CMSC 754:Fall 2014 Dave Mount

Homework 1: Intersections, Convex Hulls, and More

Handed out Thursday, Sep 18. Due at the start of class on Thursday, Sep 25. 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. You are given a collection of n semicircles whose centers lie along the x-axis (see Fig. 1). You may assume that each semicircle is specified by giving the x-coordinate of its center point and its radius. Derive an $O(n \log n)$ time algorithm that counts the number of intersections between these semicircles.

Hint: It may be helpful to assume that you have access to an enhanced version of an ordered dictionary, which supports rank queries, in addition to the usual operations:

- $r \leftarrow \text{insert}(x)$: Insert a value x and return a reference r to its location in the data structure
- $\text{delete}(r)$: Delete the value given by reference r
- $i \leftarrow \text{rank}(r)$: Return the number of elements whose value is less than or equal to the element referenced by r.

All operations can be performed in $O(\log m)$ time, where m is the number of entries in the dictionary.

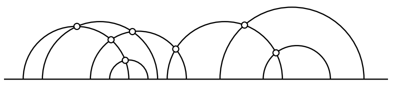


Figure 1: Problem 1

Problem 2. The objective of this problem is to explore possible variations in the guessing sequence for Chan's convex hull algorithm. Recall that we are given an n element point set P in the plane. The algorithm has the following general structure:

```
(1) $i \leftarrow 2$; $L \leftarrow$ fail
(2) while ($L \neq$ fail)
    (a) $h^* \leftarrow \min(2^{2^i}, n)$
    (b) $L \leftarrow \text{RestrictedHull}(P, h^*)$
    (c) $i \leftarrow i + 1$
(3) return $L$
```

Let h denote the actual number of vertices on the final convex hull. The function $\text{RestrictedHull}(P, h^*)$ returns the convex hull of P if $h^* \geq h$ and "fail" otherwise.

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.) Explain how you derived your answers.

(a) $h^* \leftarrow \min(i^2, n)$

(b) $h^* \leftarrow \min(2^i, n)$

(c) $h^* \leftarrow \min(2^{2^{2^i}}, n)$

You may assume any standard results on summations, e.g., http://en.wikipedia.org/wiki/Summation.

Problem 3. You are given two vertical lines and a set of n (nonvertical) line segments which pass between these vertical lines. Let a_i denote the y-coordinate where the ith line segment hits the left vertical line and let b_i denote the y-coordinate where it hits the right vertical line (see Fig. 2(a)). Scanning from left to right, whenever two segments intersect, the segment with the lower slope terminates and the one with the higher slope continues on (see Fig. 2(b)).

Present an efficient algorithm that determines for each line segment which segment caused its termination. (For example, in the example shown in Fig. 2(b), segment 1 was terminated by segment 2, segment 2 was terminated by segment 5, segment 3 was terminated by segment 5, and so on.) If a segment was not terminated (as in the cases of segments 5 and 8), you can imagine that the right vertical segment is segment $n+1$ of infinite slope that terminates all the segments that were not terminated before.

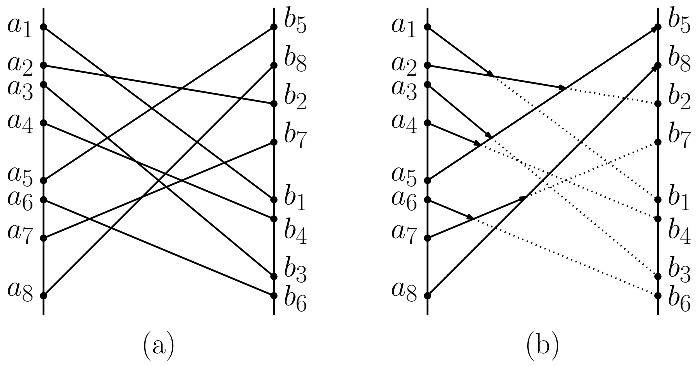


Figure 2: Problem 3

To make your algorithm's task easier, you may assume that the segments have been presorted according to any one of the following criteria (your choice): (a) in order of their intersection with the left vertical line, (b) in order of their intersection with the right vertical line, or (c) in order of slope. State which of these sorted orders you assume, and then present an $O(n)$ time algorithm, assuming the segments are given in this order. (Hint: I don't believe that all orderings admit an $O(n)$-time solution.)

Problem 4. A farmer has an orchard where various trees have been planted. Let $P = \{p_1, \ldots, p_n\}$ denote the coordinates of these trees. (The trees are very skinny, so each can be modeled by a single point. Maybe they are palm trees and the farmer sells coconuts.) The farm is bordered on east and west by roads running north-south and on the south by a road running east-west (see Fig. 3). The farmer wants to erect a straight-line fence to bound the north side of his orchard (to keep out those pesky coconut-eating armadillos). Since the city charges him tax based on the area of the farm, he wants to erect the fence to minimize the enclosed area (shaded in the figure).

Present an efficient algorithm to determine where the fence should be placed. Assuming that the upper hull of the points has already been computed (including the lower left and right corners of the property), show that it is possible to determine where to put the fence in $O(\log n)$ time. (Hint: Begin by determining what geometric properties the area-minimizing line must satisfy. You will need to include a proof of this in your solution.)

Problem 5. A cell phone service provider wants to know whether it is providing sufficient coverage to the city of College Park. For this highly simplified version of the problem, assume that College Park is modeled as a unit square (say with one corner at $(0,0)$ and the other at $(1,1)$). Each of the cell towers monitors an area that (again simplifying) is in the shape of a triangle. The union of these triangles

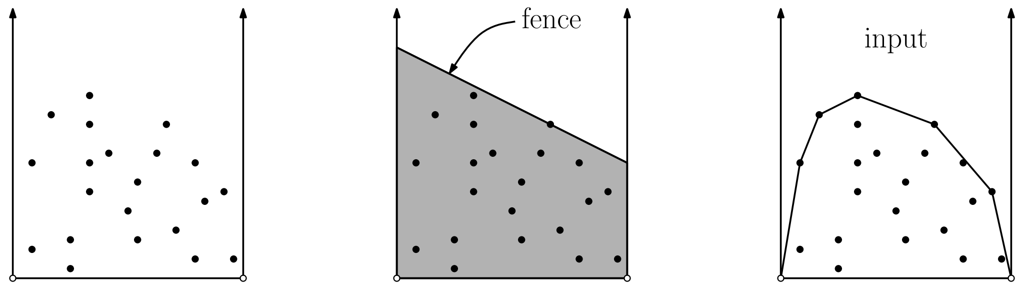


Figure 3: Problem 4

should cover the entire city. For the sake of robustness, the service provider wants each point of the city to be covered at least c times, for a given integer c.

You are to give an efficient algorithm to determine whether the current set of towers provides the desired coverage. (For example, in Fig. 4 we have shaded the regions of the city according to the number of towers that covers each region. Needless to say, this is inadequate since there are points of the city that are not covered by a single tower, probably where you live now.)

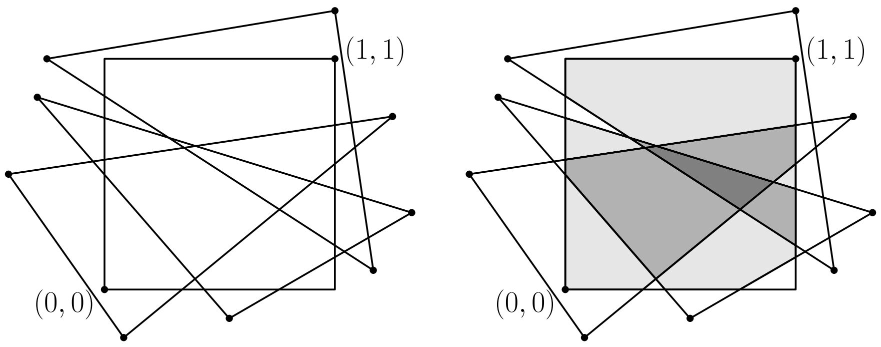


Figure 4: Problem 5

The input to your program consists of a set of n triangles, each expressed as a triple of points, and the integer c. Let $T_i = (p_i, q_i, r_i)$ denote the vertices of the ith triangle. Your program is to determine whether there exists a point of the unit square that is covered by fewer than c triangles, or indicate that there is no such point. The running time of your algorithm should be $O(n \log n + I \log n)$, where I is the number of pairs of triangles whose edges intersect each other. (Note that to be counted among the I pairs, the triangles need not intersect within the square itself.)

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.

Modify your solution to Problem 4 (orchard problem) to the following scenario. The farm is now bounded on only two sides by roads (west and south), and you are to determine the location of a fence that will create a triangle of minimum area that will bound the points of P (see Fig. 5).

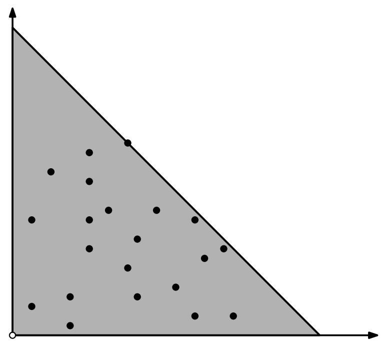


Figure 5: Challenge Problem

Some tips about writing algorithms: Henceforth, 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.

A reminder: Remember that you are allowed to discuss general solution strategies with your classmates. Since exam problems are often modifications of homework problems, it is not a good idea to pass too much information to your friends, since this will deprive them from the exploration process of winnowing down the multitude of possible solution strategies to the final one. When it comes to writing your solution, you must work independently.

Occasionally student have told me that, in the process of trying to learn more about a problem, they have searched the Web. It sometimes happens that, as a result, they discover a fact that gives away the solution or provides a major boost. While I would encourage you to try to solve the problems on your own using just the material covered in class, I have no problem with using the Web if you get stuck. My only requirement is that you cite any resources that you used. I will not deduct points for help discovered on the Web, but it is your responsibility to express the solution in your own words and to fully understand it.

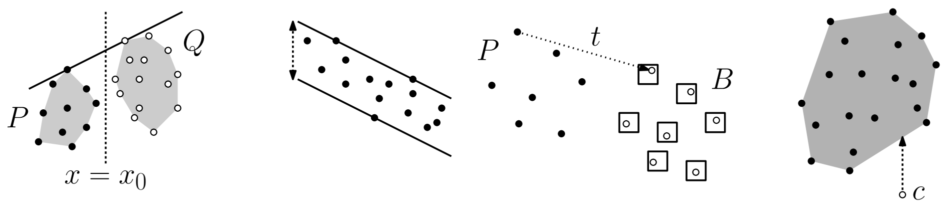
CMSC 754:Fall 2014 Dave Mount

Homework 2: Duality, LP, Trapezoidal Maps, and Point-Location

Handed out Thursday, Oct 9. Due at the start of class on Thursday, Oct 16. 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 in *general position.* Whenever asked to give an algorithm running in $O(f(n))$ time, you may give a *randomized algorithm* whose expected running time is $O(f(n))$.

