Homework 1: Hulls, Sweep, Triangulations, and More

Handed out Tuesday, Sep 13 (and updated Tuesday, Sep 20). Due at the start of class on Thursday, Sep 29. 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. Let $P = \{p_1, \ldots, p_n\}$ and $P' = \{p'_1, \ldots, p'_n\}$ be the vertex sets of two upper hulls in the plane (e.g., the output of Graham's algorithm). Each set is presented as a sequence of points sorted from left to right. Let $p_i = (x_i, y_i)$ and $p'_j = (x'_j, y'_j)$ denote the point coordinates. We assume that P lies entirely to the left of P', meaning that there exists a value z such that for all i and j, $x_i < z < x'_j$ (see Fig. 1).

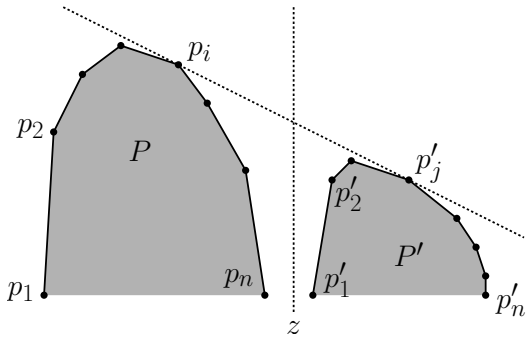

Figure 1: Problem 2: Computing the upper tangent of two hulls

Present an $O(\log n)$-time algorithm which, given P and P', computes the two points $p_i \in P$ and $p'_j \in P'$ such that the upper tangent of the two upper hulls (more formally, their common support line) passes through these two points.

Briefly justify your algorithm's correctness and derive its running time. (The correctness proof involves a case analysis. Please be careful, since a poorly drawn figure may lead you to an incorrect hypothesis.)

Problem 2. Consider a set $P = \{p_1, \ldots, p_n\}$ of points in the plane, where $p_i = (x_i, y_i)$. A *Pareto set* for P, denoted Pareto(P), (named after the Italian engineer and economist Vilfredo Pareto), is a subset of points p_i such that there is no $p_j \in P$ ($j \neq i$) such that $x_j \geq x_i$ and $y_j \geq y_i$. That is, each point of Pareto(P) has the property that there is no point of P that is both to the right and above it (see Fig. 2).

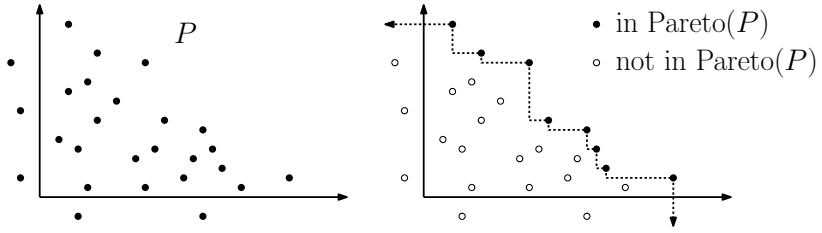

Figure 2: Problem 2: Pareto set

Pareto sets and convex hulls in the plane are similar in many respects. In this problem we will explore some of these connections.

(a) A point p lies on the convex hull of a set P if and only if there is a line passing through p such that all the points of P lie on one side of this line. Provide an analogous assertion for the points of Pareto(P) in terms of a different shape.

(b) Devise an analogue of Graham's convex-hull algorithm for computing *Pareto*(P) in $O(n \log n)$ time. Briefly justify your algorithm's correctness and derive its running time. (You do not need to explain the algorithm "from scratch", that is, you can explain what modifications would be made the Graham's algorithm.)

(c) Devise an analogue of the Jarvis march algorithm for computing *Pareto*(P) in $O(h \cdot n)$ time, where h is the cardinality of *Pareto*(P). (As in part (b), you can just explain the differences with Jarvis's algorithm.)

(d) Devise an algorithm for computing *Pareto*(P) in $O(n \log h)$ time, where h is the cardinality of *Pareto*(P). (I know of two solutions. One involves modifying Chan's algorithm. The other is conceptually simpler, but harder to analyze. Any correct algorithm that achieves the desired running time will receive full credit.)

In each case, briefly justify the correctness of your algorithm, explain any data structures that you make use of (unless they are standard), and briefly derive your algorithm's running time.

