CMSC 754: Short Reference Guide

This document contains a short summary of information about algorithm analysis and data structures, which may be useful later in the semester.

Asymptotic Forms: The following gives both the formal "c and n_0" definitions and an equivalent limit definition for the standard asymptotic forms. Assume that f and g are nonnegative functions.

Asymptotic Form	Relationship	Limit Form	Formal Definition
$f(n) \in \Theta(g(n))$	$f(n) \equiv g(n)$	$0 < \lim_{n\to\infty} \frac{f(n)}{g(n)} < \infty$	$\exists c_1, c_2, n_0, \forall n \geq n_0,\ 0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n).$
$f(n) \in O(g(n))$	$f(n) \preceq g(n)$	$\lim_{n\to\infty} \frac{f(n)}{g(n)} < \infty$	$\exists c, n_0, \forall n \geq n_0,\ 0 \leq f(n) \leq cg(n).$
$f(n) \in \Omega(g(n))$	$f(n) \succeq g(n)$	$\lim_{n\to\infty} \frac{f(n)}{g(n)} > 0$	$\exists c, n_0, \forall n \geq n_0,\ 0 \leq cg(n) \leq f(n).$
$f(n) \in o(g(n))$	$f(n) \prec g(n)$	$\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$	$\forall c, \exists n_0, \forall n \geq n_0,\ 0 \leq f(n) \leq cg(n).$
$f(n) \in \omega(g(n))$	$f(n) \succ g(n)$	$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty$	$\forall c, \exists n_0, \forall n \geq n_0,\ 0 \leq cg(n) \leq f(n).$

Polylog-Polynomial-Exponential: For any constants a, b, and c, where $b > 0$ and $c > 1$.

$$\log^a n \prec n^b \prec c^n.$$

Common Summations: Let c be any constant, $c \neq 1$, and $n \geq 0$.

Name of Series	Formula	Closed-Form Solution	Asymptotic
Constant Series	$\sum_{i=a}^{b} 1$	$= \max(b-a+1, 0)$	$\Theta(b-a)$
Arithmetic Series	$\sum_{i=0}^{n} i = 0+1+2+\cdots+n$	$= \frac{n(n+1)}{2}$	$\Theta(n^2)$
Geometric Series	$\sum_{i=0}^{n} c^i = 1+c+c^2+\cdots+c^n$	$= \frac{c^{n+1}-1}{c-1}$	$\begin{cases} \Theta(c^n)\ (c>1) \\ \Theta(1)\ (c<1) \end{cases}$
Quadratic Series	$\sum_{i=0}^{n} i^2 = 1^2+2^2+\cdots+n^2$	$= \frac{2n^3+3n^2+n}{6}$	$\Theta(n^3)$
Linear-geom. Series	$\sum_{i=0}^{n-1} ic^i = c+2c^2+3c^3\cdots+nc^n$	$= \frac{(n-1)c^{(n+1)}-nc^n+c}{(c-1)^2}$	$\Theta(nc^n)$
Harmonic Series	$\sum_{i=1}^{n} \frac{1}{i} = 1+\frac{1}{2}+\frac{1}{3}+\cdots+\frac{1}{n}$	$\approx \ln n$	$\Theta(\log n)$

Recurrences: Recursive algorithms (especially those based on divide-and-conquer) can often be analyzed using the so-called *Master Theorem*, which states that given constants $a > 0$, $b > 1$, and $d \geq 0$, the function $T(n) = aT(n/b) + O(n^d)$, has the following asymptotic form:

$$T(n) = \begin{cases} O(n^d) & \text{if } d > \log_b a \\ O(n^d \log n) & \text{if } d = \log_b a \\ O(n^{\log_b a}) & \text{if } d < \log_b a. \end{cases}$$

Sorting: The following algorithms sort a set of n keys over a totally ordered domain. Let $[m]$ denote the set $\{0, \ldots, m\}$, and let $[m]^k$ denote the set of ordered k-tuples, where each element is taken from $[m]$. A sorting algorithm is *stable* if it preserves the relative order of equal elements. A sorting algorithm is *in-place* if it uses no additional array storage other than the input array (although $O(\log n)$ additional space is allowed for the recursion stack). The *comparison-based algorithms* (Insertion-, Merge-, Heap-, and QuickSort) operate under the general assumption that there is a *comparator function* $f(x, y)$ that takes two elements x and y and determines whether $x < y$, $x = y$, or $x > y$.

Algorithm	Domain	Time	Space	Stable	In-place
CountingSort	Integers $[m]$	$O(n+m)$	$O(n+m)$	Yes	No
RadixSort	Integers $[m]^k$ or $[m^k]$	$O(k(n+m))$	$O(kn+m)$	Yes	No
InsertionSort	Total order	$O(n^2)$	$O(n)$	Yes	Yes
MergeSort	Total order	$O(n \log n)$	$O(n)$	Yes	No
HeapSort				No	Yes
QuickSort				Yes/No*	No/Yes

*There are two versions of QuickSort, one which is stable but not in-place, and one which is in-place but not stable.

Order statistics: For any k, $1 \leq k \leq n$, the kth smallest element of a set of size n (over a totally ordered domain) can be computed in $O(n)$ time.

Useful Data Structures: All these data structures use $O(n)$ space to store n objects.

Unordered Dictionary: (by randomized hashing) Insert, delete, and find in $O(1)$ expected time each. (Note that you can find an element exactly, but you cannot quickly find its predecessor or successor.)

Ordered Dictionary: (by balanced binary trees or skiplists) Insert, delete, find, predecessor, successor, merge, split in $O(\log n)$ time each. (Merge means combining the contents of two dictionaries, where the elements of one dictionary are all smaller than the elements of the other. Split means splitting a dictionary into two about a given value x, where one dictionary contains all the items less than or equal to x and the other contains the items greater than x.) Given the location of an item x in the data structure, it is possible to locate a given element y in time $O(\log k)$, where k is the number of elements between x and y (inclusive).

Priority Queues: (by binary heaps) Insert, delete, extract-min, union, decrease-key, increase-key in $O(\log n)$ time. Find-min in $O(1)$ time each. Make-heap from n keys in $O(n)$ time.

Priority Queues: (by Fibonacci heaps) Any sequence of n insert, extract-min, union, decrease-key can be done in $O(1)$ amortized time each. (That is, the sequence takes $O(n)$ total time.) Extract-min and delete take $O(\log n)$ amortized time. Make-heap from n keys in $O(n)$ time.

Disjoint Set Union-Find: (by inverted trees with path compression) Union of two disjoint sets and find the set containing an element in $O(\log n)$ time each. A sequence of m operations can be done in $O(\alpha(m, n))$ amortized time. That is, the entire sequence can be done in $O(m \cdot \alpha(m, n))$ time. (α is the *extremely* slow growing inverse-Ackerman function.)

Orientation Testing: For any constant dimension d, given any ordered $(d + 1)$-tuple of points in $\mathbb{R}^d$, it can be determined in $O(1)$ time whether these points are (a) negatively oriented (clockwise), (b) positively oriented (counterclockwise) or (c) affinely dependent (collinear). This test can be applied for many other geometric predicates, such as determining whether two given line segments in the plane intersect, whether a given point lies within a given triangle, and whether a given point lies within the circumcircle of three other given points. (This will be discussed later in the semester.)

CMSC 754: Fall 2021 Dave Mount

Homework 1: Hulls and Plane Sweep

Handed out Thursday, Sep 16. Due by the start of class on Thursday, Sep 23. (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*.

Problem 1. Present an algorithm which, given a sequence $P = \langle p_1, \ldots, p_n \rangle$ of $n \geq 3$ points in $\mathbb{R}^2$, determines whether these points constitute the (distinct) sequence of vertices of a convex polygon given either in clockwise or counterclockwise order. The output of your algorithm is either "CW" (for a valid clockwise sequence), "CCW" (for a valid counterclockwise sequence), or "invalid".

Each point p_i is given by its coordinates (x_i, y_i). For full credit, your algorithm should run in $O(n)$ time and should only involve orientation test between the points of P. (You can receive 2/3 credit if you give a correct answer that involves other *exact* computations, such as comparing two coordinates and/or orientation tests involving points that are not part of the input set.)

For this problem, you may *not* assume general position. For example, three or more points might be collinear, and there might be duplicate coordinate values.

Prove that your algorithm is correct and justify its running time. (If you are unsure whether you need to prove a geometric assertion, feel free to check with me.)

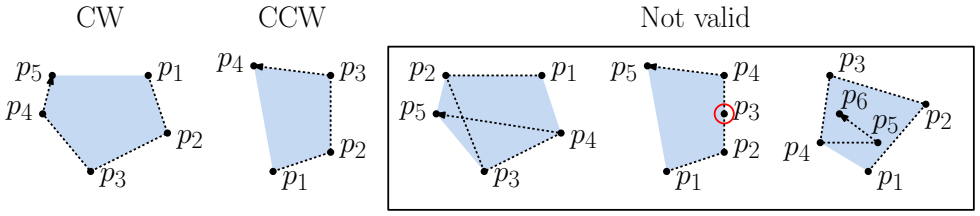


Figure 1: Problem 1: Vertices of a convex polygon?

Problem 2. A popular programming exercise is called the *skyline problem*. You are given a set of n axis-aligned rectangles, all having their bottom edge on the x-axis. The problem is to compute the union of these rectangles, called the *skyline*.

The input is presented as a set of n triples (l_i, r_i, h_i), where l_i is x-coordinate of the ith rectangle's left side, r_i is the x-coordinate of its right side, and h_i is its height (see Fig. 2). You may assume that all these values are positive reals.

The output consists of a *run-length encoding* of the horizontal edges of the boundary of the union of these rectangles. This consists of a sequence of the form $\langle (x_1, h_1), (x_2, h_2), \ldots, (x_m, h_m) \rangle$, where x_j is the start of the jth horizontal edge and h_j is its distance above the x-axis (see Fig. 2). The x values should increase monotonically ($x_j < x_{j+1}$), and consecutive heights

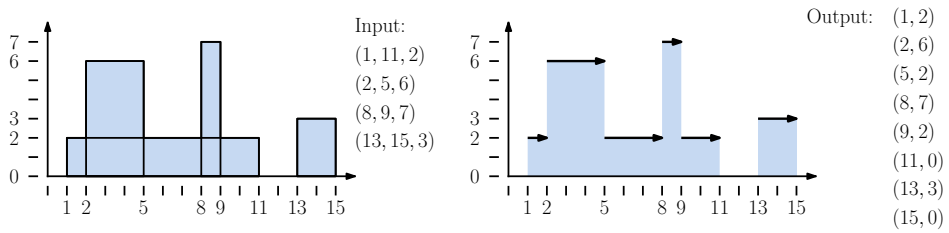


Figure 2: Problem 2: Skyline problem.

should be distinct ($h_j \neq h_{j+1}$). Each horizontal edge runs between x_j and x_{j+1} at $y = h_j$. The first height value h_1 must be nonzero, and the last height value h_m must be zero.

Present an output sensitive algorithm for the skyline problem. For full credit, your algorithm should run in time $O(n \log m)$, where m is the length of the output. (Note that $O(n \log n)$ is easy, and there are many answers on the internet, so you won't get much credit for such an algorithm.)

As always, first explain your strategy, present your algorithm (in English or pseudocode), prove its correctness, and derive its running time. The emphasis here is on getting the desired running time.

Hint: To simplify your description, you may assume that you have access to the following utility function for answering *horizontal ray-shooting queries*. (It's implementation will be left as an exercise.) Given a set of k vertical line segments, each with its lower endpoint on the x-axis, and given a query point $q = (q_x, q_y)$, find the first segment (if any) that is hit by a horizontal ray shot to the right from q. (If not segment is hit, the query returns some special value, like "no-hit".) You may assume that, given any such set of segments in $O(k \log k)$ time it is possible to build a data structure that can answer such queries in time $O(\log k)$.

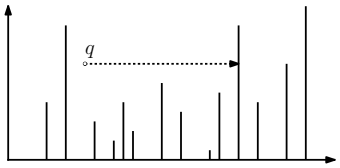


Figure 3: Horizontal ray shooting.

Problem 3. You have m robots that move along a common 1-dimensional track that runs north and south. Each robot provides you with its *motion plan*, which consists of a sequence of k *motion segments*. Each motion segment is given by a pair of reals (Δ, v), where $\Delta > 0$ denotes the duration of movement, and v is the robot's speed during this time interval. A positive speed indicates movement to the north and a negative speed indicates movement to the south. For example, the segment $(5, -13)$ means that the robot moves at 13 units per second to the south for 5 seconds. During this segment, the robot has moved a total of $5 \cdot 13 = 65$ units of distance to the south.

The full motion of the ith robot is specified as a sequence of k such segments, $\langle(\Delta_{i,1}, v_{i,1}), \ldots, (\Delta_{i,k}, v_{i,k})\rangle$. For example, for $k = 3$, given the sequence $\langle(2, 5), (3, -2), (1, 4)\rangle$, this robot will move at ve-

locity 5 for 2 seconds (moving 10 units north), then at velocity -2 for 3 seconds (moving 6 units to the south), and finally at velocity 4 for 1 second (moving 4 units north). Thus, over the span of 6 seconds, this robot has moved a net distance of $10 - 6 + 4 = 8$ units north.

The input to your program consists of the starting positions $\langle z_1, \ldots, z_m \rangle$ of the m robots along the track, together with m motion plans, one for each robot, each of length k. Once a robot reaches the end of its motion plan, it remains at its final position until all the robots have stopped moving. Let $n = km$ denote the entire size of the input.