Problem 1. Explain how to solve each of the following problems in $O(n)$ (expected) time. Each can be modeled as a linear programming (LP) problem, perhaps with some additional pre- and/or post-processing. In each case, explain how the problem is converted into an LP instance and how the answer to the LP instance is used/interpreted to solve the stated problem.

(a) You are given two n-element sets of points P and Q in the plane, and you are told that they are separated by a vertical line $x = x_0$, with P to the left and Q to the right (see Fig. 1(a)). Compute the line equation of the upper tangent of these two hulls (or more formally, the common support line that lies above both hulls).



(a) (b) (c) (d)

Figure 1: Problem 1

(b) Given a set P of n points in the plane, find the pair of (nonvertical) parallel lines that enclose these points such that the vertical distance between these lines is minimized (see Fig. 1(b)).

(c) The following problem arises in image registration. You are given a sequence of n points $P = \langle p_1, \ldots, p_n \rangle$ in the plane and sequence of n closed, axis-aligned rectangles $B = \langle B_1, \ldots, B_n \rangle$. Does there exist a translation vector, $t = (t_x, t_y)$ such that for $1 \leq i \leq n$, the translated point $p_i + t$ lies within the corresponding rectangle B_i? (See Fig. 1(c).)

(d) You are given a set of n points $P = \{p_1, \ldots, p_n\}$ in the plane, where $p_i = (p_{i,x}, p_{i,y})$. A car starts at position $c = (c_x, c_y)$ and travels vertically upwards, so at time $t \geq 0$, the car is at $(c_x, c_y + t)$. Present an algorithm, which given P and c, determines the smallest nonnegative value of t such that the car lies within the convex hull of P at time t (see Fig. 1(c)). If no such t exists, your algorithm should indicate this.

Problem 2. The objective of this problem is to consider a very simple backwards analysis of a classical data structure, an (unbalanced) binary search tree. You are given a set $X = \{x_1, \ldots, x_n\}$ of real numbers, called *keys.* The tree's internal nodes are labeled with elements of x. The left (resp., right) subtree of any internal node labeled x_i contains all points that smaller than (resp., larger than) x_i. If the tree has i internal nodes it has $i + 1$ leaves, each of which represents the interval between two consecutive keys (see Fig. 2(a)). To insert a new key x_i, you find the leaf node whose interval contains x_i, and then replace this leaf with an internal node labeled x_i and generate two leaves (see Fig. 2(b)).

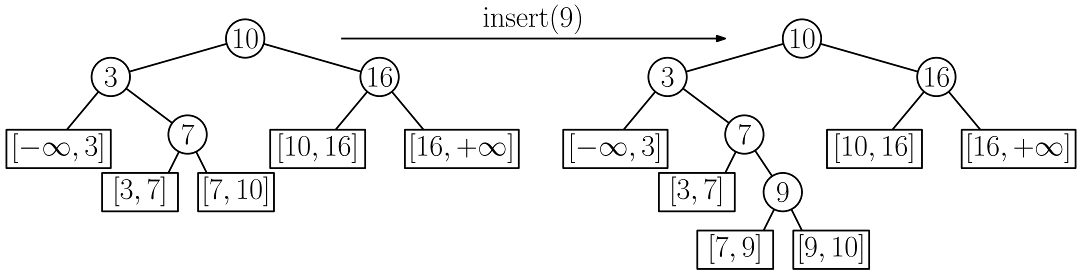


Figure 2: Problem 2: Insertion into a binary tree.

Let us suppose that the elements of X are inserted into the binary search tree in random order. Let q denote an arbitrary real number. Prove that the expected depth of the leaf node containing q is at most $c \ln n$, for some constant c. (The purpose of this exercise is to get practice with backwards analysis, so please try to solve this from first principles, rather than appealing to results proved in class or elsewhere.)

Problem 3. You are given a collection of n points $P = \{p_1, \ldots, p_n\}$ in the plane, all of which lie to the right of the y-axis. From each point p_i there is a vertical ray, denoted r_i, shooting up from this point to $+\infty$ (see Fig. 3(a)). Imagine that there is a gun that can be positioned at any point b along y-axis, and shoots a bullet along a directional vector $u = (u_x, u_y)$, where $u_x > 0$ (see Fig. 3(b)). The problem is to efficiently determine which ray (if any) is hit first by this bullet.

You are to design a data structure that uses $O(n)$ space that can answer such queries in $O(\log n)$ time. Given a query (as b, u_x, and u_y) your algorithm should either output the index of the first ray r_i that is hit by the bullet or a special status indicating that none of the rays is hit. (Hint: Using duality this can be reduced to planar point-location.)

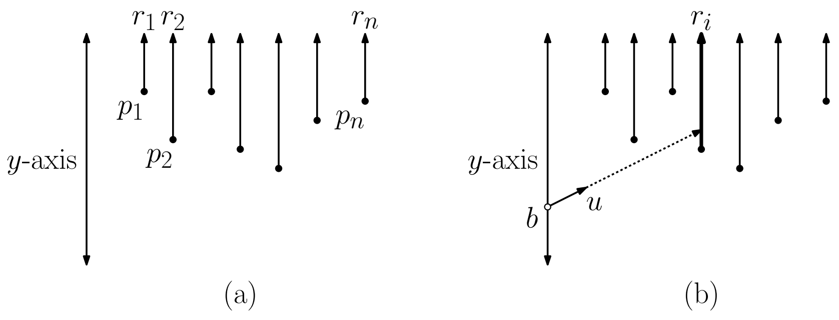
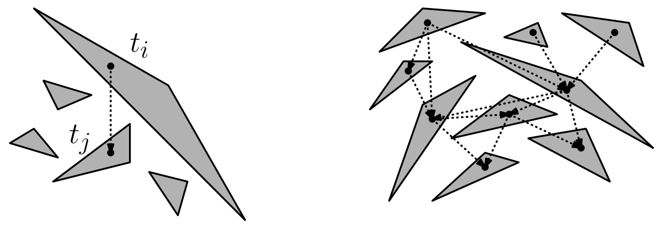


Figure 3: Problem 3: Shooting rays at rays.

Problem 4. You are given a set of non-intersecting triangles in the plane $T = \{t_1, \ldots, t_n\}$. For $1 \leq i, j \leq n$, we say that t_i is *shadows* t_j if there is a vertical line segment that connects a point of t_i above to a point of t_j below (see Fig. 4(a)). Consider a directed graph, called the *shadow graph*, whose vertices are in 1–1 correspondence with the triangles, and there is a directed edge (t_i, t_j) if and only if t_i shadows t_j.

Present an efficient algorithm that inputs a set T of n non-intersecting triangles and outputs the shadow graph for T. (Hint: $O(n \log n)$ time is possible.)



(a) (b)

Figure 4: Problem 4.

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.

(a) Prove that for any finite set T of non-intersecting triangles in the plane, the shadow graph described in Problem 4 is acyclic. (It suffices to prove the result for a set of non-intersecting line segments.)

(b) The shadow graph can easily be generalized to 3-dimensional space, where we take the vertical lines to be parallel to the z-axis. Give a counterexample that shows that the shadow graph of a set of non-intersecting triangles in 3-dimensional space may have a cycle.

CMSC 754:Fall 2014 Dave Mount

Homework 3: Voronoi/Delaunay and Arrangements

Handed out Tuesday, Oct 28. Due at the start of class on Thursday, Nov 6. 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 in *general position.* Whenever asked to give an algorithm running in $O(f(n))$ time, you may give a *randomized algorithm* whose expected running time is $O(f(n))$.

Problem 1. It is sometimes of interest to compute the Voronoi diagram of a set of sites but restricted to a lower-dimensional surface. In this problem, we'll consider a special case of this, computing the Voronoi diagram of a set of points along a line.

You are given a sequence of n sites in the plane $P = \langle p_1, \ldots, p_n \rangle$ sorted in increasing order of their x-coordinates. You are also given a horizontal line ℓ (see Fig. 1(a)). Present an algorithm that computes the Voronoi diagram of P, but restricted only to ℓ. (I don't care about the diagram above ℓ or below ℓ.)

The output consists of a sequence of (at most $n-1$) points $\langle x_1, \ldots, x_m \rangle$ where the edges of $\mathrm{Vor}(P)$ intersect ℓ, and a corresponding set of labels for each of the intervals lying between these points, indicating which site is closest (see Fig. 1(b)). Your algorithm should run in $O(n)$ time. (Hint: Start by proving that the left-to-right order of the labels is consistent with the order of the sites.)

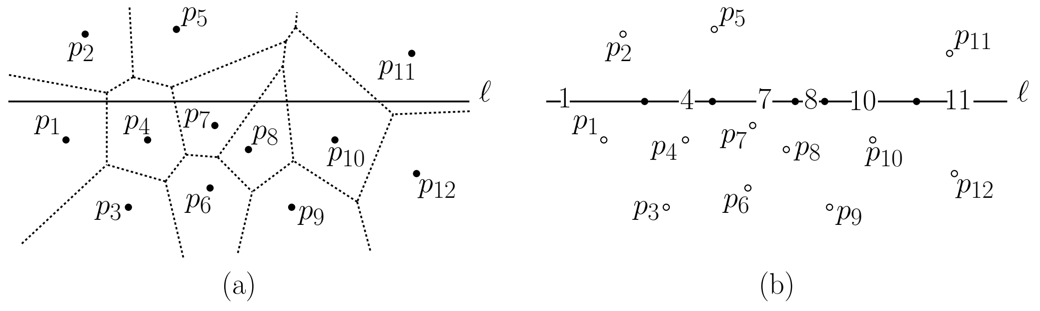


Figure 1: Problem 1.

Problem 2. Often when computing Voronoi diagrams it is desirable associate a real-valued weight with each site. The site's weight affects the distance function, and this in turn alters the shape of its Voronoi regions. There are many ways of modifying the distance function. This problem investigates one such method.

Let $P = \{p_1, \ldots, p_n\}$ denote a set of n sites in $\mathbb{R}^2$, and let $W = \{w_1, \ldots, w_n\}$ denote an associated set of weights. Define the *weighted distance* from an arbitrary point $q \in \mathbb{R}^2$ to a site p_i to be:

$$\mathrm{dist}^*(q, p_i) = \|qp_i\|^2 - w_i^2,$$

where $\|qp_i\|$ denotes the standard Euclidean distance from q to p_i. (Note that the weighted distance may be negative, particularly when q lies within a ball of radius w_i centered at p_i.) While this definition appears rather complex, it has a natural geometric intuition. The weighted distance, if positive, is the squared Euclidean distance from q to the point of tangency with a circle of radius w_i centered at p_i (see Fig. 2).

