CMSC 754: Fall 2026 Dave Mount

Homework 4

Due by the start of class on Tuesday, Oct 20. (Submissions will be through Gradescope.) 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. Consider the trapezoidal map shown in Fig. 1.

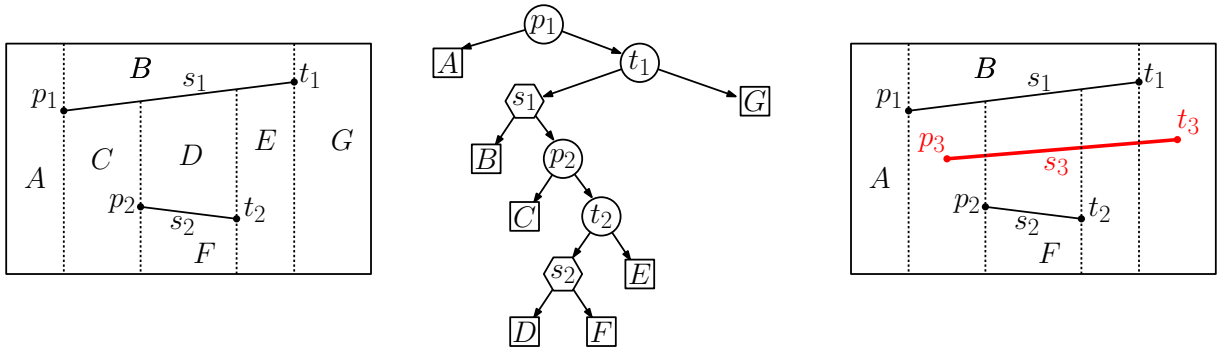


Figure 1: Trapezoidal map insertion and point location data structure.

(a) Show the trapezoidal map that results from inserting the segment $s_3 = \overline{p_3 t_3}$. **Suggestion:** To make the grading easier, label the newly created new trapezoids $H, I, J, \ldots$ from left to right.

(b) Show the point-location data structure after inserting s_3, assuming the construction given in class. **Note:** Be sure to draw internal nodes so that the left and right child pointers are clearly distinguishable. We will give partial credit if you produce a valid point-location structure, even if it is not the same as the one from the algorithm given in class.

(c) In your answer to (b), how many new nodes (both internal and leaves) were created and how many existing nodes (both internal and leaves) were removed?

Problem 2. This problem asks you to devise a data structure for answering a special type of ray-shooting query. The input consists of a $w \times \ell$ rectangle with origin at the lower-left corner and point sets $A = \{a_1, \ldots, a_m\}$ and $B = \{b_1, \ldots, b_m\}$ all lying within the rectangle, where $a_i = (a_{i,x}, a_{i,y})$ and $b_j = (b_{j,x}, b_{j,y})$. The points of A are the upper endpoints of vertical segments that extend down to the bottom of the rectangle, and the points of B are the lower endpoints of vertical segments that extend up to the top of the rectangle (see Fig. 2(a)).

The ray is determined by two quantities (see Fig. 2(a))

- the y-coordinate c along the left edge of the rectangle where your shot starts, and
- a directional vector $u = (u_x, u_y)$ of a ray along which the shot travels.

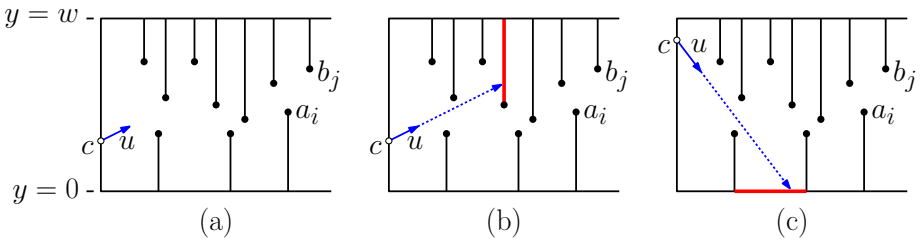


Figure 2: Ray-shooting queries.

Your objective is to preprocess A and B into a data structure so that given the triple (c, u_x, u_y), you can efficiently determine which segment is hit first (or if no segment is hit, does the ball hit the upper, lower, or right side of the rectangle). You may assume that $0 \leq c \leq w$ and $u_x > 0$. Your data structure should use $O(n+m)$ space and answer queries in time $O(\log(n+m))$. It suffices to just explain how the reduction is performed and justify its correctness, query time, and space requirements. (You do not need to explain how to compute the segments nor how to construct the point-location data structure.)

Hint: Begin by extracting the equation for the line ℓ carrying the query ray. Then show that, by applying duality, this problem reduces to point-location in $\mathbb{R}^2$, where the segments are determined by the segment vertices A and B. It may be easier to ignore the top and bottom sides of the rectangle, and imagine that the segments extend infinitely up or down. Once the result is known, consider the effect of the rectangle sides. Another useful simplification is to describe two separate data structures, one for A and the other for B, and then combine the results.

Problem 3. Computational biologists often compute physical properties of large molecules. One of these properties is something called the *accessible surface area* of the molecule. We will consider this problem in a 2-dimensional setting.

We model the atoms of our molecule as a set of unit disks, each of radius of 1, centered at a given set of points $P = \{p_1, \ldots, p_n\}$. Let b_i denote the unit disk centered at point p_i. Let $M(P) = \bigcup_{i=1}^{n} b_i$ denote the union of these disks (shaded in blue in Fig. 3(a)). Let $\partial M(P)$ denote its boundary. Note that $\partial M(P)$ consists of circular arcs and may have multiple connected components (all of which contribute to the accessible surface).