Given this input, provide answers to the following questions:

(a) We say that two robots *collide* if they occupy the same location at the same time. As a function of k and m, what is the maximum number of collisions that there might be among a collection of m robots with k-element plans? Justify your answer. (For full credit, give an exact bound. Partial credit will be given for an asymptotic bound.)

(b) Present an efficient algorithm which, given the starting positions and motion plans, determines whether any of the robots ever collide. If there is a collision, your algorithm should output the first time that any two robots collide and indicate the indices of these two robots. If there are no collisions, indicate the closest that any two robots ever get to one another in the course of the entire motion process. For full credit, your algorithm should run in time $O(n \log n)$.

(c) Suppose that when robots collide, they simply pass through each other. Present an efficient algorithm which, given the starting positions and motion plans, reports all the collisions. Letting c denote the total number of collisions, your algorithm should run in time $O((c + n) \log n)$.

In all three cases, justify your algorithm's correctness and derive its running time.

Problem 4. Given a simple polygon P with n vertices, recall that the addition of any diagonal (an internal line segment joining two visible vertices of P) splits P into two simple polygons with n_1 and n_2 vertices respectively, where $n_1 + n_2 = n + 2$.

(a) Show that given any simple polygon P with $n \geq 4$ there exists a diagonal that splits P such that $\min(n_1, n_2) \geq \lfloor n/3 \rfloor$. (Hint: It may help to consider the dual graph of any triangulation.)

(b) Show that the constant $1/3$ is the best possible, in that for any $c > 1/3$, there exists a polygon such that *any* diagonal chosen results in a split such that $\min(n_1, n_2) < cn$. (You can provide a drawing, but it should be clear how your drawing can be generalized to all sufficiently large values of n.)

Challenge Problem 1: Present a solution to the horizontal ray-shooting problem stated in Problem 1. (Given a set of vertical line segments with their lower endpoints on the x-axis, determine the first segment hit by a horizontal ray shot to the right from any query point.)

Challenge Problem 2: Present an efficient algorithm which, given a set $P = \{p_1, \ldots, p_n\}$ of n points on the integer grid, computes the polygon of minimum perimeter that encloses these points, subject to the condition that the sides of this enclosure can be horizontal, vertical, or sloped at $\pm 45^\circ$ (see Fig. 4). (Hint: $O(n)$ time is possible, but $O(n \log n)$ time acceptable.

Prove that the enclosure produced by your algorithm has the minimum perimeter. Note that there may generally be many enclosures with the same perimeter, and your algorithm may output any of them.)

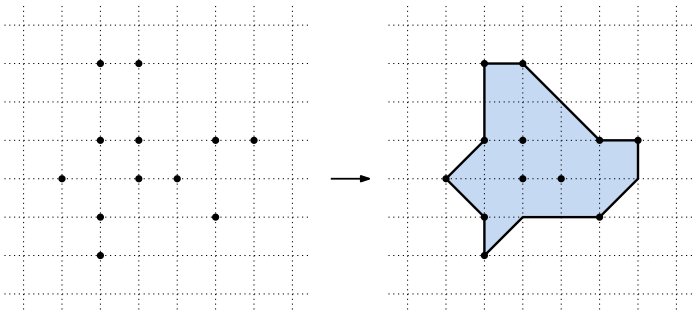


Figure 4: Minimum perimeter enclosure.

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.

CMSC 754: Fall 2021 Dave Mount

Homework 2: Duality, Linear Programming, and Point Location

Handed out Thursday, Oct 14. Due: **9:30am, Tuesday, Oct 26** (submission through Gradescope as with Homework 1). No late homeworks will be accepted, 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*. Also, when asked to give an algorithm with running time $O(f(n))$, it is allowed to give a *randomized* algorithm with *expected* running time $O(f(n))$.

Problem 1. Consider the two segments $s_1 = \overline{p_1 t_1}$ and $s_2 = \overline{p_2 t_2}$ shown in Fig. 1.

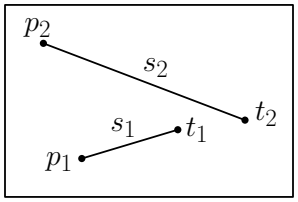


Figure 1: Problem 1: Trapezoidal map and point location.

(a) Show the (final) *trapezoidal map* for these two segments, assuming the insertion order $\langle s_1, s_2 \rangle$.

(b) Show the *point-location data structure* resulting from the construction given in class, assuming the insertion order $\langle s_1, s_2 \rangle$. (We will give partial credit if your data structure works correctly, even though it does not match the construction given in class.)

Please follow the convention given in class for the node structure. (In particular, for y-nodes, the left (resp., right) child corresponds to the region above (resp., below) the segment.)

Problem 2. Euler's formula is useful for computing the combinatorial properties of planar subdivisions. A planar graph (or more accurately, a cell complex) is a subdivision of the plane into vertices (0-dimensional), edges (1-dimensional), and faces (2-dimensional). Let v, e, and f denote the number of vertices, edges, and faces, respectively, in a given cell complex. (Note that f includes the unbounded face that extends to infinity.) Euler's formula states that these quantities are related as

$$2 \;=\; v - e + f$$

For example, in the Fig. 2(a) we show a triangulation of a set of $v = 16$ vertices with $h = 10$ vertices on the convex hull. In Fig. 2(b) we show a quadrilateral cell complex. Using Euler's formula, answer each of the following questions.

(a) Given a triangulation with v vertices, where h of these lie on the convex hull, use Euler's formula to derive the formula for the number of triangles t and the number of edges e in the triangulation as functions of v and h. (Hints: Observe that $3t = 2e - h$. This follows because $3t$ counts the total number of edges incident to all the triangles, and this counts

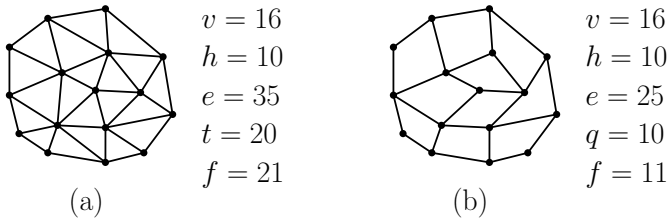


Figure 2: Problem 2: Applications of Euler's formula.

every edge twice except the h edges that form the convex hull, which are only counted once. Also observe that the number of faces is $f = t + 1$, because the infinite exterior face is counted as a face.)

(b) Given a quadrangulation (a cell complex where each face has four edges, excluding the face lying outside the convex hull) with v vertices, where h of these lie on the convex hull, use Euler's formula to derive the formula for the number of quadrangles q and the number of edges e in the quadrangulation as functions of v and h.

(c) Explain why your answer to (b) implies that a quadrilateralization does not exist if the number of hull vertices is odd.

Problem 3.

(a) You are given two sets of points, red and blue, in the plane. Let $R = \{r_1, \ldots, r_n\}$ be the red points and $B = \{b_1, \ldots, b_n\}$ be the blue points. The problem is to determine a pair of parallel, nonvertical lines ℓ_R and ℓ_B such that all the points of R lie on or above ℓ_R, all the points of B lie on or below ℓ_B, and the signed vertical distance from ℓ_B to ℓ_R is as large as possible. (More formally, if y_B and y_R are the y-intercepts of these lines, we want to maximize $y_R - y_B$.)

Note that if ℓ_R lies above ℓ_B, this distance is positive and if (as shown in the figure below) ℓ_R is below ℓ_B, this distance is negative. (When negative, the objective is to minmize the absolute value of the distance.) Present an $O(n)$ time algorithm to solve this problem. (**Hint:** Reduce to linear programming.)

(b) You are given a convex polygon K in $\mathbb{R}^2$, presented by its vertices in counterclockwise order $\langle v_1, \ldots, v_k \rangle$. We assume that the origin is contained in K's interior (see Fig. 3(b)). Given a positive real scalar α and a translation vector $t = (t_x, t_y)$, let $t + \alpha K$ denote the convex polygon that arises by scaling all the vertices of K uniformly by a factor of α (about the origin) and then translating them by vector t (see Fig. 3(c)). Such a body is called a *homothet* of K of scale α.

Present an algorithm, which given a convex polygon K (as described above) and an n-element point set $P = \{p_1, \ldots, p_n\}$ in $\mathbb{R}^2$, computes the homothet of K of smallest (positive) scale that contains all the points of P. This can be solved in $O(kn)$ time.

Hint: The reduction to LP involves multiple steps. Here are some suggestions:

(i) Explain how to express K as the intersection of k halfplanes $\{h_1, \ldots, h_k\}$.

(ii) Consider any halfplane $h = \{(x, y) \mid ax + by \leq c\}$, and let $h' = t + \alpha h$ denote the halfplane that arises by scaling the points of h by α (about the origin) and

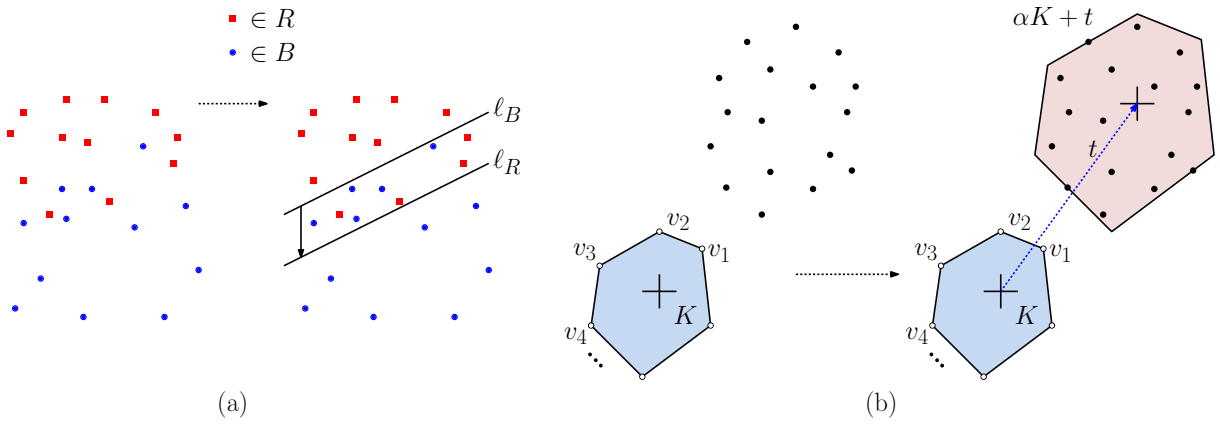


Figure 3: Problem 3: (a) Separating point sets and (b) enclosing points by a polygonal shape.

translating by the vector t. Given a point $p = (p_x, p_y)$. Prove that $p \in h'$, if and only if

$$a\frac{p_x - t_x}{\alpha} + b\frac{p_y - t_y}{\alpha} \leq c.$$

(iii) Use (i) and (ii) to obtain an $O(kn)$ time algorithm that computes the minimum-scale homothet of K enclosing P.

Since we have not done any examples of linear programming applications in class, here is simple example of how to answer one of these problems.

Sample Problem: Present an $O(n)$ time algorithm, which given two sets of points $R = \{r_1, \ldots, r_n\}$ and $B = \{b_1, \ldots, b_n\}$, both in $\mathbb{R}^3$, determines whether their exists a plane h in $\mathbb{R}^3$ such that all the points of R lie on or above h and all the points of B lie on or below h.

Sample solution: We reduce the problem to linear programming in $\mathbb{R}^3$. Let's assume that each $r_i \in R$ is given in coordinate form as $(r_{i,x}, r_{i,y}, r_{i,z})$ and similarly for B. Let's model h by the equation $z = ax + dy + e$, for some real parameters a, d, and e. To enforce the condition that each r_i lies on or above h and each b_j lies on or below it, we add the constraints

$$\begin{aligned} r_{i,z} &\geq ar_{i,x} + dr_{i,y} + e, \qquad \text{for } 1 \leq i \leq n \\ b_{j,z} &\leq ab_{j,x} + db_{j,y} + e, \qquad \text{for } 1 \leq j \leq n. \end{aligned}$$

We then invoke LP with $2n$ constraints in $\mathbb{R}^3$ (with the variables (a, d, e)). Since this is a yes-no answer, we don't really care about the objective function. We can set it arbitrarily, for example, "maximize e" (which is equivalent to using the objective vector $c = (0, 0, 1)$).

We interpret the LP's result as follows. If the result is "infeasible", then we know that no such plane exists. If the answer is "feasible" or "unbounded", then we assert that such a plane exists (assuming general position). This is clearly true if the result is "feasible",

since we can just take h to be the plane associated with the optimum vertex (a, d, e). If the result is "unbounded", then the plane is vertical, but there exists a perturbation such that R lies above and B lies below.

Problem 4. The objective of this problem is to explore some interesting properties of trapezoidal maps (which apply more generally to many geometric structures). Throughout this problem, $S = \{s_1, \ldots, s_n\}$ denotes a set of n nonintersecting, nonvertical line segments in the plane. Let $\mathcal{T}(S)$ denote the trapezoidal map of these segments. We say that a trapezoid $\Delta \in \mathcal{T}(S)$ is *incident* on a segment $s \in S$ if s borders Δ from above or below, or if one of s's endpoints bounds Δ from the left or the right (see Fig. 2(a)). For $s \in S$, define $\deg(s)$ to be the number of trapezoids of $\mathcal{T}(S)$ that are incident on s (in Fig. 4(a), $\deg(s) = 7$).