Define the *weighted Voronoi cell*, $\mathrm{Vor}^*(p_i)$, to be the set of points $q \in \mathbb{R}^2$ such that $\mathrm{dist}^*(q, p_i) \leq \mathrm{dist}^*(q, p_j)$, for all $j \neq i$. These cells partition the plane into a cell complex, which we'll call the *weighted Voronoi diagram.*

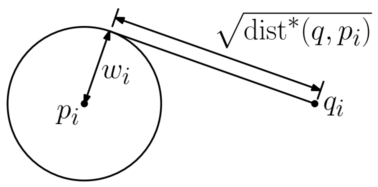


Figure 2: Problem 2.

(a) Given two sites p_i and p_j, prove that the *bisector curve* (that is, the set of points q such that $\text{dist}^*(q,p_i) = \text{dist}^*(q,p_j)$) is a line that is perpendicular to $\overline{p_i p_j}$. As a function of w_i, w_j, and $\|p_i p_j\|$, describe where this line is relative to the (unweighted) Euclidean bisector. Under what circumstances is the weighted bisector equal to the unweighted bisector.

Hint: A proof based on standard linear algebra is a bit messy. Because the Euclidean distance (and hence weighted distance) is invariant under translation and rotation, it will greatly simplify matters to assume that p_i and p_j have been transformed into a more convenient placement. For example, you may assume that the plane has been translated so that p_i is located at the origin and then rotated so that p_j lies on the positive x-axis.

(b) In the standard Voronoi diagram, every site has a nonempty Voronoi cell. Show that this is not true in the weighted case by giving a point set and a weight assignment such that at least one site has an empty Voronoi cell in the weighted diagram.

(c) Prove that for *any* weight assignment, every cite that is a vertex of the convex hull of P has a nonempty Voronoi cell in the weighted diagram.

Problem 3. This problem demonstrates an important application of computational geometry to the field of amphibian navigation. A frog who is a afraid of the water wants to cross a stream that is defined by two horizontal lines $y = y^-$ and $y = y^+$. On the surface there are n circular lily pads whose centers are at the points $P = \{p_1, \ldots, p_n\}$. The frog crosses by hopping across the circular lily pads. The lily pads grow at the same rate, so that at time t they all have radius t (see Fig. 3(a)). Help the frog out by presenting an algorithm that in $O(n \log n)$ time determines the earliest time t^* such that there exists a path from one side of the stream to the other by traveling along the lily pads (see Fig. 3(b)).

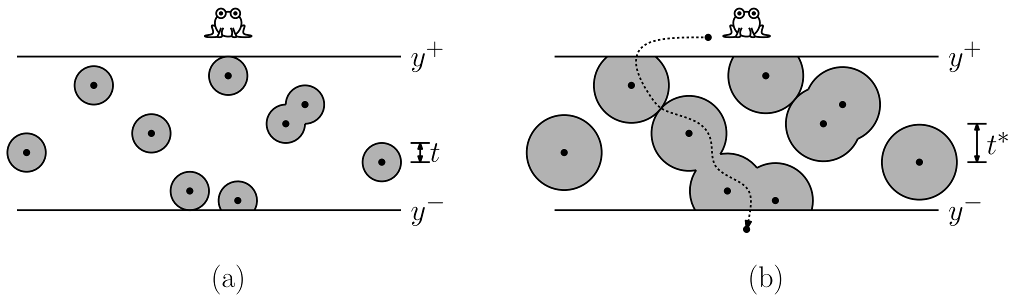


Figure 3: Problem 3.

Problem 4. Present $O(n^2)$ time algorithms for the following two problems. (The two solutions are closely related, so you can give one solution in detail and explain the modifications needed for the second one.)

(a) Given a set $S = \{s_1, \ldots, s_n\}$ of non-intersecting line segments in the plane, does there exist a line ℓ that intersects none of the segments of S such that there is at least one segment of S lying above ℓ and at least one lying below? (See Fig. 4(a).)

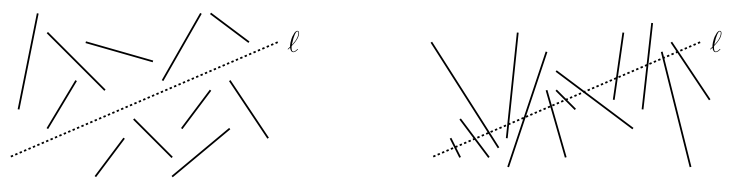


Figure 4: Problem 4.

(b) Given a set $S = \{s_1, \ldots, s_n\}$ of non-intersecting line segments in the plane, does there exist a line ℓ that intersects all of the segments of S? (See Fig. 4(b).)

Problem 5. In machine learning it is often desirable to determine whether two point sets in the plane can be partitioned by a line. When this is not possible, the goal is to find line that achieves the best possible split. Let A and B be two point sets in the plane each having n points. Given a nonvertical line ℓ, let ℓ^+ and ℓ^- denote the halfplanes lying above and below ℓ, respectively. Define ℓ's *separation defect* (see Fig. 5) to be

$$\delta(\ell) \;=\; \min\left(|A \cap \ell^+| + |B \cap \ell^-|, \;\; |A \cap \ell^-| + |B \cap \ell^+|\right).$$

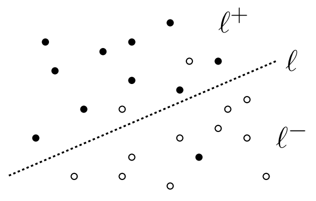


Figure 5: Problem 5: The line ℓ has separation defect 3.

Present an $O(n^2)$ algorithm that, given two n-element planar point sets A and B, computes a line ℓ that achieves the minimum separation defect. (There may generally be many. You can output any one of them. To avoid dealing with special cases in which ℓ passes through a point of A or B, you may restrict your algorithm to considering lines that do not pass through any point of either set.)

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.

Let L denote a set of n lines in the plane, and let $\mathcal{A}(L)$ denote the associated arrangement Given any face f in this arrangement, define $\deg(f)$ to be the number of vertices (equivalently the number of edges) on this face. Prove that $\sum_{f \in \mathcal{A}(L)} (\deg(f))^2 = O(n^2)$. (Hint: The proof is quite short and involves a repeated application of the Zone Theorem.)

CMSC 754:Fall 2014 Dave Mount

Homework 4: Motion Planning and Geometric Approximation

Handed out Tuesday, Nov 25. Due at the start of class on Thursday, Dec 4. 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 in *general position.* Whenever asked to give an algorithm running in $O(f(n))$ time, you may give a *randomized algorithm* whose expected running time is $O(f(n))$.

Problem 1. Given a point set P in $\mathbb{R}^2$, let $\delta_{\min}(P)$ denote the distance between the closest pair of points of P.

(a) Present an $O(n)$ time algorithm for the following problem. Given P and a candidate value $\delta^* > 0$, determine whether:

$$\delta^* < \delta_{\min}(P) \quad \text{or} \quad \delta^* = \delta_{\min}(P) \quad \text{or} \quad \delta^* > \delta_{\min}(P).$$

(Hint: Use bucketing.)

(b) Explain (without giving details) how you would generalize your algorithm to work in any (constant) dimension $\mathbb{R}^d$.

Problem 2. In class we considered motion planning problems in a static environment, where obstacles do not move. In this problem we consider the problem in a dynamic setting, which is inspired by various old video games.

You are given a *robot* that consists of a line segment of unit length that resides on the x-axis. The robot can move left or right (but not up or down) at a speed of up to one unit per second. You are given two real values x^- and x^+, and the robot must remain entirely between these two values at all times (see Fig. 1). The robot's *reference point* is its left endpoint, and at time $t = 0$, the left endpoint is located at x^-. (You may assume that $x^+ > x^- + 1$.)

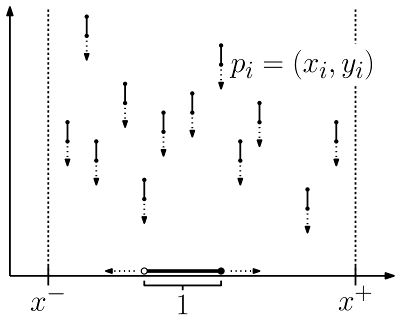


Figure 1: Problem 2.

You are also given a set of *missiles* in the form of n vertical line segments, each of length 0.2, that fall down from the sky at a rate of two units per second. Each of these vertical segments is specified by the coordinates of its lower endpoint at time $t = 0$. So, if $p_i = (x_i, y_i)$ is the starting position of the ith missile, then at time t is its lower endpoint is located at $(x_i, y_i - 2t)$, and its upper endpoint is at $(x_i, y_i - 2t + 0.2)$. You may assume that $x^- \le x_i \le x^+$.

The question is whether it is possible for the robot to move in a manner to avoid all the missiles. We will explore an algorithm for solving this problem.

(a) A natural way to define the robot's configuration at any time is as a pair (t, x), where t is the current time, and x is the location of the robot's left endpoint. Based on this, what is the C-obstacle associated with a missile whose starting position is p_i (as defined above)? In other words, describe the set of robot configurations (t, x) such that the robot intersects this missile. (Please provide low-level details, as opposed, say, to expressing this as a Minkowski sum.)

(b) Provide a complete characterization of the properties of a path in configuration space (assuming it exists) that corresponds to a motion plan for the robot that satisfies the robot's speed constraints and avoids all the missiles. Be sure to include constraints on the path's starting and ending positions and include the robot's maximum speed.

(c) Based on your answer to (b), present an $O(n \log n)$ time algorithm that determines whether a valid motion exists for the robot. Your algorithm need only answer "yes" or "no"—it does not need to output the path.

(Hint: Use plane sweep. Along the sweep line, maintain the portion that is accessible to the robot. In my solution there were *lots* of different types of events to consider. I'll be happy if you can describe the main features without getting into a full description of all the event types.)

Problem 3. In class we gave an $O(n)$ upper bound on the complexity of the union of a set of polygonal pseudodisks with n vertices in total. In this problem we are interested in getting precise bounds.

(a) Assume that the union boundary contains m original vertices of the polygons. Show that the complexity of the union boundary is at most $2n - m$. (Clearly, $m \geq 3$, and so this implies an immediate upper bound of $2n - 3$.)

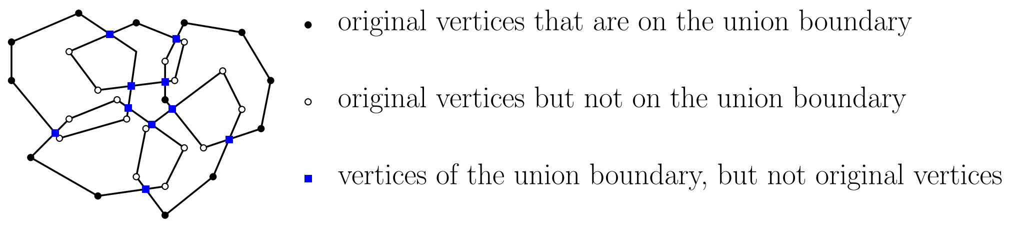

● original vertices that are on the union boundary
○ original vertices but not on the union boundary
■ vertices of the union boundary, but not original vertices

Figure 2: Problem 3. (Black circles indicate the vertices counted by m. The combination of black and white circles indicate the vertices counted by n. The combination of black circles and blue squares indicate the vertices counted in the union boundary.)

