CMSC 754: Fall 2023 Dave Mount

Homework 1: Convex Polygons and Plane Sweep

Handed out Tuesday, Sep 12. Due by the start of class on Thursday, Sep 21. (Submissions will be through Gradescope. Submission information will be forthcoming.) 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.* Listed point values are subject to change.

Problem 1. (10 points) You are given a set of four points in the plane $\{p_1, \ldots, p_4\}$, where $p_i = (x_i, y_i)$.

(a) (5 points) Present an efficient boolean function `quadSeq`(p_1, p_2, p_3, p_4) (in pseudocode) that determines whether this sequence of points define the vertices of a convex quadrilateral in cyclic order (either clockwise or counterclockwise).

(b) (5 points) Present an efficient boolean function `quadSet`(p_1, p_2, p_3, p_4) (in pseudocode) that determines whether this set of points define the vertices of a convex quadrilateral in any order.

In this problem you may *not* assume that points are in general position. If any three points are collinear, then both of the above functions should return `false` (see Fig. 1 for some examples.)

`quadSeq(p4, p3, p1, p2)` $\rightarrow$ `true`
`quadSeq(p1, p3, p4, p2)` $\rightarrow$ `true`
`quadSeq(p1, p4, p3, p2)` $\rightarrow$ `false`
All `quadSet` calls return `true`

All `quadSeq` calls return `false`
All `quadSet` calls return `false`

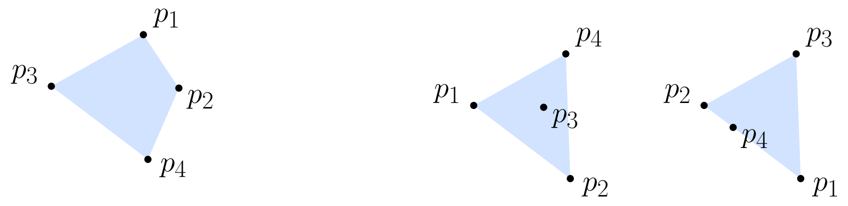

Figure 1: Examples of return results for `quadSeq` and `quadSet`.

As with any procedure, you may create variables to store intermediate values or define additional subroutines. Explain how your procedure works in English and justify its correctness. For full credit, your procedure should be based solely on orientation tests involving these four points. (E.g., a comparison of the form "`if (orient(p1, p2, p4) < 0)`" is valid, but "`if (x1 < x2)`" is not.) You should not invoke any convex hull algorithms (like Graham's algorithm). Try to do it with the smallest possible number of orientation tests. In each case, indicate how many orientation tests your procedure makes in the worst case.

Hint: Your answer to (b) is allowed to invoke your answer to (a). For `quadSet`, I think that a good way to start would be to fix two points, say p_1 and p_2, and check whether the other two points share the same orientation with respect to them, that is, "`if (orient(p1, p2, p3) == orient(p1, p2, p4))`".)

Problem 2. (15 points) Define a *chord* of a convex polygon to be any line segment $\overline{pq}$ where p and q are points on P's boundary (see Fig. 2(a)). Given a convex polygon P, we say that a parallelogram Q *circumscribes* P if it tightly encloses P. That is, P is contained within Q, and each of Q's sides touches P (see Fig. 2(b)). If Q circumscribes P, then for each pair of parallel sides of Q, there will be at least one chord common to both P and Q that is incident to the these two sides (as with $\overline{pq}$ in Fig. 2(b)). This is called a *common chord*.

A circumscribing parallelogram is *perfect* if for each pair of parallel sides of Q, there is a common chord incident to these two sides that is parallel to Q's other sides (see Fig. 2(c)). A circumscribing parallelograpm is *half-perfect* if there exists one such chord (see Fig. 2(d)).

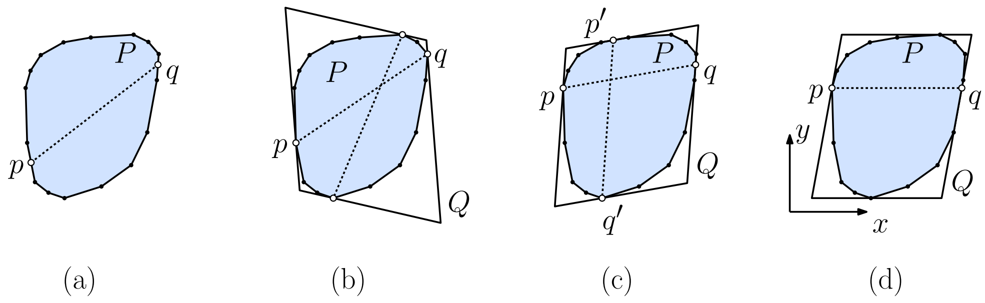

Figure 2: (a) A chord, (b) circumscribing parallelogram and common chords, (c) a perfect parallelogram (where chords $\overline{pq}$ and $\overline{p'q'}$ are parallel to Q's sides), and (d) a half-perfect parallelogram for P.