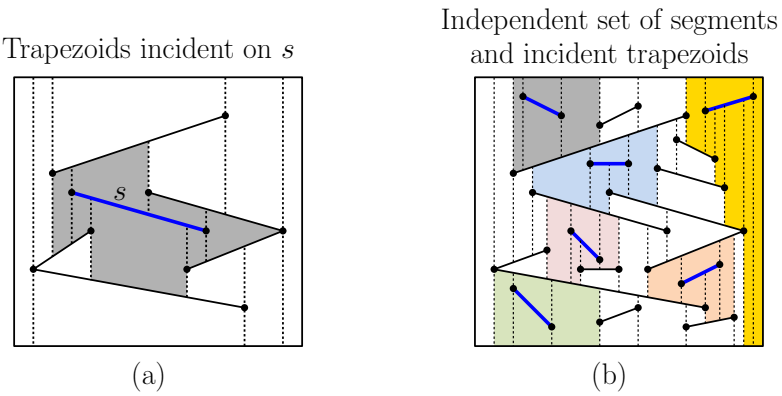


Figure 4: Problem 4: Independent sets in a trapezoidal map.

(a) Given any set S of n segments, prove that there exists a constant c, such that, for all sufficiently large n, $\sum_{s \in S} \deg(s) \leq cn$.

(b) Let c be the constant derived in your solution to (a). We say that a segment $s \in S$ is *long* if $\deg(s) \geq 2c$, and otherwise we say that s is *short*. Let $S' \subseteq S$ be the set of short segments of S. Prove that there exists a constant c' (which may depend on c) such that, for all sufficiently large n, $|S'| \geq n/c'$.

(c) We say that two segments $s_i, s_j \in S$ are *adjacent* if there exists a trapezoid $\Delta \in \mathcal{T}(S)$ that is incident on both s_i and s_j. Define an *independent set* of S to be a subset of S whose elements are pairwise non-adjacent (see Fig. 4(b)). Given the previous constants c and c', prove that there exists a constant $c'' > 1$ (depending on c and c') such that, for all sufficiently large n, S contains an independent set of size at least n/c'' *consisting entirely of short segments*. (**Hint:** Use a greedy approach.)

You might wonder why we care about independent sets at all. The existence of large independent sets is of critical to the efficiency of many algorithms based on divide-and-conquer. The idea is to find a large independent set, remove it, and solve the problem recursively on the remaining objects. (Since the number of remaining objects decreases by a constant factor, these total time for all these recursive calls will be small.) When the recursion returns, add back the elements of the independent set, and solve the problem. Since the various pieces are

independent of each other, the solutions of these independent subproblems will not interact with each other.

Problem 5. You are given two vertical lines at $x = 0$ and $x = 1$ and a set of n (nonvertical) line segments, $s_i = \overline{a_i b_i}$. The left endpoint a_i of each segment lies on a vertical line $x = 0$ and the right endpoint b_i lies on the vertical line $x = 1$ (see Fig. 5(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. 5(b)). Let us also add an imaginary "sentinel segment" s_0 that runs along the right vertical line.

Observe that for $1 \le i \le n$, every segment s_i is *terminated* by some other segment. If the segment survives to the right side, then it is terminated by segment 0. (For example, in Fig. 5(b), segment 1 is terminated by segment 2, segments 2, 3, and 4 are all terminated by segment 5, segment 5 is terminated by ∞, and so on.)

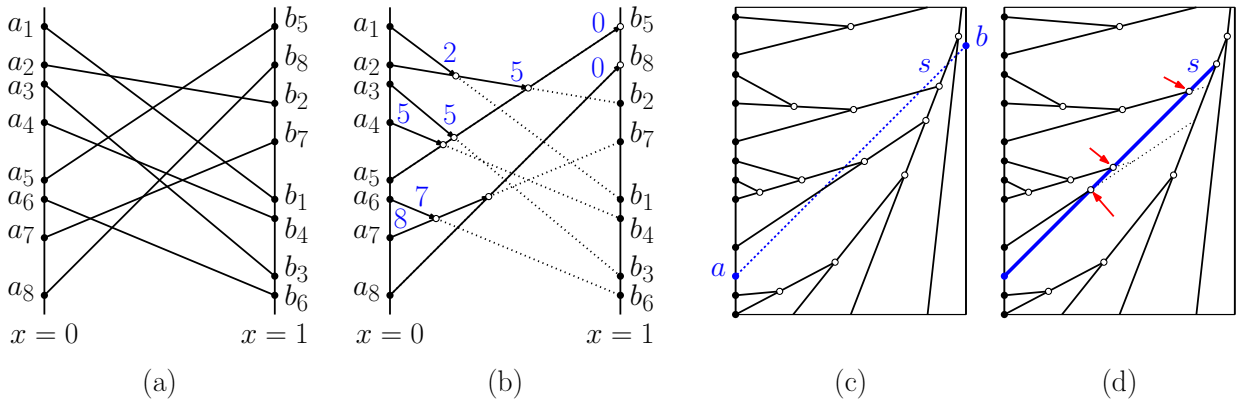


Figure 5: Problem 5(a) and (b): Terminating segments

(a) Assuming that the segments are given in sorted order according to their left endpoints (say, from top to bottom as shown in our figure), present an efficient algorithm that determines the n-element list of the indices of the segment terminators. (For example, for the input shown in the figure, the output would be $\langle 2, 5, 5, 5, 0, 7, 8, 0\rangle$.) **Hint:** $O(n)$ time is possible.

(b) Suppose that we instead insert the segments in random order. A new segment $s = \overline{ab}$ runs from left to right until it is terminated by the first segment of higher slope that it intersects. In addition all the segments from the existing structure of lower slope that intersect s are now terminated by s. (For example, the blue segment s in Fig. 5(c) and (d) changes the termination points of three existing segments, as indicated by the red arrows.)

Prove that there exists a constant c such that, if the segments are inserted in random order, the expected number of existing segments that change their termination point is at most c. (**Hint:** Apply a backwards analysis.)

(c) Recall that in Graham's scan, we computed the upper hull of a set of points by adding the points in left-to-right order. Whenever a point p_i was added, we determined the

point p_j of tangency with respect to the current upper hull, and we added the edge $\overline{p_i p_j}$. We removed all points that were "shadowed" by the newly added edge. If you look at the entire history of edges generated by Graham's scan, you obtain a tree-like structure, as shown in Fig. 6 below.

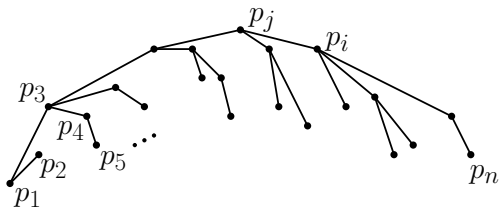


Figure 6: Problem 5(c): Graham's scan history

Prove that there is a equivalency between the tree-like structures of Fig. 5(b) and Graham-scan structure from Fig. 6. In particular, given any set of points $P = \{p_1, \ldots, p_n\}$ in the plane, explain how to map these to a set of segments $S = \{s_1, \ldots, s_n\}$ such that the edge $\overline{p_i p_j}$ is added by Graham's scan if and only if segment s_i is terminated by segment s_j.

Hint: This will involve some form of point-line duality, but you may need to modify the duality transformation given in class. It may also be necessary to change the x-coordinates associated with the left and right sides of the vertical band in the segment-termination problem, even so far as to take the limit as they tend to infinity.

Problem 6. You are given a collection of n nonintersecting circular disks in the plane, each of unit radius. Let $P = \{p_1, \ldots, p_n\}$ denote their center points. Preprocess these disks into a data structure to answer the following queries. Given a unit disk q_i (designated by its center point), determine whether it is possible for this disk to *escape* from the others, meaning that it is possible to move this disk arbitrarily far away from the others without intersecting or moving any of the disks of P.

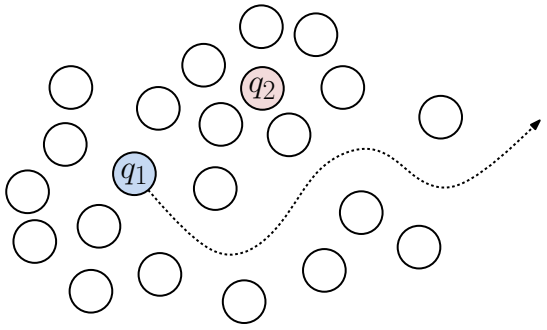


Figure 7: Problem 6. Disk q_1 can escape while q_2 cannot.

For example, given the disks in Fig. 7, the disk q_1 can escape from the others, while q_2 cannot. Present an $O(n \log n)$ that constructs such a data structure. Your data structure should have space $O(n)$ and should be able to answer queries in time $O(\log n)$. (**Hint:** Use Voronoi diagrams and point location.)

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.)

Present a randomized incremental algorithm for structure described in Problem 5(b). (In 5(b) you were asked just to show that the expected number of changes is $O(1)$. Here you must compute those changes and update the structure.)

Assume that your input is given as a collection of segments $S = \{s_1, \ldots, s_n\}$, where the endpoints lie on the vertical lines $x = 0$ and $x = 1$. Randomly permute segments and insert them one-by-one into the tree structure described in the problem. The aim is to produce the final structure in expected time $O(n \log n)$.

You may create any additional auxiliary structures you like in order to help achieve the desired running time.

Prove the correctness of your algorithm and derive its expected running time.

CMSC 754: Fall 2021 Dave Mount

Sample Problems for the Midterm Exam

The midterm exam will be this **Thursday, Oct 28 in class**. It will be *closed-book* and *closed-notes*, but 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.

Disclaimer: The following sample 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.

Problem 0. Expect a problem asking you to work through all or part of an algorithm that was presented in class on a specific example.

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

(a) Explain how to use *at most three* orientation tests to determine whether a point d lies within the interior of a triangle $\triangle abc$ in the plane. You do *not* know whether $\triangle abc$ is oriented clockwise or counterclockwise (but you may assume that the three points are not collinear).

(b) Let P be a simple polygon with n sides, where n is a large number. As a function of n, what is the maximum number of *scan reflex vertices* that it might have? Draw an example to illustrate.

(c) A convex polygon P_1 is enclosed within another convex polygon P_2 (see Fig. 1(a)). 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 sets of dual lines.

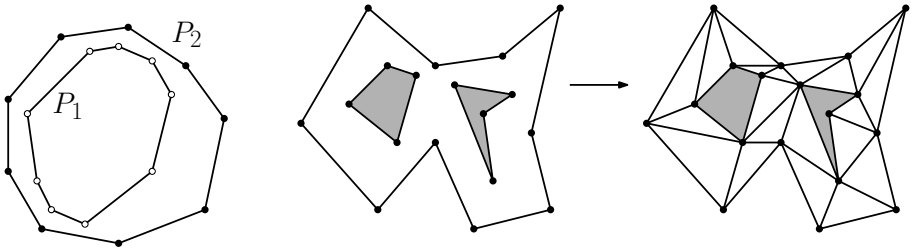


Figure 1: Problems 1(d) and 1(e).

(d) 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(b), $n = 10$, $h = 2$, and $k = 4$). As a function of n, h and k, how many triangles will such a triangulation have? Explain briefly.

(e) What was the importance of the Zone Theorem in our incremental algorithm for building line arrangements in the plane?

(f) Consider the linear-programming algorithm given in class for n constraints in dimension 2. In class we showed that the *expected-case* running time of the algorithm is $O(n)$. What is the *worst-case* running time of the algorithm? Briefly justify your answer (in a sentence or two).

(g) It is a fact that if P is a uniformly distributed random set of n points in a circular disk in the plane, the expected number of vertices of P's convex hull is $\Theta(n^{1/3})$. That is, the lower and upper bounds are both within some constant of $n^{1/3}$ for large n.

What is the average-case running time of Jarvis's algorithm for such an input? (If you forgot the running time of Jarvis's algorithm, we will give it to you for a 50% penalty on this problem.)

(h) Given a set P of n points in the plane, what is the maximum number of edges in P's Voronoi diagram? (For full credit, express your answer up to an additive constant.)

(i) 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)?

(j) 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.

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

(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)).

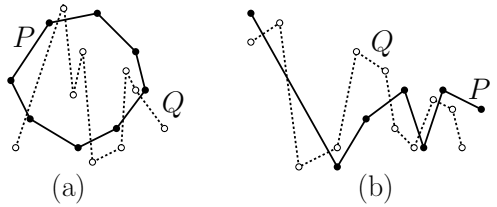


Figure 2: Maximum number of intersections.

(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. 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. 3). 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.

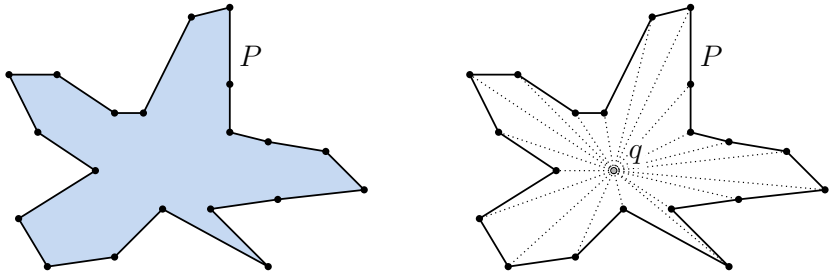


Figure 3: Determining whether a polygon is star-shaped.