(b) Prove a lower bound of $2n - 6$ on the worst-case complexity of the number of vertices on the union boundary by constructing an example that has this complexity. Formally, if $f(n)$ denotes the maximum number of boundary vertices that can be realized by any collection of polygons having n total vertices, then you are to prove that $f(n) \geq 2n - 6$. (Your example should be sufficiently generic that it is clear that it applies for arbitrarily large values of n.)

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.

Consider a set $P = \{p_1, \ldots, p_n\}$ in $\mathbb{R}^2$. Let Δ denote the set of all inter-point distances from P, that is, $\Delta = \{\|p_i p_j\| \ : \ 1 \leq i < j \leq n\}$. Assuming general position, Δ contains $\binom{n}{2}$ values. Let $k = \lfloor \binom{n}{2}/2 \rfloor$, and let δ_{med} denote the kth smallest distance from Δ, that is, the median distance of the set.

Present an $O(n \log n)$ time algorithm which, given P and a candidate value δ^*, determines whether δ^* is (roughly speaking) larger or smaller than δ_{med}. More formally:

- if $\delta^* < \delta_{\text{med}}/2$, your algorithm must report "too small,"
- if $\delta^* > 2\delta_{\text{med}}$, your algorithm must report "too large,"

- and otherwise, your algorithm may report either response.

(Hint: Use WSPDs. The following may be helpful. Suppose that you are given a multiset of real numbers $X = \{x_1, \ldots, x_m\}$, and you are told that x_i appears n_i times in the set. Let $N = \sum_i n_i$ be the total number of elements, counting multiplicities. Then, in $O(m)$ time it is possible to compute the *weighted median* of X, which means that each element x_i is counted as if it appears n_i times. In general, you can compute the kth smallest element of X for any $1 \leq k \leq N$ in $O(m)$ time.)

Homework 5: WSPDs and VC-Dimension

Handed out Tuesday, Dec 9. Due anytime (up to 11:59pm) on Mon, Dec 15. You can turn your written homework into me or directly to Phil, or scan/typeset it and submit it by email (preferably directly to Phil). 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 in *general position.* Whenever asked to give an algorithm running in $O(f(n))$ time, you may give a *randomized algorithm* whose expected running time is $O(f(n))$.

Problem 1. Given a set of n points P in $\mathbb{R}^d$, and given any point $p \in P$, its *nearest neighbor* is the closest point to p among the remaining points of P. Note that nearest neighbors are not reflexive, in the sense that if p is the nearest neighbor of q, then q is not necessarily the nearest neighbor of p. Given an approximation factor $\varepsilon > 0$, we say that a point $p' \in P$ is an *ε-approximate nearest neighbor* to p if $\|pp'\| \leq (1+\varepsilon)\|pp''\|$, where p'' is the true nearest neighbor to p.

Show that in $O(n \log n + (1/\varepsilon)^d n)$ time it is possible to compute an ε-approximate nearest neighbor for every point of P. Justify the correctness of your algorithm. Hint: This can be solved using either WSPDs or spanners.)

Note: There exists an algorithm that runs in $O(n \log n)$ time that solves this problem exactly, but it is considerably more complicated than the one I have in mind here.

Problem 2. The object of this problem is to investigate the *VC-dimension* of some simple range spaces.

(a) Consider the range space of halfplanes, that is, the set of points lying on one side, either above or below, a line in $\mathbb{R}^2$ (see Fig. 1(a)). Derive the VC-dimension of this space and justify your answer.

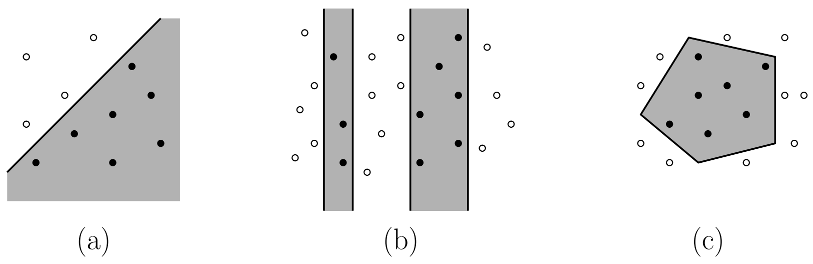


Figure 1: Problem 2.

(b) Consider the range space consisting of two disjoint vertical slabs in the plane (see Fig. 1(b)). Derive the VC-dimension of this space and justify your answer.

(c) Prove that the range space consisting of convex polygons in the plane has unbounded VC-dimension (see Fig. 1(c)). That is, show that for any $n > 0$, there exists an n-element point set that is shattered by the range space of convex polygons.

Sample Problems for the Midterm Exam

The following problems have been collected from old homeworks and exams. Because the material and order of coverage varies each semester, these problems do *not* necessarily reflect the actual length, coverage, or difficulty of the midterm exam.

The exam will be *closed-book* and *closed-notes*. You may use *one sheet of notes* (front and back). Unless otherwise stated, you may assume general position. If you are asked to present an $O(f(n))$ time algorithm, you may present a randomized algorithm whose expected running time is $O(f(n))$. For each algorithm you give, derive its running time and justify its correctness.

Problem 1. Give a short answer to each question (a few sentences suffice).

(a) You wish to use an orientation primitive to determine whether a point d lies within the interior of a triangle $\triangle abc$ in the plane. Your friend tells you that it suffices to test whether $\text{Orient}(a, b, d)$, $\text{Orient}(b, c, d)$ and $\text{Orient}(c, a, d)$ are all of the *same sign*. You realize that you don't know whether the vertices a, b, and c are given in clockwise or counterclockwise order. Is your friend's solution correct? Explain briefly.

(b) In the algorithm presented in class for decomposing a simple polygon into monotone pieces, what was the definition of $\text{helper}(e)$ and (in a few words) what role did it play in the algorithm?

(c) A convex polygon P_1 is enclosed within another convex polygon P_2. Suppose you dualize the vertices of each of these polygons (using the dual transform given in class, where the point (a, b) is mapped to the dual line $y = ax - b$). What can be said (if anything) about the relationships between the resulting two dual sets of lines.

(d) In the primal plane, there is a triangle whose vertices are the three points p, q, and r and there is a line ℓ that intersects this triangle. What can you infer about the relationship among the corresponding dual lines p^*, q^*, r^*, and the dual point ℓ^*? Explain.

(e) In the analysis of the randomized incremental point location we argued that the expected depth of a random query point is at most $12 \ln n$. Based on your knowledge of the proof, where does the factor 12 arise? (Hint: It arises from two factors. You can explain either for partial credit.)

(f) In a Voronoi diagram of a set of sites $P = \langle p_1, \ldots, p_n \rangle$ in the plane, you observe that the Voronoi edge between sites p_i and p_j is semi-infinite (that is, one end of the edge goes to infinity). What can be said about the relationship between these two sites and the rest of the point set? Explain.

(g) Any triangulation of any n-sided simple polygon has exactly $n - 2$ triangles. Suppose that the polygon has h polygonal holes each having k sides. (In Fig. 1, $n = 12$, $h = 3$, and $k = 4$). As a function of n, h and k, how many triangles will such a triangulation have? Explain briefly.

Problem 2. For this problem give an exact bound for full credit and an asymptotic (big-Oh) bound for partial credit. Assume general position.

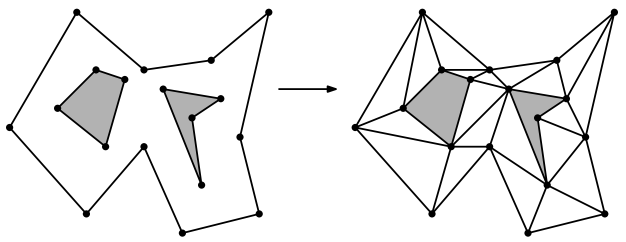


Figure 1: Problem 1(g).

(a) You are given a convex polygon P in the plane having n_P sides and an x-monotone polygonal chain Q having n_Q sides (see Fig. 2(a)). What is the maximum number of intersections that might occur between the edges of these two polygons?

(b) Same as (a), but P and Q are both polygonal chains that are monotone with respect to the x-axis (see Fig. 2(b)).

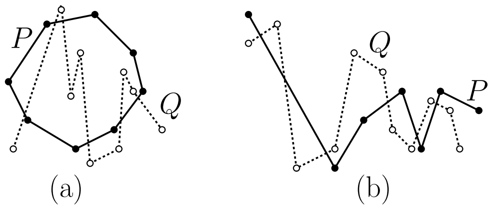


Figure 2: Problem 2.

(c) Same as (b), but P and Q are both monotone polygonal chains, but they may be monotone with respect to two different directions.

Problem 3. Consider the following randomized incremental algorithm, which computes the smallest rectangle (with sides parallel to the axes) bounding a set of points in the plane. This rectangle is represented by its lower-left point low and the upper-right point high.

```
(1) Let $P = \{p_1, p_2, \ldots, p_n\}$ be a random permutation of the points.
(2) Let low[$x$] = high[$x$] = $p_1[x]$. Let low[$y$] = high[$y$] = $p_1[y]$.
(3) For $i = 2$ through $n$ do:
    (a) if $p_i[x] <$ low[$x$] then ($*$) low[$x$] = $p_i[x]$.
    (b) if $p_i[y] <$ low[$y$] then ($*$) low[$y$] = $p_i[y]$.
    (c) if $p_i[x] >$ high[$x$] then ($*$) high[$x$] = $p_i[x]$.
    (d) if $p_i[y] >$ high[$y$] then ($*$) high[$y$] = $p_i[y]$.
```

Clearly this algorithm runs in $O(n)$ time. Prove that the total number of times that the "then" clauses of statements 3(a)–(d) (each indicated with a ($*$)) are executed is $O(\log n)$ on average. (We are averaging over all possible random permutations of the points.) To simplify your analysis you may assume that no two points have the same x- or y-coordinates.

Problem 4. You are given a set of n vertical line segments in the plane. Present an efficient an algorithm to determine whether there exists a line that intersects all of these segments. An example is shown in the figure below. (Hint: $O(n)$ time is possible.) Justify your algorithm's correctness and derive its running time.

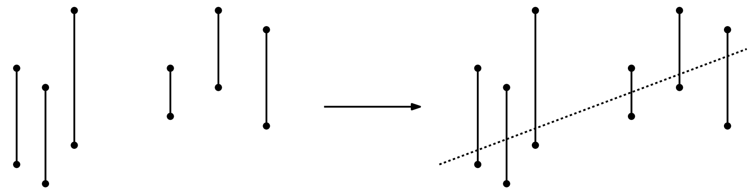


Figure 3: Problem 5.

