coordinate of a.

(iv). Pick c to maximize the sum of the (Euclidean) distances $d(a, c) + d(b, c)$, from a to c and from b to c.

(v). Pick c to minimize the sum of the (Euclidean) distances $d(a, c) + d(b, c)$, from a to c and from b to c.

ADDITIONAL PRACTICE PROBLEMS: TRY THEM, UNDERSTAND THEM, YOU ARE RESPONSIBLE FOR THEM

(3). (a). *Devadoss-O'Rourke: Problem 2.12: Alter the incremental algorithm so that it still works for point sets that may have two or more points with the same x-coordinate, without rotating the set into general position.*

(b). *Devadoss-O'Rourke: Problem 2.19: Describe a point set with n points that serves as the worst-case for the gift-wrapping algorithm. What is its time complexity in this case?*
Devadoss-O'Rourke: Problem 2.20: Describe a point set with n points that constitutes the best-case for the gift-wrapping algorithm. What is its time complexity in this case?

(c). (related to *Devadoss-O'Rourke: Problem 2.23*): Given a point set S, design an algorithm that finds a *star-shaped* (i.e., 1-guardable) simple polygon whose vertices are precisely S.

(d). *Devadoss-O'Rourke: Problem 2.31: It might seem that the highest and lowest points of A and B should always be the points of tangency we are looking for. Find examples where this is not the case.* Draw your example carefully, showing what the correct points of tangency are, and showing also the highest and lowest points, making it clear that they do not give the common tangent that we seek.

(4). (a). O'Rourke, problem 4, section 3.2.3, page 68: An *affine combination* of points $x_1, \ldots, x_k$ is a sum of the form $\alpha_1 x_1 + \cdots + \alpha_k x_k$, with $\alpha_1 + \cdots + \alpha_k = 1$. Note that this differs from the definition of a convex combination in that the condition $\alpha_i \geq 0$ is dropped. In two dimensions, what is the affine hull of two points? Three points? $n > 3$ points? In three dimensions, what is the affine hull of two points? Three points? Four points? $n > 4$ points?

(b). The *green combination* of points $x_1, \ldots, x_k$ is a sum of the form $\alpha_1 x_1 + \cdots + \alpha_k x_k$, with $\alpha_1 + \cdots + \alpha_k = 1$, $\alpha_i \geq 0.25$. What does the green combination of two points in the plane look like? Describe and give some examples: specifically, draw the green combinations of the following pairs of points: $\{(0,0),(1,1)\}$, $\{(1,2),(2,2)\}$, $\{(-1,0),(1,0)\}$.

(5). O'Rourke, problem 4, section 3.4.1, page 72: *QuickHull worst case: Construct a generic point set that forces QuickHull to its worst-case quadratic behavior. By "generic" is meant a construction that works for arbitrarily large values of n (i.e., "general" n),*

(6). (a). Let S be the set of points {(10,4), (10,0), (4,1), (1,4), (4,5), (5,3), (12,3), (-1,1), (-1,3), (-1,0), (4,-3), (1,-3), (4,-1), (0,1)}. When the Graham Scan code of Section 3.5 is run on this data, what is the sorting (as in Table 3.1) and what is the history of the stack (as in the example on page 86)? Plot the points and the resulting convex hull.

(b). Give a specific example of a set S of points in the plane, having integer coordinates (give them!) so that, when Graham's algorithm the history of the stack, and the value of `i` at the top of the `while` loop is as follows (as on page 86 of [O'Rourke]):

```
i = 2 : 5, 2
i = 3 : 3, 5, 2
i = 4 : 0, 3, 5, 2
i = 4 : 3, 5, 2
i = 4 : 5, 2
i = 5 : 6, 5, 2
i = 6 : 4, 6, 5, 2
```

Draw your example clearly (in addition to giving the integer coordinates of the 7 points of S).

(7). The algorithm below is proposed to find the convex hull of a simple polygon P that has n vertices, in $O(n)$ time. The goal is to avoid the need to *sort*, by using the order given by the vertices of P, which are assumed to be given in clockwise order around the boundary of P. The algorithm is essentially just doing a variant of the Graham Scan, using the order of the vertices about P in the incremental insertion.

Show that this algorithm can fail by finding a counterexample (an instance of a simple polygon) for which the output is erroneous. Explain.

Algorithm: *Let the stack Q be initialized as (q_0, q_1), where q_0 is the leftmost point of the polygon and q_1 is the (clockwise) successor of q_0 in the list of points of the polygon. March around the polygon (clockwise), starting with the successor of q_1. If at some stage the stack is $Q = (q_0, \ldots, q_i)$, with $i \geq 1$, then let p be the successor of q_i. If (q_{i-1}, q_i, p) is a* right *turn, then push p onto the stack Q; otherwise, pop Q (and continue popping Q) until this right-turn condition is satisfied or until there is only one point remaining in the stack, and then push p onto Q. Stop when you return to point q_0.*