Problem 4. A *slab* is the region lying between two parallel lines. You are given a set of n slabs, where each is of vertical width 1 (see Fig. 4). Define the *depth* of a point to be the number of slabs that contain it. The objective is to determine the maximum depth of the slabs using plane sweep. (For example, in Fig. 4 the maximum depth is 3, as realized by the small triangular face in the middle.)

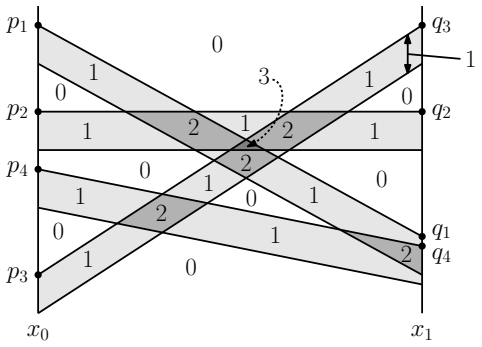


Figure 4: Maximum depth in a set of slabs.

We assume that the slabs lie between two parallel lines at $x = x_0$ and $x = x_1$. The ith slab is identified by the segment $\overline{p_i q_i}$ that forms its upper side (and the lower side is one unit below this). Let I denote the number of intersections between the line segments (both upper and lower) that bound the slabs. Present an $O((n + m) \log n)$-time algorithm to determine the maximum depth. (Hint: Use plane-sweep.)

Problem 5. 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 $\text{low}[x] = \text{high}[x] = p_1[x]$. Let $\text{low}[y] = \text{high}[y] = p_1[y]$.

(3) For $i = 2$ through n do:

(a) if $p_i[x] < \text{low}[x]$ then $(*)$ $\text{low}[x] = p_i[x]$.
(b) if $p_i[y] < \text{low}[y]$ then $(*)$ $\text{low}[y] = p_i[y]$.
(c) if $p_i[x] > \text{high}[x]$ then $(*)$ $\text{high}[x] = p_i[x]$.
(d) if $p_i[y] > \text{high}[y]$ then $(*)$ $\text{high}[y] = p_i[y]$.

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 6. You are given a set of n vertical line segments in the plane $S = \{s_1, \ldots, s_n\}$, where each segment s_i is described by three values, its x-coordinate x_i, its upper y-coordinate y_i^+ and its lower y-coordinate y_i^-. Present an efficient an algorithm to determine whether there exists a line $\ell : y = ax + b$ that intersects all of these segments (see Fig. 5). Such a line is called a *transversal*. (Hint: $O(n)$ time is possible.) Justify your algorithm's correctness and derive its running time.

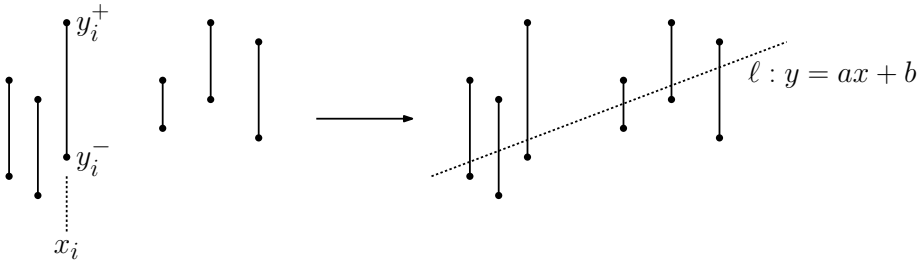


Figure 5: Existence of a transversal.

Problem 7. You are given three n-element point sets in $\mathbb{R}^2$, $R = \{r_1, \ldots, r_n\}$, called *red*, $G = \{g_1, \ldots, g_n\}$, called *green*, and $P = \{p_1, \ldots, p_n\}$, called *purple*. For each of the following two problems, present a reduction to linear programming in a space of constant dimension. Indicate which variables are used in the LP formulation, what the constraints are, and what the objective function is. Indicate what to do if the LP returns an answer that is infeasible or unbounded (if that is possible).

(a) A (linear) *slab* is a region of the plane bounded by two parallel lines, $y = ax + b^+$ and $y = ax + b^-$. Given R, G, and P, compute the slab (if it exists) of minimum vertical height such that all the points of R lie strictly above the slab, all the points of G lie within the slab, and all the points of P lie strictly below the slab (see Fig. 6(a)). If no such slab exists, you should detect and report this.

(b) A *parabolic slab* is the region of the plane bounded between two "parallel" parabolas, $y = ax^2 + bx + c^+$ and $y = ax^2 + bx + c^-$. Given R, G, and P, compute the parabolic slab of minimum vertical distance such that all the points of R lie strictly above the slab, all the points of G lie within the slab, and all the points of P lie strictly below the slab (see Fig. 6(b)). If no such parabolic slab exists, you should detect and report this.

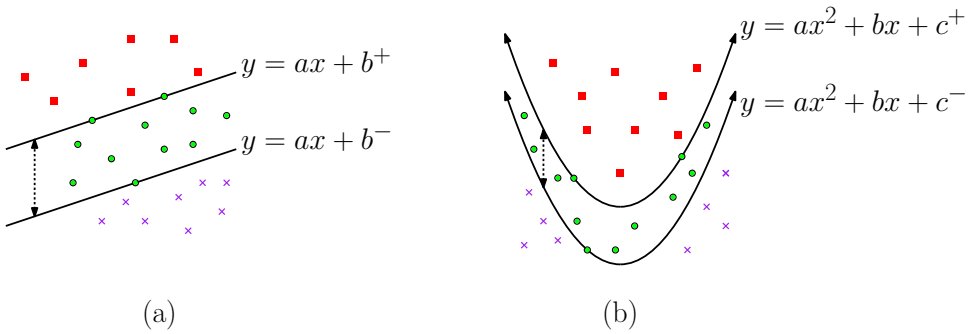


Figure 6: Linear and parabolic slabs.

Problem 8. 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. (Hint: It may be easier to consider the question in its inverse form, do the convex hulls *not* intersect.)

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)).

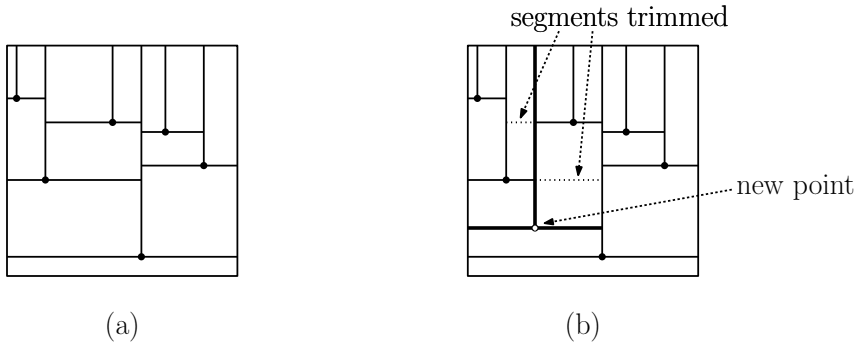


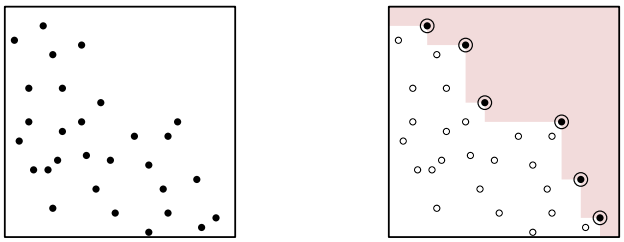
Figure 7: Building a subdivision.

(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. 7(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 *Pareto maximal* if it is not dominated by any other point of P (shown as black points in Fig. 8(b)).



(a) (b)

Figure 8: Paresto maxima.

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.)

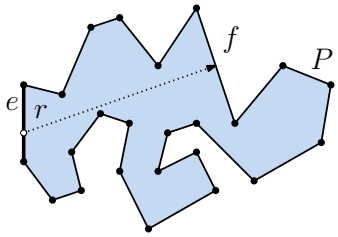


Figure 9: Ray-shooting queries.

Problem 12. You are given a set P of n points in $\mathbb{R}^2$. 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.)

(a) The input to the query is a nonvertical line ℓ. Such a line 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. 10(a)). The answer is the maximum vertical distance h between two lines parallel to h that lie between $P^+(\ell)$ and $P^-(\ell)$ (see Fig. 10(b)).

For simplicity, you may assume that neither set is empty (implying that h is finite).

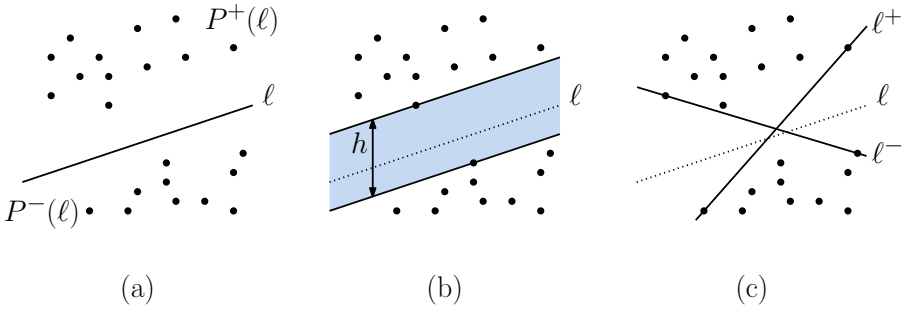


Figure 10: Separation queries.

(b) Again, the input to the query is a nonvertical line ℓ. The answer to the query consists of the two lines ℓ^- and ℓ^+ of minimum and maximum slope, respectively, that separate $P^+(\ell)$ from $P^-(\ell)$ (see Fig. 10(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 13. 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. 11.)

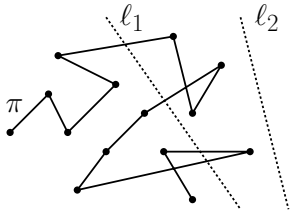


Figure 11: Path crossing queries.

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. 11 the answer for ℓ_1 is "yes," and the answer for ℓ_2 is "no.") (Hint: Duality is involved, but the solution requires a bit of "lateral thinking.")

Problem 14. 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 $\text{box}(p, q)$ to be the square centered at the midpoint of $\overline{pq}$ having two sides parallel to the segment $\overline{pq}$ (see Fig. 12(a)). The edge (p, q) is in the box graph if and only if $\text{box}(p, q)$ contains no other point of P (see Fig. 12(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. 12(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. 12(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.)

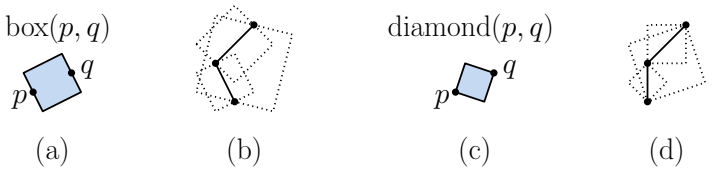


Figure 12: The box and diamond graphs.

Problem 15. 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. 13(a) are connected, but the disks of Fig. 13(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.

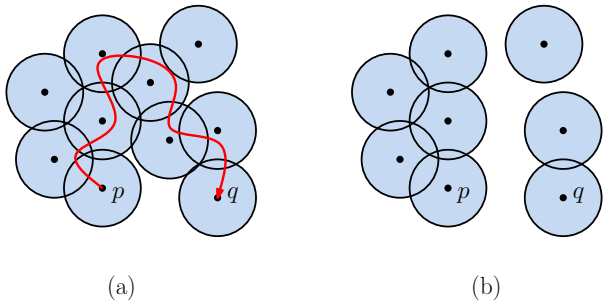


Figure 13: Safe connectivity.

Problem 16. 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. 14 below the string would be "$p_2p_1p_2p_5p_7p_9p_{10}$".)

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

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

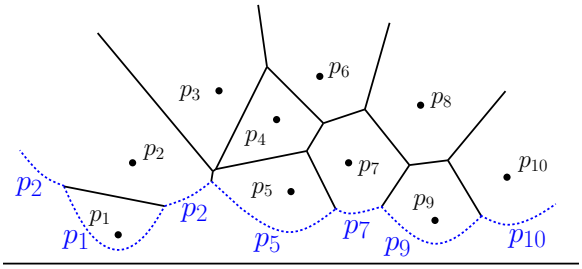


Figure 14: Beach-line complexity.

(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.)

Problem 17. Consider an n-element point set $P = \{p_1, \ldots, p_n\}$ in $\mathbb{R}^2$, and an arbitrary point $q \in \mathbb{R}^2$ (which is not in P). We say that q is *k-deep* within P if any line ℓ passing through q has at least k points of P on or above the line and at least k points of P on or below it.

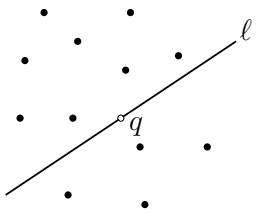


Figure 15: Point q is 4-deep within P.

For example, the point q in Fig. 15 is 4-deep, because any line passing through q has at least four points of P on either side of it (including lying on the line itself).

(a) Assuming we use the usual dual transformation, which maps point $p = (a, b)$ to line $p^* : y = ax - b$, explain what it means for a point q to be k-deep within P (in terms of the dual line q^* and the dual arrangement $\mathcal{A}(P^*)$).

(b) Present an efficient algorithm which, given P and q, determines the maximum value k such that q is k-deep within P. (**Hint:** $O(n \log n)$ time is possible. I will accept a slower algorithm for partial credit.)