(a) (5 points) Prove that for any convex polygon P, there exists a half-perfect parallelogram where two of its sides are horizontal (that is, parallel to the x-axis) and the common chord is also horizontal (see Fig. 2(c)).

(**Hint:** Two approaches come to mind. One is based on varying the slopes of the non-horizontal sides of the parallelogram and showing that some slope yields a horizontal common chord. The other is based on varying the y-coordinate of the horizontal chord of P and showing that for some y-value, there a parallelogram that shares this chord.)

(b) (2 points) Prove that if P is in general position, the half-perfect parallelogram is unique. We will accept either of two answers. First, you can prove that multiple common chords (even if there is a unique half-perfect parallelogram) implies a violation of general position, or second, you can prove that multiple half-perfect parallelograms (even if there is a unique common chord) implies a violation of general position.

(c) (8 points) Present an $O(n)$ time algorithm which given a convex polygon P in general position, computes the half-perfect parallelogram described in parts (a) and (b). (If you can see how to do this faster, check out the Challenge Problem)

Problem 3. (10 points) The objective of this problem is to explore a natural adaptation of the notion of convex hulls to rectilinear objects. An *axis-parallel rectangle* is formally defined as the Cartesian product of two nonempty, closed intervals, one along the x-axis and one along the y-axis, that is $r = [a, b] \times [c, d]$, where $a < b$ and $c < d$ (see Fig. 3(a)).

You are given a collection of n axis-parallel rectangles in the plane $R = \{r_1, \ldots, r_n\}$, where $r_i = [a_i, b_i] \times [c_i, d_i]$. Let $\mathcal{R}$ denote the union of these rectangles, that is, $\mathcal{R} = \bigcup_{i=1}^{n} r_i$. It will be convenient to assume that $\mathcal{R}$ is *path connected*, which means that any two points $p, q \in \mathcal{R}$ are connected by a path that lies entirely within $\mathcal{R}$ (see Fig. 3(b)).

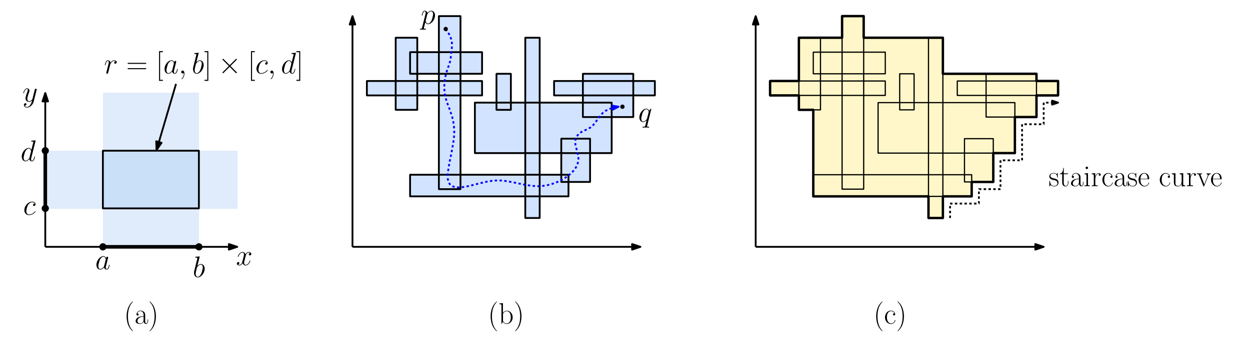

Figure 3: (a) An axis-parallel rectangle, (b) a path-connected set of rectangles, and (c) the orthogonal hull and one of its southeast staircase curve.

We say that a set H is *orthogonally convex* if the intersection of any horizontal or vertical line with H is either empty or a single line segment. The *orthogonal hull* of a set is defined to be the smallest orthogonally convex set that contains the set (see Fig. 3(c)).