(a) Present an $O(n \log n)$-time algorithm that, given a set P of n points in the plane, computes the length of the accessible surface (see the solid curve in Fig. 3(a)).

Note: I am mostly interested in identifying the circular arcs that make up the boundary of $M(P)$, not the actual perimeter value. To simplify matters, you may assume that you have access to a black-box function that computes the length of a given circular arc.

(b) In most drug-design applications, the molecule resides in liquid solvent, such as water. A more relevant statistic is the *solvent-accessible surface*. We model a water molecule as a disk of some radius r_0. Imagine that we roll such a disk around the entire boundary of $M(P)$ and trace out the path taken by the center of this disk (the dashed curve in Fig. 3(b)). As with the accessible surface, this boundary may have multiple components, all of which contribute to the final result.

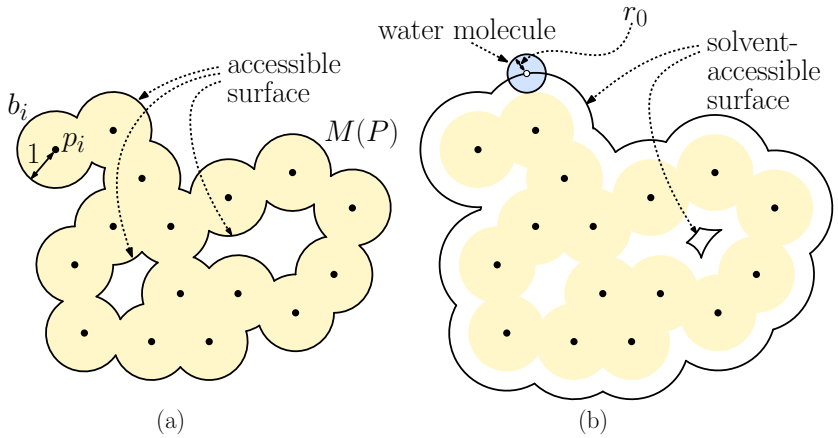


Figure 3: Accessible surface.

Present an $O(n \log n)$-time algorithm that, given P and r_0, computes the length of the solvent-accessible surface (the solid curve in Fig. 3(b)).

Hint: Modify your answer to part (a).

Problem 4. (Optional - Ungraded) Consider an n-sided simple polygon P in the plane. Let us suppose that the leftmost edge of P is vertical (see Fig. 4(a)). Let e denote this edge. Explain how to construct a data structure to answer the following queries in $O(\log n)$ time with $O(n)$ space. Given a ray r whose origin lies on e and which is directed into the interior of P, find the first edge of P that this ray hits. For example, in the figure below, the query for ray r should report edge f. (**Hint:** Reduce this to a point location query in an appropriate planar subdivision. It may help to preprocess P, say by triangulating or computing a trapezoidal map.)

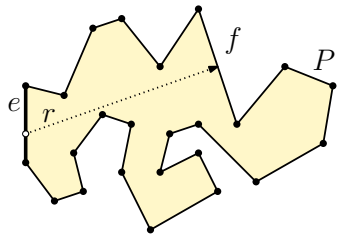


Figure 4: More ray-shooting queries.

Note: Challenge problems are not graded as part of the homework. The grades are recorded separately. After final grades have been computed, I may "bump-up" a grade that is slightly below a cutoff threshold based on these extra points. (But there is no formal rule for this.)

Challenge Problem: In class we argued that the number of parabolic arcs along the beach line in Fortune's algorithm is at most $2n - 1$. The goal of this problem is to prove this result in a somewhat more general setting.

Consider the beach line at some stage of the computation, and let $\{p_1, \ldots, p_n\}$ denote the sites that have been processed up to this point in time. Label each arc of the beach line with the index of its associated site. Reading the labels from left to right defines a string (see Fig. 5).

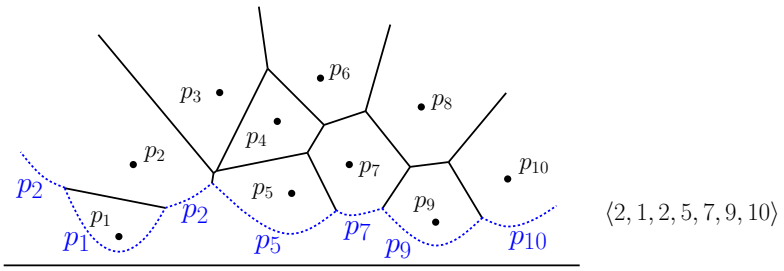


Figure 5: Beach-line complexity.

(a) Prove that for any two labels a, b, the following alternating subsequence *cannot* appear anywhere within such a string:

$$\ldots a \ldots b \ldots a \ldots b \ldots$$

(b) Prove that any string of n symbols that does not contain any repeated symbols ($\ldots aa \ldots$) and does not contain the alternating sequence of the form "$\ldots a \ldots b \ldots a \ldots b \ldots$" cannot be of length greater than $2n - 1$. (**Hint:** Use induction on n. Fix any symbol, and split the string into three substrings based on this symbol's first and last occurrences. Analyze where the remaining symbols can occur, based on the alternation restriction.)

Sequences that contain no forbidden subsequence of alternating symbols are famous in combinatorics. They are known as *Davenport-Schinzel sequences*, and their growth properties are an interesting and open problem. They have numerous applications in computational geometry, this being one.