Problem 5. You are given three convex polygons in the plane P, Q, and M. Let n denote the total number of vertices in all three polygons. Each is given as a sequence of vertices in counterclockwise order. Present an $O(n)$ time algorithm to determine whether there exists a line segment $\overline{pq}$ such that $p \in P$, $q \in Q$, and the midpoint of p and q lies within M (see Fig. 4).

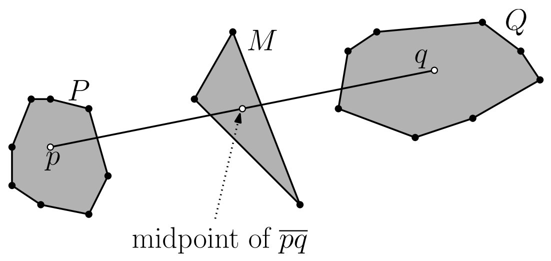


Figure 4: Problem 5.

Problem 6. A simple polygon P is *star-shaped* if there is a point q in the interior of P such that for each point p on the boundary of P, the open line segment $\overline{qp}$ lies entirely within the interior of P (see Fig. 5). Suppose that P is given as a counterclockwise sequence of its vertices $\langle v_1, v_2, \ldots, v_n \rangle$. Show that it is possible to determine whether P is star-shaped in $O(n)$ time. (Note: You are *not* given the point q.) Prove the correctness of your algorithm.

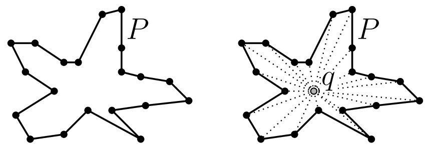


Figure 5: Problem 6.

Problem 7. You are given two sets of points in the plane, the red set R containing n_r points and the blue set B containing n_b points. The total number of points in both sets is $n = n_r + n_b$. Give an $O(n)$ time algorithm to determine whether the convex hull of the red set intersects the convex hull of the blue set. If one hull is nested within the other, then we consider them to intersect.

Problem 8. You are given a simple polygon P with the property that its leftmost and rightmost edges are vertical. Let $\langle u_1, \ldots, u_n \rangle$ denote the sequence of vertices on the polygonal chain joining the two upper endpoints of the leftmost and rightmost edges, and let $\langle v_1, \ldots, v_n \rangle$ denote the sequence of vertices of the polygonal chain joining the two lower endpoints (see Fig. 6).

Present an efficient algorithm to determine whether there exists a point p on the leftmost edge and a point q on the rightmost edge such that the line segment $\overline{pq}$ lies entirely within P. (The line segment $\overline{pq}$ is allowed to pass through vertices of P, but it cannot intersect the exterior of P.)

Derive your algorithm's running time and justify its correctness.

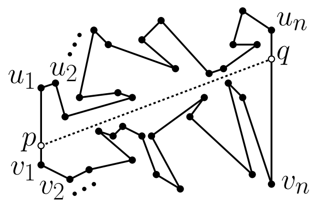


Figure 6: Problem 8.

Problem 9. Given a set of n points P in the plane, we define a subdivision of the plane into rectangular regions by the following rule. We assume that all the points are contained within a bounding rectangle. Imagine that the points are sorted in increasing order of y-coordinate. For each point in this order, shoot a bullet to the left, to the right and up until it hits an existing segment, and then add these three bullet-path segments to the subdivision (see Fig. 7(a)).

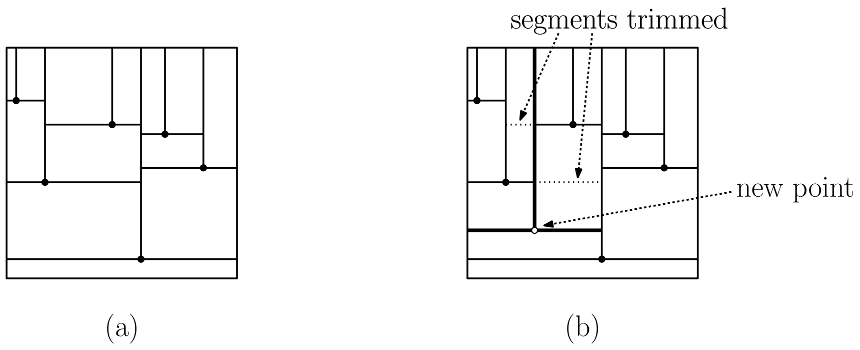


Figure 7: Problem 9.

(a) Show that the resulting subdivision has size $O(n)$ (including vertices, edges, and faces).

(b) Describe an algorithm to add a new point to the subdivision and restore the proper subdivision structure. Note that the new point may have an arbitrary y-coordinate, but the subdivision must be updated as if the points had been inserted in increasing order of y-coordinate (see Fig. 6(b)).

(c) Prove that if the points are added in random order, then the expected number of structural changes to the subdivision with each insertion is $O(1)$.

Problem 10. Given two points $p_1 = (x_1, y_1)$ and $p_2 = (x_2, y_2)$ in the plane, we say that p_2 *dominates* p_1 if $x_1 \leq x_2$ and $y_1 \leq y_2$. Given a set of points $P = \{p_1, p_2, \ldots, p_n\}$, a point p_i is said to be *maximal* if it is not dominated by any other point of P (shown as black points in Fig. 8(b)).

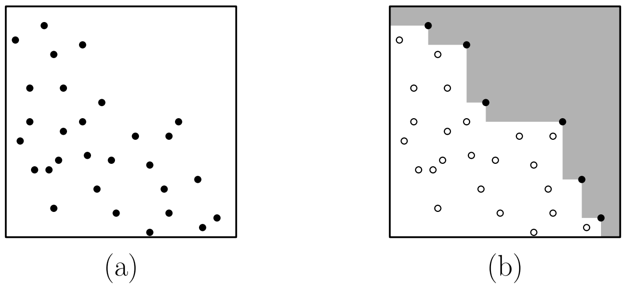


Figure 8: Problem 10.

Suppose further that the points of P have been generated by a random process, where the x-coordinate and y-coordinate of each point are independently generated random real numbers in the interval $[0, 1]$.

(a) Assume that the points of P are sorted in increasing order of their x-coordinates. As a function of n and i, what is the probability that p_i is maximal? (Hint: Consider the points p_j, where $j \geq i$.)

(b) Prove that the expected number of maximal points in P is $O(\log n)$.

Problem 11. Consider an n-sided simple polygon P in the plane. Let us suppose that the leftmost edge of P is vertical (see Fig. 9(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.)

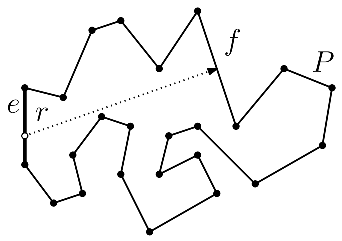


Figure 9: Problem 11.

Problem 12. You are given a set of n sites P in the plane. Each site of P is the center of a circular disk of radius 1. The points within each disk are said to be *safe*. We say that P is *safely connected* if, given any $p, q \in P$, it is possible to travel from p to q by a path that travels only in the safe region. (For example, the disks of Fig. 10(a) are connected, but the disks of Fig. 10(b) are not.)

Present an $O(n \log n)$ time algorithm to determine whether such a set of sites P is safely connected. Justify the correctness of your algorithm and derive its running time.

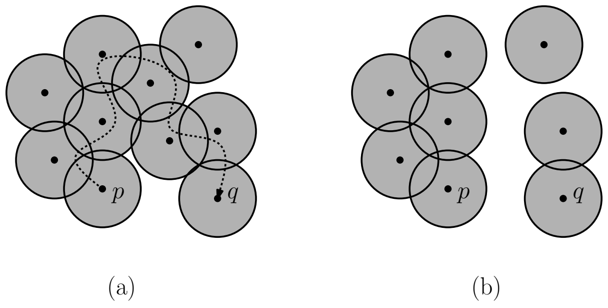


Figure 10: Problem 12.

Problem 13. 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 its the associated site. Reading the labels from left to right defines a string. (In Fig. 11 below the string would be $p_2p_1p_2p_5p_7p_9p_{10}$.)

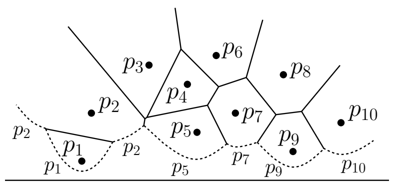


Figure 11: Problem 13.

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

$$\ldots p_i \ldots p_j \ldots p_i \ldots p_j \ldots$$

(b) Prove that any string of n distinct symbols that does not contain any repeated symbols $(\ldots p_i p_i \ldots)$ and does not contain the alternating sequence[1] of the type given in part (a) cannot be of length greater than $2n - 1$. (Hint: Use induction on n.)

[1]Sequences that contain no forbidden subsequence of alternating symbols are famous in combinatorics. They are known as *Davenport-Schinzel sequences*. They have numerous applications in computational geometry, this being one.

Problem 14. Given a set P of n points in the plane and an integer k ($1 \leq k \leq n-1$), a *k-set* is defined to be a subset of $P' \subseteq P$, where $|P'| = k$ and $P' = P \cap h$ for some halfplane h. That is, a k-set is any set of k points that can be separated from the rest of P by a single line. (In Fig. 12(a) we show two examples of 3-sets, $\{a, b, c\}$ and $\{a, d, h\}$.)

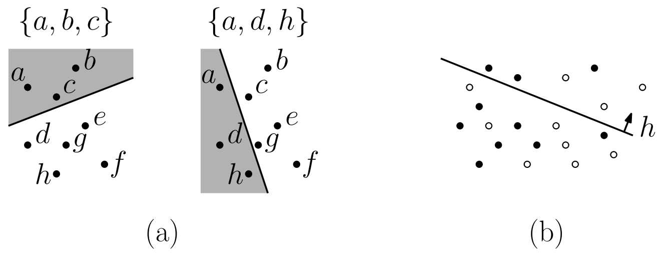


Figure 12: Problem 14.

(a) Given what you know about point-line duality, explain how a k-set manifests itself in the dual plane, in terms of the arrangement of the dual lines P^*.

(b) Present an $O(n^2 \log n)$ time and $O(n)$ space algorithm which, given two planar point sets P and Q, each of size n, and an integer k, determines whether there is there a halfspace h, such that $|P \cap h| = |Q \cap h| = k$. (See Fig. 12(b) for the special case $k = 3$.)

CMSC 754: Midterm Exam

This exam is closed-book and closed-notes. You may use one sheet of notes, front and back. Write all answers in the exam booklet. If you have a question, either raise your hand or come to the front of class. Total point value is 100 points. Good luck!