(c) Present an efficient algorithm which, given P and an integer k, determines whether there exists a point q that is k-deep within P. (**Hint:** First consider what this means in the dual setting. $O(n^2 \log n)$ time is possible. I will accept a slower algorithm for partial credit.)

For parts (b) and (c) briefly justify your algorithm's correctness and derive its running time.

[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.

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 on the exam paper. 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 inputs 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. (20 points; 4–6 points each) Short-answer questions.

(a) In our plane-sweep algorithm for computing line segment intersections, we were careful to store only those intersection points involving pairs of segments that are *adjacent on the current sweep line*. Why did we do this?

(b) You are given four points a, b, c, d in $\mathbb{R}^2$. Using just orientation tests, show how to test whether the line segment $\overline{ab}$ intersects the line segment $\overline{cd}$. (Briefly explain.)

(c) True or False: If a simple polygon is both x-monotone and y-monotone, then it is monotone with respect to any direction. (Briefly explain your answer.)

(d) Consider the line arrangement shown in the figure below. Suppose that we insert the line ℓ_i into this arrangement. Indicate (by redrawing the figure) which edges of the arrangement are traversed by the insertion algorithm presented in class.

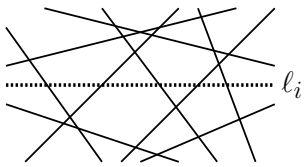


Figure 1: Inserting a line into an arrangement.

Problem 2. (15 points) The objective of this problem is to explore independent sets in triangulations. Throughout this problem, let $P = \{p_1, \ldots, p_n\}$ denote a set of n sites in the plane, and let $T(P)$ denote an arbitrary (not necessarily Delaunay) triangulation of P (see Fig. 2(a)). Define the degree of any site $p \in P$, denoted $\deg(p)$, to be the number of edges of $T(P)$ incident on it.

(a) (5 points) Prove that there exists a constant c, such that $\sum_{p \in P} \deg(p) \leq cn$, for all sufficiently large n.

(Recall that if there are h points on the convex hull, there are $2n - h - 2$ triangles and $3n - h - 3$ edges. Ideally, your answer should apply for any value of h, but for partial credit, you may assume that h has a specific value of your choosing.)

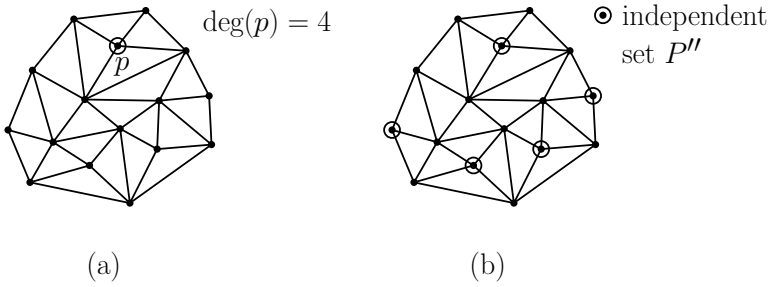


Figure 2: Independent sets in a triangulation.

(b) (5 points) Let c be the constant derived in your solution to (a). We say that a site $p \in P$ has *high degree* if $\deg(p) \geq 3c$, and otherwise it has *low degree*. Let $P' \subseteq P$ be the subset of low degree sites of P. Prove that there exists a constant c' (which may depend on c) such that, for all sufficiently large n, $|P'| \geq n/c'$.

(c) (5 points) Define an *independent set* to be a subset $P'' \subseteq P$ such that no two sites in P'' are adjacent in $T(P)$ (see Fig. 2(b)). Given the previous constants c and c', prove that there exists a constant $c'' > 1$ (depending on c and c') such that, for all sufficiently large n, P contains an independent set of size at least n/c'' *consisting entirely of low-degree sites*.

Problem 3. (30 points) In parts (a) and (c) below, you are asked to give a reduction to linear programming (LP). In each case, explain how the problem is formulated as an instance of LP (and what the dimension of the space is), and how the result of the LP (feasible, infeasible, unbounded) is to be interpreted in answering the problem.

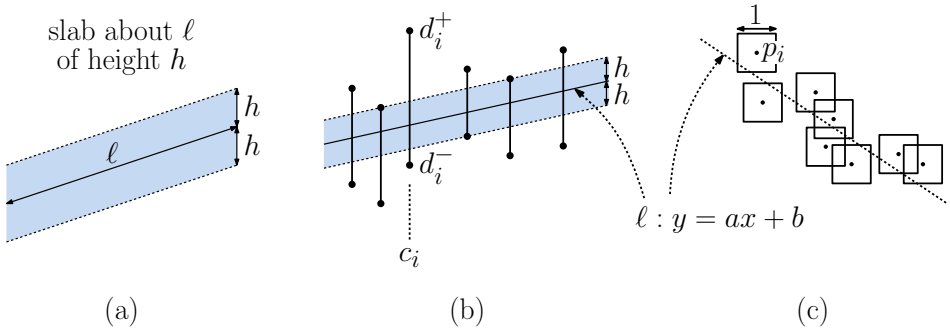


Figure 3: Stabbing segments.

(a) (10 points) Given a line ℓ, define the *slab* of height h centered about ℓ to be the region bounded between the two lines parallel to ℓ, one h units above and on h units below (see Fig. 3(a)).

You are given a set of n vertical line segments in the plane $S = \{s_1, \ldots, s_n\}$, where each segment s_i is described by three values, its x-coordinate c_i, its upper y-coordinate d_i^+ and its lower y-coordinate d_i^- (see Fig. 3(b)).

Apply LP to determine whether there exists a line $\ell : y = ax + b$ that intersects all of these segments. Further, if such a line exists, return the line ℓ with the property that it is the center of the slab of *maximum height* h that cuts through all the segment. (You can solve just the existence problem for partial credit.)

(b) (10 points) Suppose that your LP from part (a) reveals that there is no line that stabs all the segments. Instead, you decide to solve the following optimization problem. Given a set of vertical line segments S (as in part (a)), find a line $\ell : y = ax - b$ that intersects the *maximum* number of segments of S.

You decide to solve this problem in the dual setting. Using the dual transformation given in class, explain what the *equivalent optimization problem* is in the dual setting. (That is, explain how to dualize the line segments of S, how to dualize the line ℓ, and what property the dual point ℓ^* must satisfy so that we are effectively solving the same optimization problem.) You do *not* need to explain how to solve this dual problem.

(c) (10 points) You are given a collection of n axis-aligned unit squares in the plane. The squares are centered at the points $P = \{p_1, \ldots, p_n\}$, where $p_i = (c_i, d_i)$ (see Fig. 3(c)).

Apply LP to determine whether there exists a line $\ell : y = ax + b$ that intersects all of these squares. If it exists, return any such line.

Problem 4. (20 points) In this problem, we will consider two query problems involving a set of n circular disks in the plane (which may overlap), each of unit radius. Let $P = \{p_1, \ldots, p_n\}$ denote their centers, and let us assume that at least one of these disks contains the origin O.

For each of the parts below, explain how to preprocess these disks into a data structure to answer the specified query. In each case, your data structure should use $O(n)$ space, be constructed in $O(n \log n)$ time, and answer queries in $O(\log n)$ time. Briefly justify the correctness and running times of your solutions.

(a) (15 points) Given a query point q, determine whether it is possible to move q to the origin, so that the path lies *entirely within the union of these disks*. For example, in Fig. 4(a), q_1 can reach the origin O but q_2 cannot. Note that if q does not lie within any disk, the answer is trivially "no".

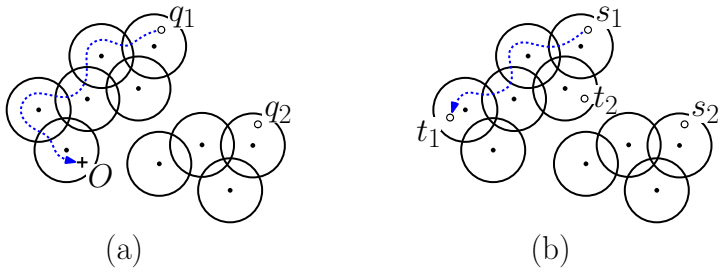


Figure 4: Motion planning among disks.

(b) (5 points) Given two query points s and t, determine whether it is possible to move from s to t, so that the path lies entirely within the union of these disks. For example, in Fig. 4(b), s_1 can reach t_1, but s_2 cannot reach t_2. (Hint: You can explain the changes you would make to the solution from (a).)

Problem 5. (15 points) This problem is inspired from applications in surveillance. Given a simple polygon P, we say that two points p and q are *visible* to each other if the open line segment between them lies entirely within P's interior. We allow for p and q to lie on P's boundary, but the segment between them cannot pass through any vertex of P (see Fig. 5(a)).

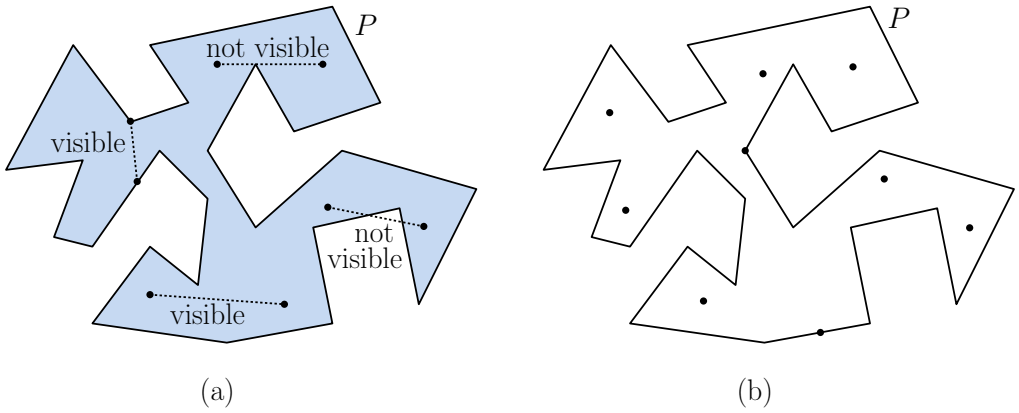


Figure 5: (a) Visibility and (b) a guarding set of size 9 for P.

A *guarding set* for P is any set of points G, called *guards*, lying in P (either on its boundary or in its interior) such that every point in P's interior is visible to at least one guard of G. Note that guards may be placed on vertices, along edges, or in P's interior (see Fig. 5(b)).

Prove that there exists a constant $c \geq 1$ such that (for all sufficiently large n) every n vertex simple polygon P has a guarding set of size at most n/c. For full credit, show that $c = 3$ works. For partial credit, show that some smaller value of c (e.g., $c = 2$) works. You do *not* need to show how to compute this set. (Hint: Decompose the polygon into simpler pieces.)

CMSC 754: Fall 2021 Dave Mount

Homework 3: Arrangements, Search, and Approximations

Handed out Tue, Nov 16. Due at the start of class on **Tue, Nov 30**. 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 $O(f(n))$. If your algorithm involves a plane sweep of a line arrangement and runs in time $O(n^2 \log n)$, you may assume that there exists a topological plane sweep variant that runs in time $O(n^2)$.

Problem 1. 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 (nonvertical) 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. 1(a).)

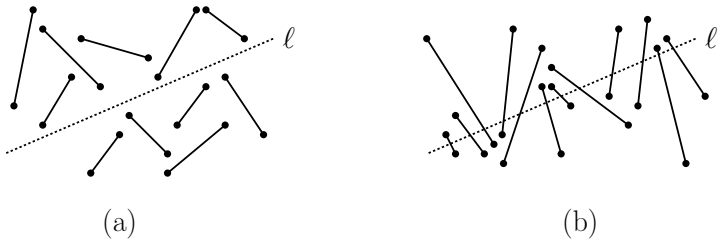


Figure 1: Stabbing segments.

(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. 1(b).)

(c) Part (b) looks very similar to a problem we have seen before of stabbing vertical segments. Provide an intuitive explanation as to what goes wrong if you were to try to adapt the LP-based solution to that problem to solve this problem.

Problem 2. *It was pointed out on Piazza that this problem as originally stated does not make sense since there always exist balanced slabs of unbounded height. Here is a version that (I hope) fixes these issues. If you believe that there are nonsensical or trivial solutions, please post a note to Piazza.*

You are given two sets of points R and B (red and blue) in $\mathbb{R}^2$. Let $n = |R| + |B|$ denote the total number of points. A *slab* $S(a, b^-, b^+)$ is the region between two pair of parallel lines, $S(a, b^-, b^+) = \{p : ap_x + b^- \leq p_y \leq ap_x + b^+\}$ (see Fig. 2(a)). Points lying on the slab's boundary may either be counted or excluded as lying within the slab. (This is needed,

since otherwise it is possible to have solutions that are arbitrarily close to optimal, but not optimal.) The *height* of the slab is the vertical distance between the lines, $b^+ - b^-$.

A slab is *balanced* if it contains an equal number of red and blue points. In this problem, we will consider computing the balanced slab of maximum height, which we'll call maxBS(R, B). Note that the maximum height slab may be of infinite height. To avoid some trivial special cases involving vertical (or nearly vertical) slabs, we add the additional constraint that the slab must contain at least one point from each set, and the slope a of the slab's bounding lines must satisfy $-1 \leq a \leq +1$.