Problem 3. You are given as input a collection of n circles $\{C_1, \ldots, C_n\}$ in $\mathbb{R}^2$, where circle C_i is presented as its center point $q_i = (x_i, y_i)$ and radius $r_i > 0$. Present an $O(n \log n)$ time algorithm that determines whether any two circles intersect. Note that one circle may be nested within another without intersecting (see Fig. 3). Your algorithm should either output that there is no intersection, or that there is at least one intersection, and if so it will output the indices of i and j of two circles C_i and C_j that intersect. Irrespective of the number of intersecting pairs, it need only output *one* intersecting pair.

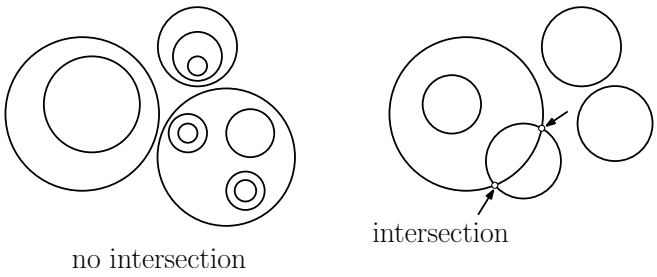

Figure 3: Problem 3: Intersecting circles

Hint: Use plane-sweep. Explain clearly (1) what the sweep-line status stores and what data structure is used to store this information and (2) what future events are stored and what data structure is used. You may assume that you have access to whatever *primitive operations* that you need in constant time. For example, if you want to determine (a) whether two circles intersect, (b) the coordinates of an intersection, (c) the intersection of a line with a circle, (d) whether a point is contained within a circle's interior, etc., you may simply *assume* the existence of a function that runs in $O(1)$ time. As always, you may make whatever general-position assumptions you like.

Problem 4. A friend of yours from the civil engineering department wants to analyze whether a dangerous portion of a river will flood. He presents you with the following (admittedly rather unrealistic) model of the river. The portion of the river of interest is modeled as an x-monotone polygon P that is bounded between two vertical lines at $x = x^-$ and $x = x^+$ (see Fig. 4). The river is bounded on its left and right ends by two vertical line segments of lengths w^- and w^+, respectively. Inside the polygon are

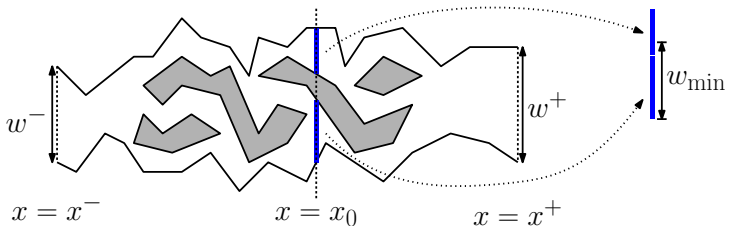

Figure 4: Problem 4: River flooding

some number of disjoint x-monotone polygons that represent islands in the river. Let n denote the total number of vertices, including both the outer banks of the river and the islands.

Your friend tells you that the in order to avoid a flood, the width of the river (not counting islands) at *every* vertical cut must be at least some minimum value $w_{\min}$. For example, in the figure, the sum of the two blue vertical segments at $x = x_0$ must be at least $w_{\min}$ in order to avoid a flood.

Given the polygon P and the value $w_{\min}$, present an $O(n \log n)$ time algorithm that determines whether the river will flood, that is, whether there is a vertical cut whose total width is smaller than $w_{\min}$. If it will flood, your algorithm should output the value x_0 of the *bottleneck*, that is, the location where the sum of vertical lengths (excluding islands) is the smallest.

Hint 1: There is an (uncountably) infinite number of possible vertical cuts to consider. Prove that it suffices to check the width at a discrete set of locations, whose number is $O(n)$.

Hint 2: There is a bit of a trick to updating the vertical widths (excluding the islands). For partial credit, explain how to do it under the assumption that the sweep line can only intersect a constant number of islands at any time. (For example, in the figure, the sweep line never hits more than two islands at a time.) For full credit, explain how to do it even if the number of islands hit by the sweep line at any time could be as high as $\Omega(n)$.

Challenge Problem. Challenge problems count for extra credit points. These additional points are factored in only after the final cutoffs have been set, and can only increase your final grade.