In all problems, unless otherwise stated, you may assume that points are in *general position*. You may make use of any results presented in class and any well known facts from algorithms or data structures. If you are asked for an $O(T(n))$ time algorithm, you may give a *randomized* algorithm with *expected* time $O(T(n))$.

Problem 1. (40 points; 5–8 points each) Give a short answer (a few sentences) to each question.

(a) Recall that in a simple polygon, a *scan-reflex vertex* is a vertex having both incident edges on the same side of a vertical line passing through the vertex. Given a simple polygon P with n vertices and r scan-reflex vertices, what is the maximum and minimum number of diagonals that might be needed to decompose it into monotone pieces? Explain briefly.

(b) In the plane-sweep algorithm for computing line-segment intersections, we were careful to *remove* intersection events from the priority queue whenever the two lines defining such an event were no long consecutive along the sweep line. Why did we bother to do this?

(c) What are the three possible outcomes of an instance of linear programming? (No explanation needed.)

(d) Fortune's sweep-line algorithm for computing Voronoi diagrams involves two principal types of events, *site events* and *Voronoi vertex events* (which our text called *circle events*). Briefly explain the circumstances under which each event arises. (You *do not* need to explain what the algorithm does in each case.)

(e) What does it mean to say that the Delaunay triangulation of a set of points in the plane is a 2.418-spanner?

(f) You are given a procedure for computing the convex hull of a set of points in $\mathbb{R}^3$. Explain how to use this procedure (as a black box) to compute the Delaunay triangulation of a set of points P in $\mathbb{R}^2$. (No need to prove correctness.)

Problem 2. (10 points) Consider two (nonvertical) lines ℓ_1 and ℓ_2, where ℓ_1 has a strictly smaller slope that ℓ_2. Define the *double wedge*, denoted $X(\ell_1, \ell_2)$ to be the "scissor-like" shaped region of the plane lying between these lines. Define the *size* of the wedge to be the difference in the slopes of ℓ_1 and ℓ_2. We say that a double wedge *separates* two sets P and Q if they are contained within the wedge, but lie on opposite sides (see Fig. 1(b)).

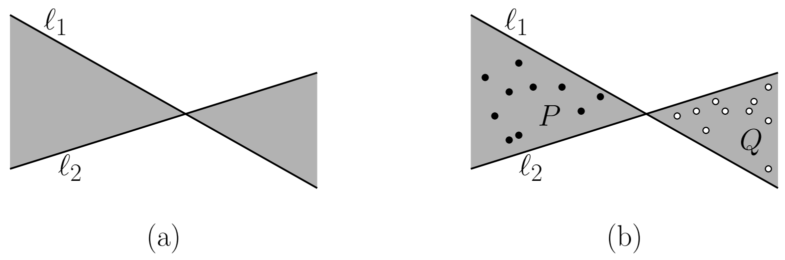
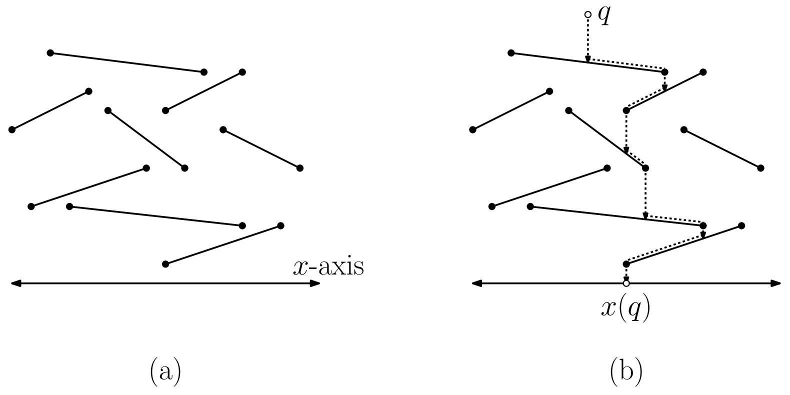


Figure 1: Problem 2.

Let P and Q be two n-element point sets that are separated by a vertical line, with P to the left and Q to the right. Present an efficient algorithm that computes the double wedge of minimum size that separates these two sets. Briefly justify your algorithm's correctness and derive its running time. (The amount of credit you receive will depend on how efficient your algorithm is.)

Problem 3. (15 points) You are given a set $S = \{s_1, \ldots, s_n\}$ of n non-intersecting line segments in the plane all lying above the x-axis, such that no line segment is horizontal or vertical (see Fig. (a)). Explain how to construct an efficient data structure for answering the following queries. Given any point $q \in \mathbb{R}^2$ that lies above the x axis, suppose that a drop of water falls at q and drips from one segment to the next until reaching the x-axis. The answer to the query is the x-coordinate where this drop finally lands (see Fig. (b)). Show that your data structure uses space $O(n)$, answers queries in time $(\log n)$, and (very important) it can be constructed in time $O(n \log n)$.



Problem 4. (20 points) You are given two n-element point sets P and Q in the plane that are separated by the y-axis, with P to the left and Q to the right (see Fig. 2(a)). We are interested in finding a line ℓ with the property that there are two points of P lying on or below ℓ and two points of Q lying on or below ℓ (see Fig. 2(b)).

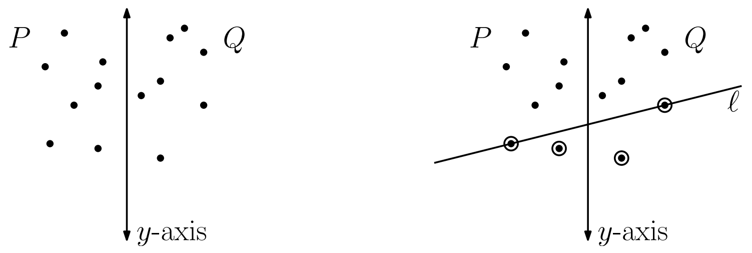


Figure 2: Problem 4.

(a) Given what you know about point-line duality, explain what properties the dual of ℓ possesses relative to duals of P and Q. What is the effect of P and Q lying on opposite of the y-axis?

(b) If ℓ is constrained to pass through a point of P and a point of Q, prove that ℓ is unique. (Hint: If you do a good job on part (a), this should be easy.)

(c) Present an $O(n^2)$ time algorithm to find the line ℓ from part (b). (There are faster algorithms, but $O(n^2)$ suffices for full credit.)

Problem 5. (15 points) Here is a twist on the frog problem. A frog who *loves the water* wants to cross a stream that is defined by two horizontal lines $y = y^-$ and $y = y^+$. On the surface there are n circular lily pads whose centers are at the points $P = \{p_1, \ldots, p_n\}$. The frog crosses the stream by *swimming between* the circular lily pads. To prevent the frog from going entirely around the lily pads, we require that the frog remain between the vertical lines $x = x^-$ and $x = x^+$. The lily pads grow at the same rate, so that at time t they all have radius t (see Fig. 3(a)).

Help the frog out by presenting an algorithm that given P and a time value t, determines whether there exists a feasible path for the frog at time t. Your algorithm should run in $O(n \log n)$ time. (Note that the value of t is given to you! You don't need to compute it.)

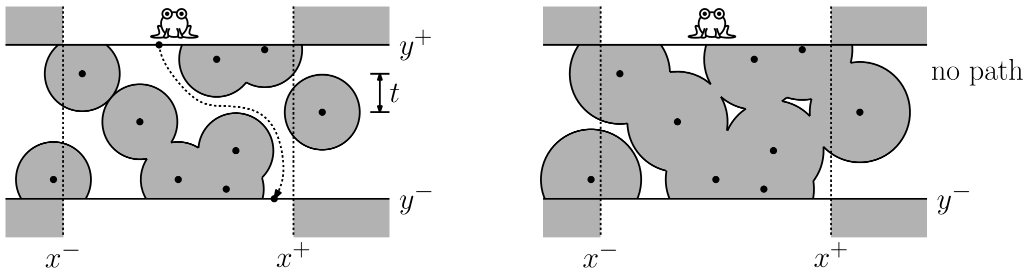


Figure 3: Problem 5.

(Hint: Show that if a feasible path exists for the frog, then such a path may be assumed to travel along the edges of an appropriately defined graph having $O(n)$ edges.)

Sample Problems for the Final Exam

The final exam will be **Fri, Dec 19, 1:30-3:30pm**. The exam will be closed-book and closed-notes. You may use *two* sheets of notes (front and back). The following problems have been collected from old homeworks and exams. They do not necessarily reflect the actual difficulty or coverage of questions on the final exam. The final will be comprehensive, but will emphasize material since the midterm.

In all problems, unless otherwise stated, you may assume general position, and you may use of any results presented in class or any well-known result from algorithms and data structures.

Problem 1. Give a short answer (a few sentences) to each question. Unless explicitly requested, explanations are not required, but may be given for partial credit.

(a) A *dodecahedron* is a convex polyhedron that has 12 faces, each of which is a 5-sided pentagon. How many vertices and edges does the dodecahedron have? Show how you derived your answer.

(b) Consider a set of n nonintersecting line segments in the plane. If these segments are inserted in the worst possible order (not randomly), what is the worst-case space complexity of the resulting trapezoidal map? What is the worst-case depth of the resulting point location data structure? (Express your answers asymptotically as a function of n.)

(c) True or false: Given a planar point set P, if p and q are the closest points of P then $\overline{pq}$ is an edge of P's Delaunay triangulation.

(d) When the ith site is added to the Delaunay triangulation using the randomized incremental algorithm, what is the worst-case number of edges that can be incident on the newly added site? What can you say about the expected-case number of such edges (assuming that points are inserted in random order)?

(e) What is the (asymptotic) maximum number of faces of a convex hull of n points in $\mathbb{R}^d$?

(f) Given an arrangement $\mathcal{A}$ of a set of n lines in the plane and given an arbitrary line ℓ, what is the (asymptotic) maximum complexity (number of edges) of the zone of ℓ relative to $\mathcal{A}$?

(g) Suppose that you are given a set of nonintersecting polygonal obstacles in $\mathbb{R}^2$, and a start point s and a destination point t. The *visibility graph* consists of all pairs of obstacle vertices (together with s and t) such that the segment between these points does not intersect the interior of any obstacle. True or false: The shortest obstacle-avoiding path from s to t travels entirely along edges of the visibility graph. Explain briefly.

(h) Generalizing (e), we can define the visibility graph on the set of vertices of a set of nonintersecting polyhedra in $\mathbb{R}^3$. True or false: The shortest obstacle-avoiding path from s to t in 3-space travels along edges of the visibility graph. Explain briefly.