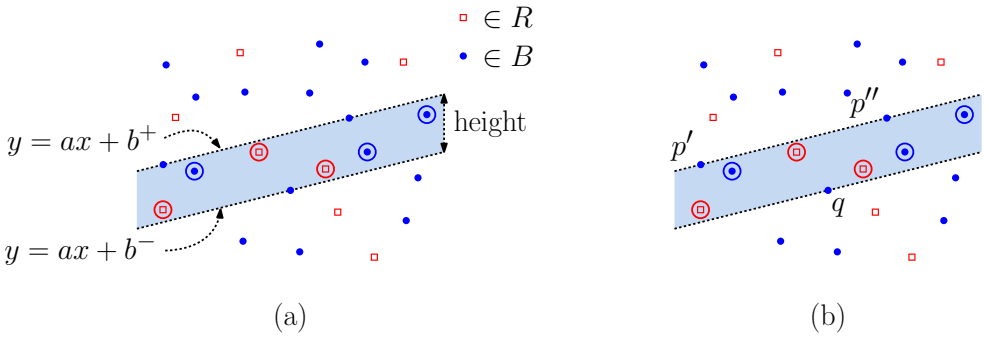


Figure 2: Balanced slab. (Note that the slab shown is *not* the maxBS(R, B).)

(a) Assuming the standard dual transformation given in class $(a, b) \leftrightarrow y = ax - b$, describe the dual-equivalent formulation of this problem. That is, what is the slab? what is its height? what condition is satisfied for the slab to have maximum height? (Please read parts (b) and (c) before writing down your answer.)

(b) Assuming that the point set $R \cup B$ is in general position, prove that if maxBS(R, B) is of finite height and its slope is strictly between -1 and $+1$ then it has three points on its boundary, two on one line and one on the other (see Fig. 2(b)).

(c) Present an efficient algorithm, which given R and B, computes the maxBS(R, B). Derive your algorithm's running time and justify its correctness.

Hint: This can be done in time $O(n^2 \log n)$, but it even processing is not obvious. You will get half credit if your algorithm run in $O(n^3)$ time.)

Problem 3. As mentioned in class, a WSPD is an efficient (approximate) representation for the complete graph on a set of points P. Another important structure is the complete *bipartite graph* on a pair of point sets. In this problem, we will explore this topic.

You are given two sets of points in $\mathbb{R}^d$, called R (for red) and B (for blue). Throughout, let $n = |R| + |B|$. A *bichromatic pair* is any pair of points (p, q), where $p \in R$ and $q \in B$. Given a parameter $s > 0$, define a *bichromatic s-WSPD* is a collection of pairs of subsets $\{(R_1, B_1), (R_2, B_2), \ldots\}$ such that

(i) $R_i \subseteq R$ and $B_i \subseteq B$,

(ii) R_i and B_i are s-well separated, and

(iii) for every bichromatic pair (p, q) there is exactly one pair (R_i, B_i) such that $p \in R_i$ and $q \in B_i$.

Given this definition, answer the following questions:

(a) Explain how to modify the standard WSPD algorithm given in class to produce a bichromatic s-WSPD for the sets R and B. (I do not need a complete algorithm description. You can explain what changes to make to the algorithm given in class.)

(b) Show that the asymptotic running time and total size of your bichromatic WSPD construction are the same as for the standard WSPD construction.

(c) Given R and B, define the *average bichromatic distance* to be

$$\Delta(R, B) \;=\; \frac{1}{|R| \cdot |B|} \sum_{p \in R} \sum_{q \in B} \|pq\|.$$

Present an algorithm that, given R, B, and $0 < \varepsilon < 1$, computes an ε-approximation to the average bichromatic distance. That is, your algorithm should return value Δ^* such that

$$\frac{\Delta(R, B)}{1 + \epsilon} \;\leq\; \Delta^* \;\leq\; (1 + \varepsilon) \cdot \Delta(R, B).$$

Your algorithm should run in time $O(n \log n + n/\varepsilon^d)$ time.

Problem 4. In this problem, we will consider how to use/modify range trees to answer a number of queries. In each case, the input is an n-element point set P in $\mathbb{R}^2$. In each case, explain what points are stored in the range tree, what the various levels of the range tree are, and how queries are answered. Finally, justify your algorithm's correctness and derive its storage and running time as a function of n.

(a) A *skewed rectangle* is defined by two points $q^- = (x^-, y^-)$ and $q^+ = (x^+, y^+)$. The range shape is a parallelogram that has two vertical sides and two sides with a slope of $+1$. The lower left corner is q^- and the upper right corner is q^+. The answer to the query is the number of points of P that lie within the parallelogram. (In Fig. 3(a), the answer to the given query is 3.)

(b) In a *vertical segment-sliding query* (VSS), you are given a vertical line segment with x-coordinate x_0 and endpoints at y-coordinates y_0 and y_1, where $y_0 < y_1$. The answer to the query is the point that is first hit if the segment is translated to the right. More formally, among the points of P whose y-coordinates lie within the interval $[y_0, y_1]$, the answer is the point with the smallest x coordinate greater than or equal to x_0. If there are no points that satisfy these conditions, the query returns the special value `null`. (For example, in Fig. 3(b), the query returns point a.)

(c) A *sector query* is defined by two nonnegative lengths $r_- < r_+$ and two nonnegative angles $\theta_- < \theta_+ \leq 2\pi$. This defines a shape bounded between two rays emanating from the origin at angles θ_- and θ_+ and two circular arcs between these rays at distances r_- and r_+ from the origin (see Fig. 3(c)). The answer to the query is the number of points of P lying within this region.

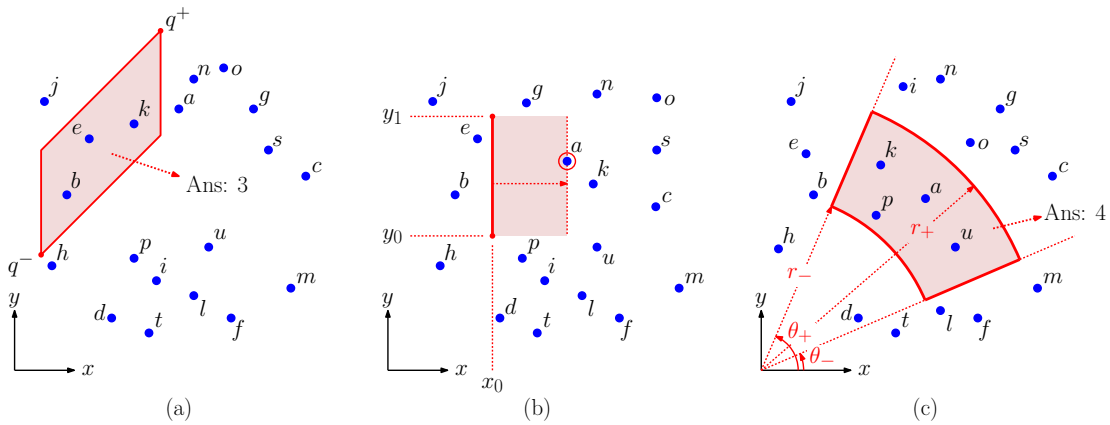


Figure 3: Using range trees to answer various queries.

Hint: In all cases, it is possible to answer queries in $O(\log^2 n)$ time without the need to resort to cascading.

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.

Challenge Problem. You are given a ball B in $\mathbb{R}^2$ and a collection of n lines $L = \{\ell_1, \ldots, \ell_n\}$ in the plane. For each of the following tasks, provide an $O(n \log n)$ time solution. (If you answer (c), you immediately have an answer to both (a) and (b).)

(a) Determine whether none of the lines L intersect within B (see Fig. 4(a)).

(b) Determine whether all of the lines of L intersect each other within B (see Fig. 4(b)).

(c) Count the number of pairs of lines of L that intersect within B (see Fig. 4(c)).

Hint: You may assume that all the lines intersect the ball. Start by labeling the points where the lines intersect the boundary of the ball with the line's index.

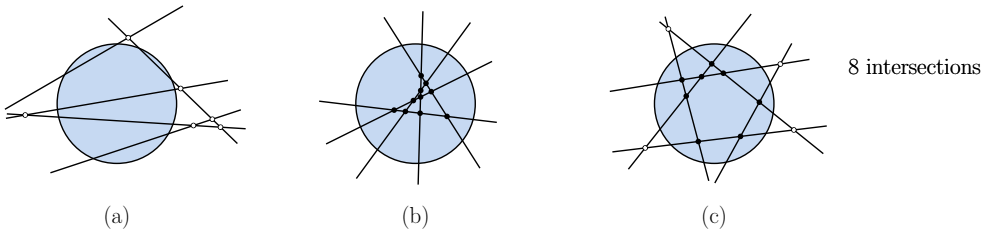


Figure 4: Challenge Problem.

CMSC 754: Fall 2021 Dave Mount

Homework 4: Sampling, Motion Planning, and More

Handed out Tue, Nov 30. Due, Tue, Dec 14, 9:30am. (There is no class that day, but I'll record pretend lecture where I will discuss the solutions.) 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 $O(f(n))$.

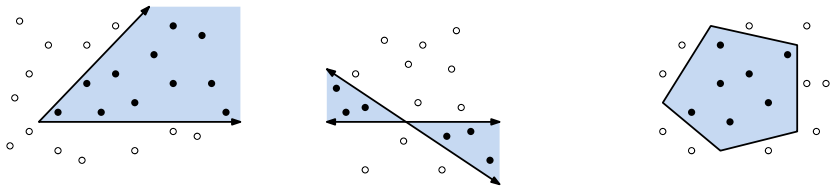
Problem 1. The objective of this problem is to investigate the *VC-dimension* of some range spaces. Recall that a *range space* Σ is a pair $(X, \mathcal{R})$, where X is a (finite or infinite) set, called *points*, and $\mathcal{R}$ is a (finite or infinite) family of subsets of X, called *ranges*.

For each of the following range spaces, derive its VC-dimension and prove your result. (Note that in order to show that the VC-dimension is k, you need to give an example of a k-element subset that is shattered and prove that no set of size $k+1$ can be shattered.) Throughout, you may assume that points are in general position.

(a) $\Sigma = (\mathbb{R}^2, W)$, where W consists of all horizontal wedges. A *horizontal wedge* is the region bounded between two rays, one horizontal and the other of arbitrary (finite) slope (see Fig. 1(a)). (It is my intention that the wedge angle is acute, but if you want to consider obtuse wedges instead, that is fine with me.)

(b) $\Sigma = (\mathbb{R}^2, W')$, where W' consists of all horizontal double wedges. A *horizontal double wedge* is the region bounded between two lines, one horizontal and the other of arbitrary (finite) slope. It consists of the points that lie above one line and below the other (see Fig. 1(b)).

Hint: Getting a tight upper bound on the VC-dimension seems to be very difficult. For this part, it suffices to come up with any constant upper bound, by whatever method you like. (See, e.g., the material on *canonical shapes* from the Lecture 20 Notes.) Deriving a better bound is left as the Challenge Problem.



(a) (b) (c)

Figure 1: VC-Dimension of some range spaces.

Also, answer the following.

(c) Prove that the range space $\Sigma = (\mathbb{R}^2, C)$ 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.

Here is a sample solution for parts (a) and (b), to give you some idea of what I am looking for. (It would be even nicer if you include some figures.)

Example: Consider the range space $\Sigma = (\mathbb{R}^2, H)$ where H consists of all closed horizontal halfspaces, that is, halfplanes of the form $y \geq y_0$ or $y \leq y_0$. We claim that $\text{VC}(\Sigma) = 2$.

$\text{VC}(\Sigma) \geq 2$: Consider the points $a = (0, -1)$ and $b = (0, 1)$. The ranges $y \geq 2$, $y \geq 0$, $y \leq 0$ and $y \leq 2$ generate the subsets $\{\emptyset, \{a\}, \{b\}, \{a, b\}\}$, respectively. Therefore, there is a set of size two that is shattered.

$\text{VC}(\Sigma) < 3$: Consider any three element set $\{a, b, c\}$ in the plane. Let us assume that these points have been given in increasing order of their y-coordinates. Observe that any horizontal halfplane that contains b, must either contain a or c. Therefore, no 3-element point set can be shattered.

Problem 2. In this problem we will explore an idea for constructing a *weak ε-net* for a set of P points in the plane. By a "weak" ε-net, we mean a set of points that satisfies the standard definition of an ε-net, but it can be formed from *any* set of points, not just the points of P. We'll give the broad outline, and you will fill in the details.