Let P be a simple polygon in the plane, and consider any triangulation of P (without Steiner points). Recall from class that the *dual graph* of the triangulation is an undirected graph whose nodes are the triangles, and two nodes are adjacent of the corresponding triangles are adjacent. In class we showed (informally) that the dual graph of a triangulation of a simple polygon is a tree, that is, a connected, acyclic graph.

Observe that the nodes of the dual graph of a triangulation of a simple polygon are of degree either one, two, or three. We say that a tree is a *vine* if all of its nodes are of either degree one or two—there are no nodes of degree three (see Fig. 5). A triangulation is called a *vine* if its dual graph is a vine.

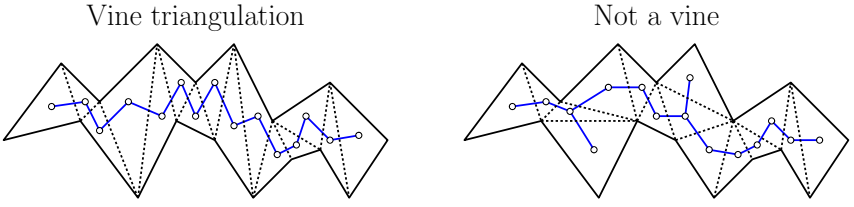

Figure 5: Challenge Problem: Vine triangulation

(a) True or False: If P is an x-monotone simple polygon, the triangulation generated by the algorithm given in class is always a vine. (If false, present a concise counterexample. If true, present a proof.)

(b) Only if you answered False to part (a): True or False: Every x-monotone simple polygon has a triangulation that is a vine. (If false, present a concise counterexample. If true, present a proof.)

(c) Only if you answered False to both parts (a) and (b): Present a polynomial-time algorithm that, given an x-monotone simple polygon, determines whether it has a triangulation that is a vine.

Some tips about writing algorithms: Henceforth, whenever you are asked to present an "algorithm," you should present three things: the algorithm, an informal proof of its correctness, and a derivation of its running time. Remember that your description is intended to be read by a human, not a compiler, so conciseness and clarity are preferred over technical details. Unless otherwise stated, you may use any results from class, or results from any standard textbook on algorithms and data structures. Also, you may use results from geometry that: (1) have been mentioned in class, (2) would be known to someone who knows basic geometry or linear algebra, or (3) is intuitively obvious. If you are unsure, please feel free to check with me.

Giving careful and rigorous proofs can be quite cumbersome in geometry, and so you are encouraged to use intuition and give illustrations whenever appropriate. Beware, however, that a poorly drawn figure can make certain erroneous hypotheses appear to be "obviously correct."

Throughout the semester, unless otherwise stated, you may assume that input objects are in *general position*. For example, you may assume that no two points have the same x-coordinate, no three points are collinear, no four points are cocircular. Also, unless otherwise stated, you may assume that any geometric primitive involving a constant number of objects each of constant complexity can be computed in $O(1)$ time.

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

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

Homework 2: Linear Programming, Point-Location and More

Handed out Thursday, Oct 20. Due at the start of class on Thursday, Oct 27. Late homeworks are not accepted (unless an extension has been prearranged) so please turn in whatever you have completed by the due date. Unless otherwise specified, you may assume that all inputs are in *general position*. Whenever asked to give an algorithm running in $O(f(n))$ time, you may give a *randomized algorithm* whose expected running time is $O(f(n))$.

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

(a) You are given two point sets $P = \{p_1, \ldots, p_n\}$ and $Q = \{q_1, \ldots, q_m\}$ in the plane, and you are told that they are separated by a vertical line $x = x_0$, with P to the left and Q to the right (see Fig. 1(a)). Compute the line equations of the two "crossing tangents," that is, the lines ℓ_1 and ℓ_2 that are both supporting lines for $\text{conv}(P)$ and $\text{conv}(Q)$ such that P lies below ℓ_1 and above ℓ_2 and the reverse holds for Q. (Note that you are *not* given the hulls, just the point sets.) Your algorithm should run in time $O(n + m)$.