(i) Let P and Q be two simple polygons in $\mathbb{R}^2$, where P has m vertices and Q has n vertices. What is the maximum number of vertices on the boundary of the Minkowski sum $P \oplus Q$ (asymptotically) assuming:

(i) P and Q are both convex

(ii) P is convex but Q is arbitrary

(iii) P and Q are both arbitrary

Problem 2. You are given a set P of n points in the plane and a path π that visits each point exactly once. (This path may self-intersect. See Fig. 1.)

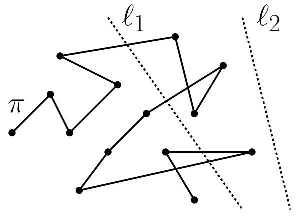


Figure 1: Problem 2.

(a) Explain how to build a data structure from P and π of space $O(n)$ so that given any query line ℓ, it is possible to determine in $O(\log n)$ time whether ℓ intersects the path. (For example, in Fig. 1 the answer for ℓ_1 is "yes," and the answer for ℓ_2 is "no.") (Hint: Use duality, but be clever in how you dualize the path. There is an easy way to do this and a hard way.)

(b) This is a generalization of part (a). Explain how to build a data structure from P and π so that given any line ℓ, it is possible to report all the segments of π that intersect ℓ. The space should be at most $O(n \log n)$ and the query time should be at most $O(k \log^2 n)$, where k is the number of segments reported. (Hint: Use divide-and-conquer, applying the data structure from part (a).)

Problem 3. Consider the following two geometric graphs defined on a set P of points in the plane.

(a) *Box Graph:* Given two points $p, q \in P$, define box(p, q) to be the square centered at the midpoint of $\overline{pq}$ having two sides parallel to the segment $\overline{pq}$ (see Fig. 2(a)). The edge (p, q) is in the box graph if and only if box(p, q) contains no other point of P (see Fig. 2(b)). Show that the box graph is a subgraph of the Delaunay triangulation of P.

(b) *Diamond Graph:* Given two points $p, q \in P$, define diamond(p, q) to be the square having $\overline{pq}$ as a diagonal (see Fig. 2(c)). The edge (p, q) is in the diamond graph if and only if diamond(p, q) contains no other point of P (see Fig. 2(d)). Show that the diamond graph may not be a subgraph of the Delaunay triangulation of P. (Hint: Give an example that shows that the diamond graph is not even planar.)

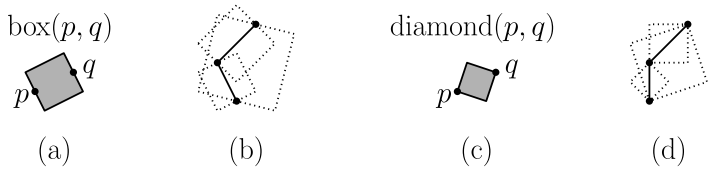


Figure 2: Problem 3: The box and diamond graphs.

Problem 4. You are given a set $P = \{p_1, \ldots, p_n\}$ of n points in the plane. Consider all the $\binom{n}{2}$ lines passing through each pair of distinct points $p_i, p_j \in P$, and let $\ell_{\max}$ be the line of this set with the maximum slope. We are interested in computing $\ell_{\max}$.

(a) What is the interpretation of $\ell_{\max}$ in the dual plane?

(b) Give an $O(n \log n)$ algorithm that computes $\ell_{\max}$. Justify your algorithm's correctness. (Hint: This can be done either in the dual or the primal space.)

Problem 5. The objective of this problem is to compute the discrepancy of a set of points in the plane with respect to axis-parallel rectangles. Let P denote a set of n points in the unit hypercube $U = [0,1]^2$. Given any axis-parallel rectangle R define $\mu(R)$ to be the area of $R \cap U$ and define $\mu_P(R) = |P \cap R|/|P|$ to be the fraction of points of P lying within R (see Fig. 3). Define the discrepancy of P with respect to R to be $\Delta_P(R) = |\mu(R) - \mu_P(R)|$, and define the *rectangle discrepancy* of P, denoted $\Delta(P)$ to be the maximum (or more accurately, the supremum) of $\Delta_P(R)$ over all axis-parallel rectangles R in U.

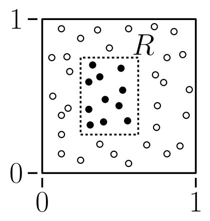


Figure 3: Problem 5: Rectangle discrepancy.

Present an $O(n^4)$ time and $O(n^2)$ space algorithm for computing rectangle discrepancy of P by answering the following parts. Throughout you may assume that the points of P are in general position, but the axis-parallel rectangles that are used in the computation of the discrepancy are arbitrary (that is, the corner points may be any two points in U).

(a) Establish a *finiteness criterion* for this problem by showing that there exists a set of at most $O(n^4)$ rectangles such that $\Delta(P)$ is given by one of these rectangles. Call these the *canonical rectangles* for P.

(b) Develop a rectangle range counting data structure of size $O(n^2)$ that can be used to compute the number of points of P lying within any canonical rectangle in $O(1)$ time. Your procedure should allow the option of either including or excluding points on the boundary of the rectangle. (Hint: The solution involves a grid, but of rectangles, not squares.)

(c) Using your solution to (b), show how to compute the discrepancy for P in $O(n^4)$ time and $O(n^2)$ space.

Problem 6. Consider the range space (P, R) where P is a set of n points in the plane, and R is the set of all ranges arising by intersecting P with a closed halfplane.

(a) Show that the VC-dimension of halfplane ranges is at least three by giving an example of a set of three points in the plane that are shattered by the set of halfplane ranges.

(b) Show that the VC-dimension of halfplane ranges is at most three, by proving that no four-element set can be shattered by halfplane ranges.

Problem 7. You are given a set P of n points in $\mathbb{R}^d$ and an approximation factor $\varepsilon > 0$. An (exact) *distance query* is defined as follows. You are given a real $\delta > 0$, and you are to return a count of all the pairs of points $(p,q) \in P \times P$, such that $\|pq\| \geq \delta$. In an *ε-approximate distance query*, your count *must* include all pairs (p,q) where $\|pq\| \geq \delta(1+\varepsilon)$ and it *must not* include any pairs (p,q) where $\|pq\| < \delta/(1+\varepsilon)$. Pairs of points whose distances lie between these two bounds may or may not be counted, at the discretion of the algorithm.

Explain how to preprocess P into a data structure so that ε-approximate distance counting queries can be answered in $O(n/\varepsilon^d)$ time and $O(n/\varepsilon^d)$ space. (Hint: Use a well-separated pair decomposition. Explain clearly what separation factor is used and any needed modification to the WSPD construction.)

Problem 8. A desirable property of spanners is that every vertex should have constant degree.

(a) Show that there exists a set of points in the plane such that the WSPD-based spanner construction given in class results in at least one vertex of degree $\Omega(n)$. That is, the degree is at least cn for some $c > 0$, which might depend on the dimension and separation factor, but not on n. (Note: The algorithm for constructing the quadtree did not specify how representatives are chosen in the quadtree. To obtain the worst case, you will need to select representatives carefully.)

(b) Show that it is possible to modify the WSPD-based spanner construction so that the number of edges is the same, but each vertex has constant degree (where the constant depends on the stretch factor). For this, you may make the simplifying assumption that the quadtree is not compressed and that every internal node has at least two nonempty children. (Hint: Show that in such a tree it is possible to distribute representatives so that each point occurs as the representative for at most a constant number nodes of the tree.)

Problem 9. The purpose of this problem is to consider an approximation algorithm to an important problem that arises in clustering. You are given n points P in the plane. Given any integer k, $1 \leq k \leq n$, define $b_k(P)$ to be the Euclidean ball of minimum radius that encloses at least k points of P (see Fig. 4). (Note that the center of $b_k(P)$ may be anywhere in $\mathbb{R}^2$, not necessarily at a point of P). Let $r_k(P)$ denote the radius of this ball. The objective of this problem is to derive an algorithm that computes a factor-2 approximation, that is, it computes a ball of radius at most $2r_k(P)$ that contains at least k points of P.

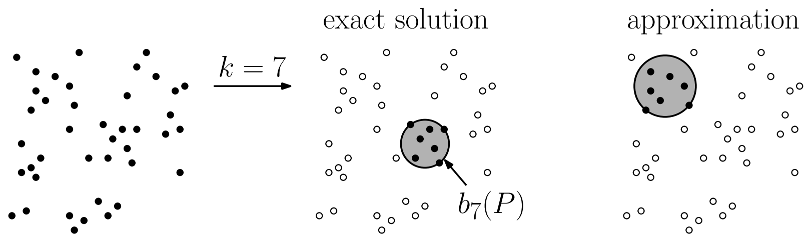


Figure 4: Problem 9: Smallest k-disk approximation.

The approximation algorithm is based on computing a small number of *candidate centers* $Q = \{q_1, \ldots, q_m\}$, such that at least one of these candidate centers is guaranteed to lie within $b_k(P)$. For each candidate center q_i, we will determine the radius of the smallest ball centered at q_i that contains k points, and return the smallest such ball.

The candidate centers are constructed as follows. First, sort the points by their x-coordinates, letting $x_1 \leq \ldots \leq x_n$ be the resulting set. Let $k' = \lfloor (k-1)/2 \rfloor$. Let X be the set that results by taking every k'-th point in the sequence, that is $X = \{x_{k'}, x_{2k'}, \ldots, x_{zk'}\}$, where $z = \lfloor n/k' \rfloor$. Do the same for the y-coordinates by letting Y denote the result of taking every k'-th point of the y-coordinates in sorted order. Finally, let $Q = X \times Y$, be the set of points that result by taking any x-coordinate from X and any y-coordinate from Y.

(a) Prove that there exists a point $q \in Q$ such that $q \in b_k(P)$. (Hint: Show that Q is a $(2k'/n)$-net for P in the sense that *any* ball that fails to contain a point of Q can contain at most $2k'$ points of P.)

(b) For each $q_i \in Q$, let r_i denote the radius of the smallest ball centered at q_i that contains at least k points of P. Let $r_{\min} = \min_i r_i$ and let $b_{\min}$ denote the associated ball. Prove that $r_{\min} \leq 2r_k(P)$. (This establishes the fact that $b_{\min}$ is a factor-2 approximation to $b_k(P)$.)

(c) Show that the algorithm's overall running time is $O(n \log n + n^3/k^2)$. For what choices of k is the running time $O(n \log n)$?

CMSC 754: Final Exam

This exam is closed-book and closed-notes. You may use two sheets of notes, front and back. Write all answers in the exam booklet. If you have a question, either raise your hand or come to the front of class. Total point value is 100 points. Good luck!

In all problems, unless otherwise stated, you may assume that points are in *general position*. You may make use of any results presented in class and any well known facts from algorithms or data structures. If you are asked for an $O(T(n))$ time algorithm, you may give a *randomized* algorithm with *expected* time $O(T(n))$.

Problem 1. (30 points; 5–10 points each) Give a short answer (a few sentences at most) to each question. Except where requested, explanations are not required.