You are given n points P in $\mathbb{R}^2$. Let us make the general-position assumption that there are no duplicate x- or y-coordinates. We construct a set $S \subset \mathbb{R}^2$ as follows. We first compute an integer $k \geq 1$, and let $m = \lfloor n/k \rfloor$. Next, we sort the points of S in increasing order according to their x-coordinates, and let $\langle x_1, \ldots, x_n \rangle$ denote the resulting sorted sequence of coordinates. We take every kth element from this sorted sequence:

$$X \;=\; \{x_k, x_{2k}, x_{3k}, \ldots, x_{mk}\}.$$

We repeat the same process for the y-coordinates, by first sorting them in increasing order as $\langle y_1, \ldots, y_n \rangle$, and setting

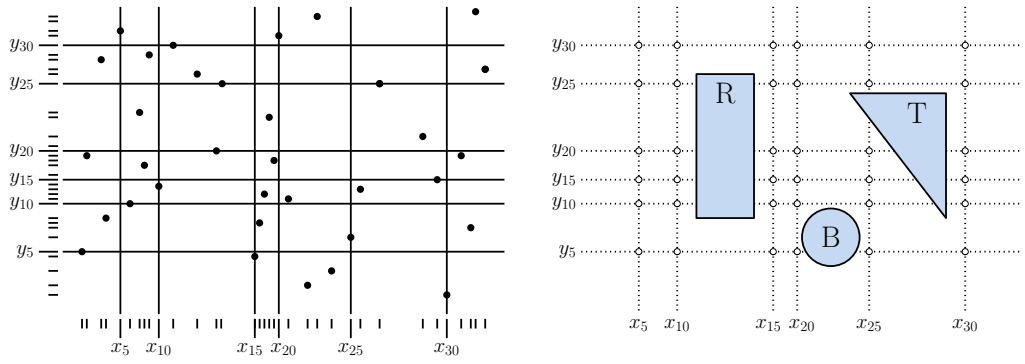
$$Y \;=\; \{y_k, y_{2k}, y_{3k}, \ldots, y_{mk}\}.$$

(Note that we use the same value of k and m in defining both X and Y.) Finally, we set $S = X \times Y$, that is, for $1 \leq i, j \leq m$, we include the point (x_i, y_j) into S. Clearly, S has m^2 elements. Observe that while they are based on the coordinates of the points of P, the points of S need not belong to P.

(a) For each of the following range spaces, answer the following question. If a range from the set contains no elements of S, what is the maximum number of elements of P that it might contain? (That is, if $Q \cap S = \emptyset$ then how large can $|Q \cap P|$ be?) In each case justify your answer. (Express your answer precisely, not as an asymptotic function.)

(R) Axis-aligned rectangles (of any width and height).

(T) Axis-aligned right triangles. This is defined to be any right triangle such that the legs (i.e., the sides incident to the 90° angle) are parallel to the x- and y-axes. The hypotenuse can have any slope.



(a) (b)

Figure 2: Weak ε-net construction (for $k = 5$).

(B) Euclidean balls (of any radius).

(b) Suppose you are given a parameter $\varepsilon > 0$. Based on your answers to (R), (T), and (B) above, what value should we set k to (as a function of n and ε) in the above construction so that the resulting set S is an ε-net (in the weak sense) for P. We want k to be as large as possible, so that the resulting ε-net is as small as possible.

(c) Suppose that further, we would like a weak ε-sample. For each of the range spaces, (R), (T), and (B) above, is there any value of k such that the resulting set S is an ε-sample of P, where the size of S is a function of ε (but not n)? If so, give this value and justify its correctness. If not, explain why the resulting set S's size must depend on n.

Problem 3. This problem has many parts, but each answer is very short, and sentence, formula, or short derivation suffices for most.

Iterative reweighting (seen in Lecture 19) can be used to solve a number of geometric optimization involving set systems of constant VC-dimension. In this problem, we will derive an algorithm for computing the circular disk enclosing a set of n points in $\mathbb{R}^2$. (We will derive an $O(n \log n)$ time solution, but there is actually an $O(n)$ time more efficient solution based on a randomized incremental approach.)

Given a set of points $P \subset \mathbb{R}^2$, define the *minimum enclosing ball*, MEB(P), to be the circular disk B of smallest radius that contains P. Generally, MEB(P) is determined by a constant number of points P^* on the boundary of the ball, called the *basis* (see Fig. 3(a)). (There at most three in $\mathbb{R}^2$ and generally at most $d+1$ points in $\mathbb{R}^d$). The following two facts are easy to prove:

- For any set S such that $P^* \subseteq S \subseteq P$, MEB($S$) = MEB($P^*$) (see Fig. 3(b)).
- For any set S such that $S \subseteq P$, radius(MEB(S)) $\leq$ radius(MEB(P)) (see Fig. 3(c)). (We cannot infer that MEB(S) $\subseteq$ MEB(P), since one disk might "bulge out" from the other.)

Suppose that we have access to a black-box algorithm for computing the MEB, but it is very slow and can only be applied to sets of constant size. Our algorithm operates by computing

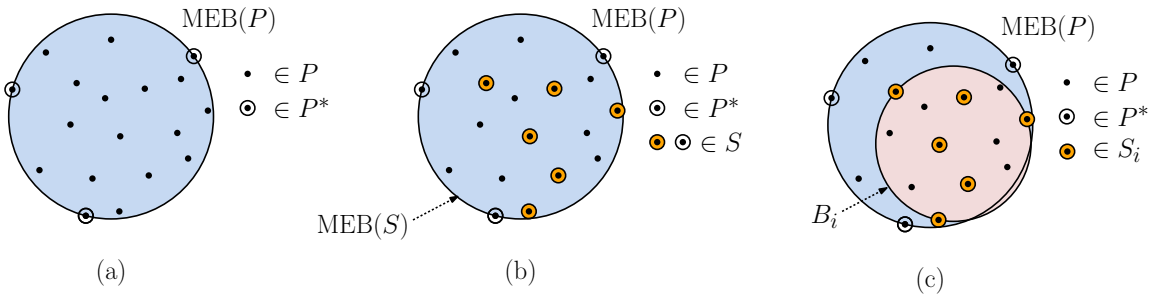


Figure 3: Reweighting algorithm for MEB.

an ε-net S of constant size for P with respect to the set system where the ranges are the points of P that lie *outside* of a circular disk. (We will see that this set system has constant VC-dimension.) We then invoke our slow MEB algorithm to compute the MEB for S. If the resulting ball contains P, we are done. If not, we double the weights of all the points of P outside this ball, and repeat. Here is the algorithm:

(1) Set $\varepsilon \leftarrow$ "fill this in later". For each $p \in P$, set $w_0(p) = 1$.

(2) Repeat for $i \leftarrow 1, 2, \ldots$

  (a) Let S_i be a weighted ε-net for P of size $O((1/\varepsilon)\log(1/\varepsilon))$ with respect to the weights w_{i-1} for ranges consisting of points lying *outside* a circular disk.

  (b) Let $B_i \leftarrow \text{MEB}(S_i)$

  (c) If $P \subseteq B_i$, then return B_i as the MEB (success)

  (d) Otherwise, for each $p \in P \setminus B_i$, set $w_i(p) \leftarrow 2w_{i-1}(p)$.

We will prove that this algorithm is correct (subject to a suitable choice of ε).

(a) Consider any set system $(X, \mathcal{R})$. The *complimentary* set system is $(X, \overline{\mathcal{R}})$, where $\overline{\mathcal{R}}$ is defined as follows. For each $R \in \mathcal{R}$, $\overline{\mathcal{R}}$ contains the compliment set $\overline{R} = X \setminus R$.

Prove that if $(X, \mathcal{R})$ has VC-dimension d, then its compliment $(X, \overline{\mathcal{R}})$ also has VC-dimension d. (Given the fact that the set system of Euclidean ball ranges has constant VC-dimension, this implies that the set system of points outside a Euclidean ball also has the same VC-dimension.)

(b) To establish (partial) correctness, prove that if in step 2(c), if the ball $B_i = \text{MEB}(S_i)$ contains all the points of P, then $B_i = \text{MEB}(P)$.

(c) Prove that after i (unsuccessful) iterations, $W_i(P) \leq n(1+\varepsilon)^i$. (Using the fact that $1 + x \leq e^x$, it further follows that $W_i(P) \leq ne^{\varepsilon i}$.)

(d) Suppose that the ith iteration is not successful. Prove that there exists at least one point of the basis P^* that does not lie in B_i.

(e) Using (d), prove that after i (unsuccessful) iterations, $W_i(P^*) \geq 2^{i/3}$. (**Hint:** You may use the fact given in Lecture 19 that the growth rate is slowest if the points of P^* are doubled equally often.)

(f) Based on (c) and (e), prove that $i \leq \lg n/((1/3) - \varepsilon \lg e)$.

(g) Based on (f), specify a (constant) value of ε such that the algorithm is guaranteed to terminate within $O(\log n)$ iterations.

(h) Assuming that an ε-net can be computed in $O(n)$ time, show that this algorithm runs in time $O(n \log n)$.

Hint: The analysis is structurally similar to the one of Lecture 19 on the iterative reweighting hitting set algorithm. I would suggest consulting the latex lecture notes, which has a bit more details than the hand-written notes.

Problem 4. In this problem, we will be planning the motion of a line-segment robot $\mathcal{R}$ in the plane amidst a collection of obstacles consisting of n disjoint obstacles $\mathcal{P} = \{P_1, \ldots, P_n\}$. Each obstacle is an axis-parallel rectangle. In particular $P_i = [x_i^-, x_i^+] \times [y_i^-, y_i^+]$. The robot is 2-units long, and its reference position oriented vertically with its midpoint at the origin (see Fig. 4(a)). The robot is restricted to two types of motion:

Translation: It can translate through a vector $t = (t_x, t_y)$, moving its reference point from its current position p to $p + t$.

Rotation: The robot can rotate about its reference point by either $+90°$ (that is, counterclockwise) or $-90°$ (that is, clockwise). When a rotation is performed, the entire circular arc swept out by the any point on the segment must be free of any obstacles. (Think of the segment as spinning in the plane—not picking it up, rotating it, and putting it down.)

Therefore, the robot's *configuration* consists of a point (x, y) where its center point is located and its orientation $o \in \{V, H\}$, where "V" indicates that the robot is parallel with the y-axis and "H" indicates that it is parallel with the x-axis.

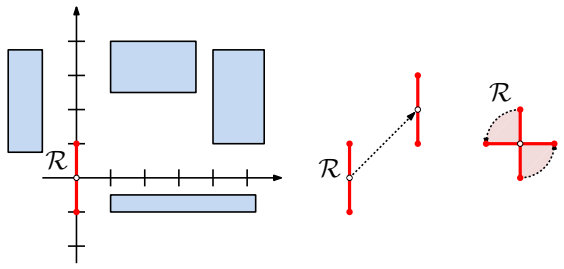


Figure 4: Motion planning for a rotating/translating segment.

A motion plan consists of a sequence of translations and rotations. Note that the robot either translates or rotates. It cannot translate while rotating.

(a) For each of the possible orientations of the robot ("H" or "V"), describe the shape of the corresponding collision obstacle $\mathcal{C}_{\mathcal{R}}(P_i)$ (in terms of the parameters $x_i^-, x_i^+, y_i^-, y_i^+$).

(b) For each of the possible orientations of the robot ("H" or "V"), describe the shape of the corresponding collision obstacle $\mathcal{C}_{\mathcal{R}}(P_i)$ in the cases of $+90°$ (counterclockwise) and

$-90°$ (clockwise) rotation (in terms of the parameters $x_i^-, x_i^+, y_i^-, y_i^+$). There are four shapes in all and their boundaries will involve circular arcs.

(c) Given your answers to (a) and (b), present an algorithm to determine whether there exists a motion plan from an arbitrary starting placement configuration $s = (x_s, y_s, o_s)$ to a given target $t = (x_t, y_t, o_t)$. You may assume that both s and t are collision-free.

Hint: I'm really looking for a high level description of how to combine reachability among the various collision obstacles. Efficiency is not a huge consideration, but your solution should run in polynomial time in n.

Challenge Problem. Provide a better upper bound on the VC-dimension for horizontal double wedges. (This is intentionally open-ended, and you don't need to have the upper bound exact to get full credit for this problem.)

CMSC 754: Fall 2021 Dave Mount

Sample Problems for the Final Exam

The final exam will be **Thu, Dec 16, 8:00am-10:00am** in class. 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. Every vertex has degree 3. How many vertices and edges does the dodecahedron have? Show how you derived your answer.

(b) Given a set P of n points in the plane, what is the maximum number of edges in P's Voronoi diagram? (For full credit, express your answer up to an additive constant.)

(c) 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)?

(d) 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

(e) 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.

(f) 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

(g) 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.

Problem 2. 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)).

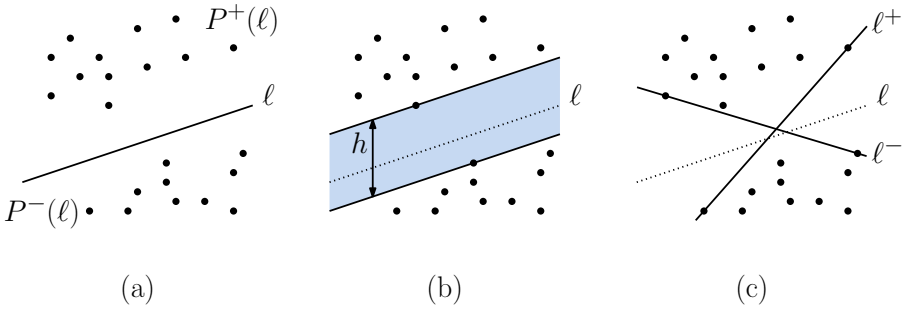


Figure 1: Query problem.

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.)

(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. 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. 2.)

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. 2 the answer for ℓ_1 is "yes," and the answer for ℓ_2 is "no.") (Hint: Duality is involved, but the solution requires a bit of lateral thinking.)

Problem 4. Consider the following two geometric graphs defined on a set P of points in the plane.

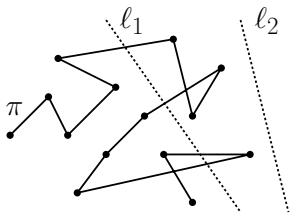


Figure 2: Path crossing queries.

(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. 3(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. 3(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. 3(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. 3(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.)