(a) (8 points) Present an efficient algorithm which, given a set $R = \{r_1, \ldots, r_n\}$ of axis-parallel rectangles, computes the orthogonal hull of $\mathcal{R}$. The output of your algorithm consists of a counterclockwise sequence of edges that form the boundary of the hull. You may assume that the rectangles are in general position (and in particular, no two rectangles share the x- or y-coordinates) and their union is path connected. Explain your algorithm and derive its running time.

(**Hints:** It is a fact (which you do not need to prove) that the boundary of the orthogonal hull of a path-connected set of orthogonal rectangles can be divided into four "staircase curves", connecting the leftmost and rightmost vertical edges with the top and bottom horizontal edges (see Fig. 3(c)). Present an algorithm to build any one of these staircase curves, and argue that the others follow from symmetry. Finally, combine the four staircases to form the final output. Try plane sweep. You should aim for an overall running time of $O(n \log n)$.)

(b) (2 points) In the problem description, we made the convenience assumption that $\mathcal{R}$ is path connected. Show (by giving an example and an explanation) that if this were not true, the orthogonal hull might consist of multiple connected components.

Problem 4. (15 points) We are given a set of axis-parallel rectangles $R = \{r_1, \ldots, r_n\}$ in the plane, where $r_i = [a_i, b_i] \times [c_i, d_i]$. All the rectangles lie within a horizontal strip $\mathcal{S} = \{(x, y) \mid 0 \leq y \leq 1\}$ (see Fig. 4(a)).

In this application we imagine that water is flowing along the "stream" $\mathcal{S}$ from left to right, and it must flow around the rectangular "rocks" of R. The question is whether any water can make it through, or do the rocks effectively block all the flow. Present an algorithm which, given R, returns `true` or `false`, depending on whether there exists a x-monotone path through S that avoids all the rectangles of R. (Such a path does exist in the example of Fig. 4(b) but not in the case of Fig. 4(c)).

Present an efficient algorithm that solves the problem. You may make the general position assumption that, except for $y = 0$ and $y = 1$, no two rectangles have the same x- or y-coordinates. Explain your algorithm and derive its running time. (**Hint:** Use plane sweep. Aim for a running time of $O((n + m) \log n)$, where m is the total number of edge-to-edge intersections among the sides of the rectangles. I believe that it is possible to improve this to $O(m + n \log n)$, but you make need to make some additional assumptions about your ordered dictionary.)

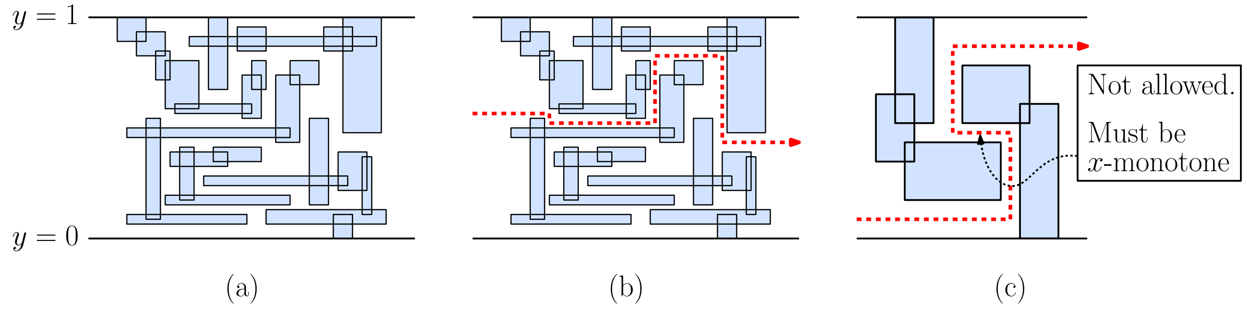

Figure 4: Does there exist an x-monotone path that avoids all the rectangles?.

Challenge Problem 1: Show that it is possible to solve Problem 2(c) in sublinear time. Explain your algorithm and derive its running time.

Challenge Problem 2: Given a convex polygon P, let Q be a parallelogram of minimum area that encloses P. Prove that Q must be perfect. (**Hint:** Show how to perturb any non-perfect parallelogram to form a new parallelogram enclosing P but with smaller area.)

Challenge Problem 3: Can you solve Problem 4 in $O(n \log n)$ time, irrespective of m? (I honestly do not know, so this may be quite challenging!)

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. (If the source is from outside of class, you must cite your sources.) 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.