(a) For each of the following assertions about the Delaunay triangulation of a set P of n points in the plane, which are True and which are False?

(i) The Delaunay triangulation is a t-spanner, for some constant t
(ii) The Euclidean minimum spanning tree of P is a subgraph of the Delaunay triangulation
(iii) Among all triangulations of P, the Delaunay triangulation maximizes the minimum angle
(iv) Among all triangulations of P, the Delaunay triangulation minimizes the maximum angle
(v) Among all triangulations of P, the Delaunay triangulation minimizes the total sum of edge lengths

(b) An arrangement of n lines in the plane has exactly n^2 edges. How many edges are there in an arrangement of n planes in 3-dimensional space? (Give an exact answer for full credit or an asymptotically tight answer for half credit.) Explain briefly.

(c) In each of the following cases, what is the asymptotic worst-case complexity (number of vertices) on the boundary of the union of n of the following objects in $\mathbb{R}^2$:

(i) axis-parallel squares
(ii) axis-parallel rectangles (of arbitrary heights and widths)
(iii) rectangles (of arbitrary heights and widths which need not be axis parallel)
(iv) axis-parallel rectangles, where the width to height ratio is either 4×1 or 1×4.

(d) In the randomized incremental algorithm for computing the trapezoidal map and point-location data structure, what were the source(s) of the $O(\log n)$ factor in the $O(n \log n)$ running time? (List all that apply.)

(i) The expected time to locate the trapezoid containing a segment's endpoint is $\Theta(\log n)$
(ii) The expected number of updates to the map with each insertion is $\Theta(\log n)$
(iii) The expected number of new nodes added to the history DAG with each insertion is $\Theta(\log n)$
(iv) Computing the initial random permutation of the segments takes time $\Theta(n \log n)$.

Problem 2. (20 points) You are given a set P of n points in $\mathbb{R}^2$. A nonvertical line ℓ partitions P into two (possibly empty) subsets: $P^+(\ell)$ consists of the points lie on or above ℓ and $P^-(\ell)$ consists of the points of P that lie strictly below ℓ (see Fig. 1(a)).

Given the point set P, present data structures for answering the following two queries. In each case, the data structure should use $O(n^2)$ space, it should answer queries in $O(\log n)$ time. (You do not need to explain how to build the data structure, but it should be constructable in polynomial time in n.)

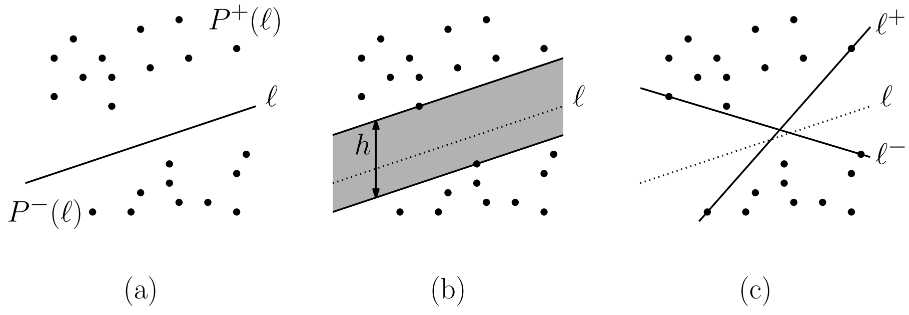


Figure 1: Problem 2: Query problem.

(a) The input to the query is a nonvertical line ℓ. The answer is the maximum vertical distance h between two lines parallel to h that lie between $P^+(\ell)$ and $P^-(\ell)$ (see Fig. 1(b)). For simplicity, you may assume that neither set is empty (implying that h is finite).

(b) Again, the input to the query is a nonvertical line ℓ. The answer to the query are the two lines ℓ^- and ℓ^+ of minimum and maximum slope, respectively, that separate $P^+(\ell)$ from $P^-(\ell)$ (see Fig. 1(c)). You may assume that $P^+(\ell)$ from $P^-(\ell)$ are *not* separable by a vertical line (implying that these two slopes are finite).

Problem 3. (20 points) In this problem we will consider the VC-dimension of two simple range spaces. Define a *quad* to be a four-sided polygon that is bounded to the left and right by vertical sides, on the bottom by a horizontal side, and the slope of the top side is arbitrary (see Fig. 2(a)). Define a *restricted quad* to be a quad whose left side is at $x = 0$, whose right side is at $x = 1$, and whose bottom side is at $y = 0$ (see Fig. 2(b)).

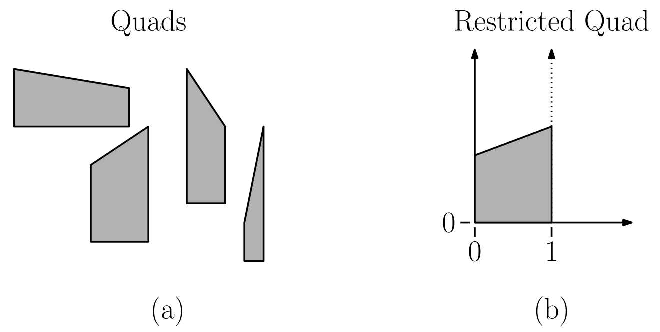


Figure 2: Problem 3: Quads and restricted quads.

(a) Prove that the VC-dimension of *restricted quads* is at least 2 by showing that there exists a 2-element point set in $\mathbb{R}^2$ that is shattered by the set of restricted quads.

(b) Prove that the VC-dimension of *restricted quads* is at most 2 by showing that no point 3-element point set in $\mathbb{R}^2$ is shattered by the set of restricted quads. (Hint: Label the three points p_1, p_2, and p_3 from left to right. There are two cases depending on whether p_2 lies above or below the segment $\overline{p_1 p_3}$.)

(c) Prove that the VC-dimension of (general) *quads* is at least 5 by showing that there exists a 5-element point set in $\mathbb{R}^2$ that is shattered by the set of quads.

(d) Prove that the VC-dimension of (general) *quads* is at most 5 by showing that no point 6-element point set in $\mathbb{R}^2$ is shattered by the set of restricted quads. (Hint: A careful proof with full details will take too long. It suffices to briefly explain how to generalize your answer to part (b).)

Problem 4. (15 points) You are given two sets P_r and P_b, each consisting of n points in $\mathbb{R}^d$. These are called the *red points* and *blue points*, respectively. The *closest red-blue pair* consists of the points $p \in P_r$ and $q \in P_b$ that minimize the Euclidean distance $\|pq\|$ over all red-blue pairs (see Fig. 3(a)).

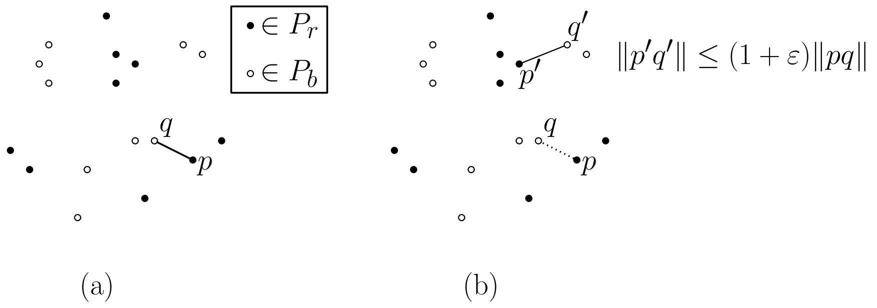


Figure 3: Problem 4: Approximate closest red-blue pair.

Given an approximation factor $\varepsilon > 0$, present an algorithm that computes an ε-approximation to the closest red-blue pair, by which we mean that it returns a pair of points $p' \in P_r$ and $q' \in P_b$ such that

$$\|p'q'\| \;\leq\; (1+\varepsilon)\|pq\|.$$

(see Fig. 3(b)). Your algorithm should run in time $O(n \log n + n/\varepsilon^d)$. Be sure to justify the correctness of your algorithm.

Problem 5. (15 points) This is a modification of the motion-planning problem from Homework 4 where:

(a) Robot and missiles are rectangles rather than line segments.

(b) Each missile has its own size and (fixed) velocity,

(c) The robot can move only monotonically from left to right.

You are given a *robot* that consists of a rectangle of width Δ_x and height Δ_y that is placed with its bottom edge on the x-axis (see Fig. 4(a)). The robot's reference point is the lower right corner of this rectangle, which at time $t = 0$ is placed at the origin. The robot can move to the right or stand still (but never up, down, or to the left) at a maximum speed of one unit per second.

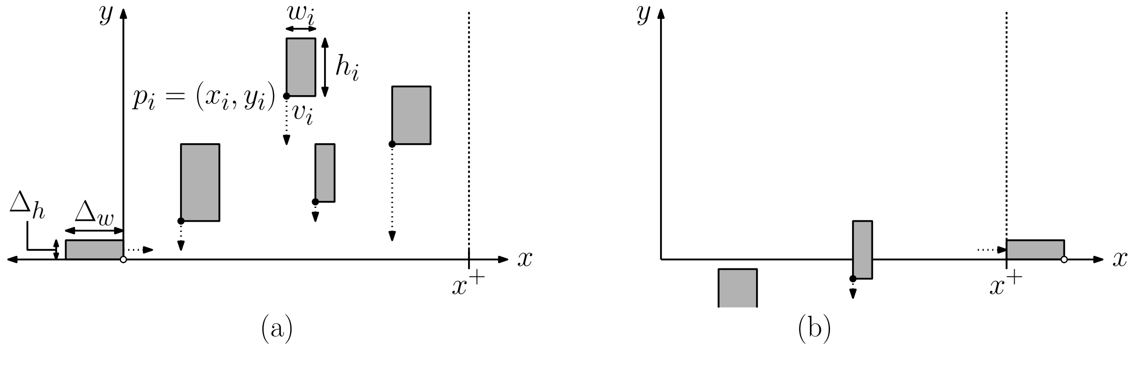


Figure 4: Problem 5: Motion Planning.

You are also given a set of *missiles* in the form of n rectangles, where the ith missile is of height h_i and width w_i. The reference point of each missile is its lower left corner, which at time $t = 0$ starts at position $p_i = (x_i, y_i)$. This missile moves vertically downward at a constant speed of v_i units per second. The objective is to compute a path for the robot that avoids all the missiles and ends entirely to the right of the vertical line $x = x^+$ (see Fig. 4(b)).

(a) The robot's configuration at any time is given as a pair (t, x), where t is the current time, and x is the location of the robot's reference point. The C-obstacle associated with the ith missile is defined as the set of robot configurations (t, x) such that the robot placed with its reference point at x at time t intersects the ith missile. Give a detailed description of this C-obstacle.

(b) Provide a complete characterization of the properties of a path in configuration space (assuming it exists) that corresponds to a collision-free motion plan for the robot that satisfies all the above constraints. (You do *not* need to show how to compute this path.)