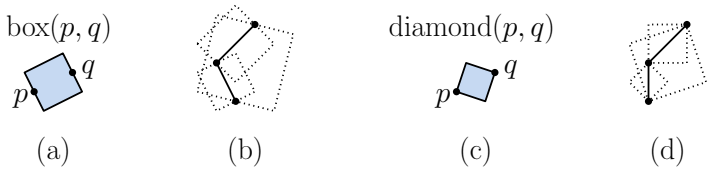


Figure 3: The box and diamond graphs.

Problem 5. Consider the range space $(P, \mathcal{R})$ where P is a set of n points in the plane, and $\mathcal{R}$ 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.

(c) Prove from first principles that $|\mathcal{R}| = O(n^2)$, where $n = |P|$. You are *not* allowed to appeal to Sauer's lemma. (Hint: Explain how to map each range to one of $O(n^2)$ *canonical halfplanes*, containing the same set of points as the original halfplane.)

Problem 6. (Here is another problem on VC-dimension from another semester.)

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. 4(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. 4(b)).

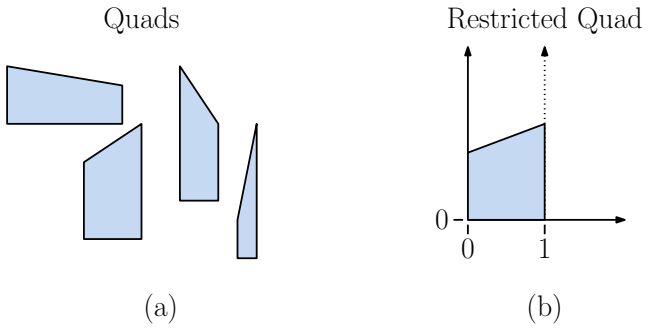


Figure 4: Quads and restricted quads.

(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 $\overline{p_1p_3}$.)

(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 7. 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 8. A set P of n points in $\mathbb{R}^d$ determines a set of $\binom{n}{2}$ different distances. Define $\Delta(P)$ to be this set of distances $\{\|p_i - p_j\| : 1 \leq i < j \leq n\}$. Given an integer k, where $1 \leq k \leq \binom{n}{2}$, we are interested in computing the kth smallest distance from this set. Normally, this would take $O(n^2)$ time, so let's consider a fast approximation algorithm.

Let $\delta(P, k)$ denote the exact kth smallest distance in $\Delta(P)$. Given $\varepsilon > 0$, a distance value x

is an ε-approximation to $\delta(P,k)$ if

$$\frac{\delta(P,k)}{1+\varepsilon} \;\leq\; x \;\leq\; (1+\varepsilon)\delta(P,k).$$

Present an efficient algorithm to compute such a value x. Justify your algorithm's correctness and derive its running time. (**Hint:** Use well-separated pair decompositions. You may assume that, when the quadtree is computed, each node u of the quadtree is associated with an integer $\mathrm{wt}(u)$, which indicates the number of points of P lying within u's subtree.)

You may assume that d and ε are constants (independent of n). I know of an algorithm that runs in time $O(n \log n + n/\varepsilon^d)$ time, but I will accept for full credit an algorithm that runs in time $O((n \log n)/\varepsilon^d)$.

Problem 9. 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 10. This problem considers motion planning 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. 5). 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 $x^+ > x^- + 1$.)

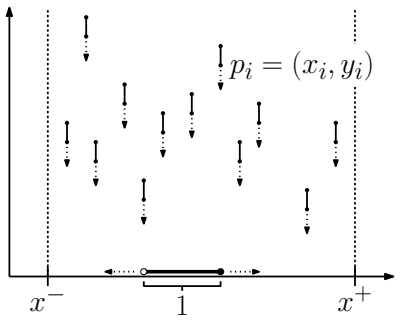


Figure 5: Robot motion planning.

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 $x^- \leq x_i \leq x^+$.

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.

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 on the exam paper. If you have a question, either raise your hand or come to the front of class. Total point value is 135 points. Good luck!

In all problems, unless otherwise stated, you may assume that inputs 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. (35 points) Short-answer questions. Unless requested, explanations are not required.

(a) (5 points) Given a set of points P in $\mathbb{R}^2$, there are k_1 edges on P's upper hull and k_2 edges on P's lower hull. What can we say about the number of vertices on the upper and lower envelopes of the dual line arrangement P^*? (Assume the standard dual transformation we use in class.)

(b) (15 points) For each of the following, indicate whether it *can or cannot* be solved by a reduction to a constant number of instances of linear programming of size $O(n)$ in a space of constant dimension. If it can be, indicate the *dimension* of the LP instance. *Do not* give the LP formulation.

(i) Given a set of n vertical line segments in $\mathbb{R}^3$ (that is, all parallel to the z-axis), does there exist a plane ℓ that intersects all of these segments?

(ii) Given a set of n vertical line segments in $\mathbb{R}^2$, does there exist a line ℓ that stabs at least half of these segments?

(iii) Given a set of n points in $\mathbb{R}^2$, compute the slab (region bounded by two parallel lines) of minimum vertical height that encloses all these points.

(iv) Given a set of n points in $\mathbb{R}^2$, compute the closest pair of points.

(c) (5 points) Given an n-element point set P in $\mathbb{R}^2$, we want to compute the largest circular disk that has its center in P's convex hull (or on its boundary) and contains no points of P in its interior. Describe a set of $O(n)$ points, computable in $O(n \log n)$ time, such that the optimal center lies at one of these points. (Briefly explain.)

(d) (5 points) In the randomized incremental algorithm for computing a trapezoidal map, when the ith segment is added, *up to constant factors*, the expected number of newly created trapezoids is (select one):

- 1
- i
- $\log i$
- n/i
- $\log n$
- None of these. What is it?

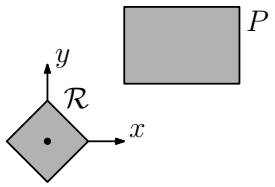


Figure 1: C-obstacle.

(e) (5 points) You are doing translational motion planning in $\mathbb{R}^2$. The obstacle P is an axis-aligned rectangle. The robot $\mathcal{R}$ is a 45° rotation of a square. What is the maximum number of edges possible in the C-obstacle $(P \oplus (-\mathcal{R}))$? (Briefly explain.)

Problem 2. (20 points) You are given a set $P = \{p_1, \ldots, p_n\}$ in $\mathbb{R}^2$. A *slab* $S(a, b^-, b^+)$ is the region between two pair of parallel lines, $S(a, b^-, b^+) = \{p : ap_x + b^- \leq p_y \leq ap_x + b^+\}$. The *height* of the slab is the vertical distance between the lines, $b^+ - b^-$.

Given a parameter k, where $0 \leq k \leq \frac{n-3}{2}$, we say the slab is *k-centered* if there are k points lying strictly above the slab and k points lying strictly below the slab (see Fig. 2). We are interested in the k-centered slab of minimum height.

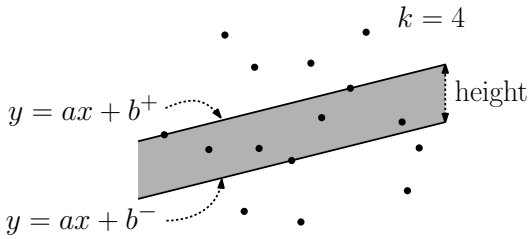


Figure 2: k-Centered slabs.

(a) (5 points) Assuming the standard dual transformation given in class $(a, b) \leftrightarrow y = ax - b$, describe the dual-equivalent formulation of this problem. That is, what is the slab? what is its height? what condition is satisfied for the slab be k-centered?

(b) (5 points) Assuming that P is in general position, prove that the minimum height k-centered slab has three points on its boundary, with two points on one line and one on the other.

(c) (10 points) Present an efficient algorithm, which given P and k $(0 \leq k \leq \frac{n-3}{2})$, computes the minimum-height k-centered slab. Derive your algorithm's running time and justify its correctness. (**Hint:** Plane sweep in the dual arrangement.)

Problem 3. (20 points) Explain how to use/modify range trees to answer the following queries. The input is an n-element point set P in $\mathbb{R}^2$.

(a) (10 points) A *right-triangle query* involves a region defined by a right triangle whose two legs are parallel to the coordinate axes and of *equal length*, such that the right angle is in the lower-left corner of the triangle. The query is defined by its lower left vertex $v = (v_x, v_y)$ and the length w of its two legs (see Fig. 3(a)). The answer

is the number of points that lie within the triangle. Present the data structure and query algorithm. Derive its space usage and the query time.

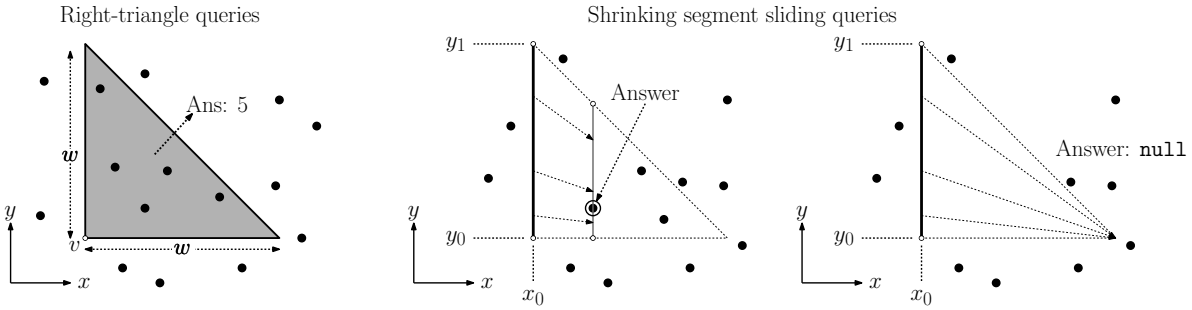


Figure 3: Range tree queries.

(b) (10 points) In a *shrinking segment-sliding query*, you are given a vertical line segment with x-coordinate x_0 and endpoints at y-coordinates y_0 and y_1, where $y_0 < y_1$. As this segment slides to the right it shrinks in height. The lower endpoint of the segment stays at $y = y_0$, but for each w units the segment slides horizontally, its height decreases by w. The answer to the query is the first point that is hit by the sliding-shrinking segment. If the segment slides so far that it shrinks to height zero, the answer is `null` (see Fig. 3(b)). Present the data structure and query algorithm. Derive its space usage and the query time.

Problem 4. (25 points) Consider the range space $\Sigma = (\mathbb{R}^2, \mathcal{T})$, where $\mathcal{T}$ is the set of all right triangles whose two legs are parallel to the coordinate axes, so that the right angle is in the lower-left corner of the triangle (see Fig. 4).

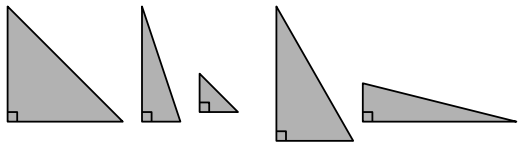


Figure 4: Set system of axis-aligned right triangles.

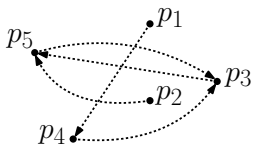
(a) (5 points) Give an example of a 4-element point set P in $\mathbb{R}^2$ that is shattered by Σ, and demonstrate why it is shattered. (For preciseness, indicate the coordinates of the points, but you can present a drawing to illustrate how to shatter them.)

(b) (12 points) Prove that no 5-element point set in $\mathbb{R}^2$ is shattered by Σ. (You can continue your answer on the top of the next page)

(c) (3 points) What is the VC-dimension of Σ? (No justification needed.)

(d) (5 points) Given a point set P in $\mathbb{R}^2$ with n points, give a (tight) asymptotic upper bound on the number of distinct subsets of P determined by Σ. (Using the notation given in class, this is $|\mathcal{T}_{|P}|$. No justification needed.)

Problem 5. (20 points) Given a set $P = \{p_1, \ldots, p_n\}$ of points in $\mathbb{R}^d$, for each $p_i \in P$ its *farthest neighbor* is the point $p_j \in P$ from P that is farthest from p_i.



i:	Index of p_i's farthest neighbor:
1	4
2	5
3	5
4	3
5	3

Figure 5: All farthest neighbors.

(a) (10 points) Prove that there must exist at least one pair of points $p_i, p_j \in P$ such that p_j is the farthest neighbor of p_i and vice versa. (For example, (p_3, p_5) above satisfies this.) Assume by general position that all inter-point distances are distinct.

(b) (10 points) Assuming that the points are in the plane, prove that the farthest neighbor of any point is a vertex of P's convex hull.

Problem 6: (15 points) Let $P = \{p_1, \ldots, p_n\}$ be a set of points in $\mathbb{R}^d$, and let Δ denote the diameter of P, that is, the distance between its farthest pair. We say that a pair $p_i, p_j \in P$ is *diametrical* if $\|p_j - p_i\| = \Delta$. (Let us ignore general position and imagine that there may be many diametrical pairs.)

Present an approximation algorithm that, given P, $\Delta = \text{diam}(P)$, and $\varepsilon > 0$, counts the number of pairs that are approximately diametrical. This means that for any pair $p_i, p_j \in P$:

- if $\|p_j - p_i\| = \Delta$ then the pair *must* be counted, and
- if $\|p_j - p_i\| < \Delta/(1+\varepsilon)$ then the pair *must not* be counted.

Otherwise, your algorithm is free to count or ignore the pair. Justify your algorithm's correctness and derive its running time. **Hint:** WSPDs. Your algorithm should run in time $O(n \log n + n/\varepsilon^d)$.