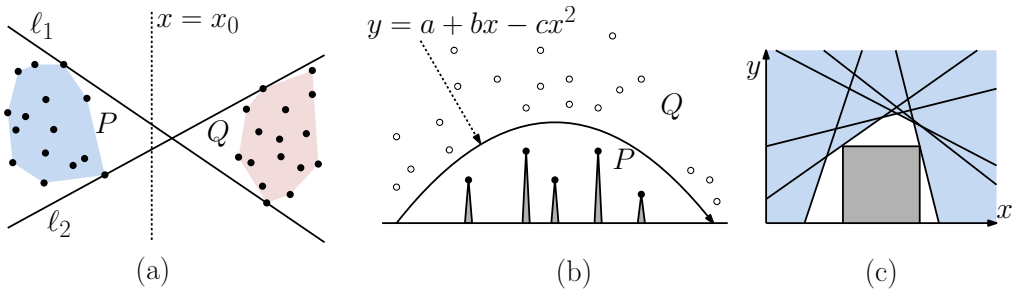

Figure 1: Problem 1: LP applications

(b) You have a cannon in $\mathbb{R}^2$. It has three controls labeled "a," "b," and "c". A projectile shot from this cannon travels along the parabolic arc $y = a + bx - cx^2$. You are asked to determine whether it is possible to adjust the controls so that the projectile travels above a set of n building tops, represented by a point set $P = \{p_1, \ldots, p_n\}$ and beneath a set of m floating balloons, represented by a point set $Q = \{q_1, \ldots, q_m\}$ (see Fig. 1(b)). Your algorithm should run in time $O(n + m)$. (I do not care where the cannon is actually located. If your solution is based on some assumption about the cannon's location, please state this.)

(c) You are given a set of n halfplanes $H = \{h_1, \ldots, h_n\}$, where h_i is given as a pair (a_i, b_i) and it consists of all the points of the plane that lie on or beneath the line $y = a_i x + b_i$. Compute the axis-parallel square of the largest side length whose lower edge lies on the x-axis (see Fig. 1(c)). If no such square exists, your algorithm should indicate this.

Problem 2. The objective of this problem is to get some practice working with backwards analysis. Consider the following randomized incremental algorithm for the Pareto-set problem. Recall that the input is a set of n points $P = \{p_1, \ldots, p_n\}$ in the plane, where $p_i = (x_i, y_i)$, and the objective is to compute the subset of points p_i such that there is no $p_j \in P$ $(j \neq i)$ such that $x_j \geq x_i$ and $y_j \geq y_i$. Our approach will be to add the points one-by-one in random order. We make the usual general-position assumption that there are no duplicate x- or y-coordinates.

The algorithm begins by randomly permuting the point set. Let $P = \langle p_1, \ldots, p_n \rangle$ denote the permuted sequence. Let's add two sentinel points $p_{-1} = (-\infty, +\infty)$ and $p_0 = (+\infty, -\infty)$. The initial Pareto set Q_0 consists of the pair $\langle p_{-1}, p_0 \rangle$. Observe that as we add points to the Pareto set, p_{-1} will always be the leftmost point, and p_0 will always be the rightmost. For $1 \le i \le n$, let Q_i denote the sequence of points of the Pareto set after inserting $\langle p_{-1}, \ldots, p_i \rangle$ sorted from left to right. The final output will be Q_n. Also, let $R_i = \langle p_{i+1}, \ldots, p_n \rangle$ denote the sequence of points that *remain* to be inserted.

Let us assume inductively that we have already inserted $i-1$ points, and we are about to insert p_i. In order to make it easy to determine where to insert the new point within the existing Pareto sequence, we will store all the remaining points of R_{i-1} in a collection of *buckets*, one for each point in the current Pareto set. In particular, let us suppose that we have already inserted points $i-1$ points into the Pareto set. Let $Q_{i-1} = \langle p_{j_0}, p_{j_1}, \ldots, p_{j_m} \rangle$. (By our earlier observation, $j_0 = -1$ and $j_m = 0$.) For $1 \le k \le m$, define $B(k)$ to be the subset of remaining points that lie within the vertical strip between the $(k-1)$-st and kth points of the current Pareto sequence, that is,

$$B(k) \;=\; \{p_\ell \in R_{i-1} \;:\; x_{j_{k-1}} < x_\ell < x_{j_k}\}.$$

(For example, in Fig. 2 the white points in the shaded region belong to the bucket $B(3)$ because they lie between $x_{j_2} = x_1$ and $x_{j_3} = x_4$.)

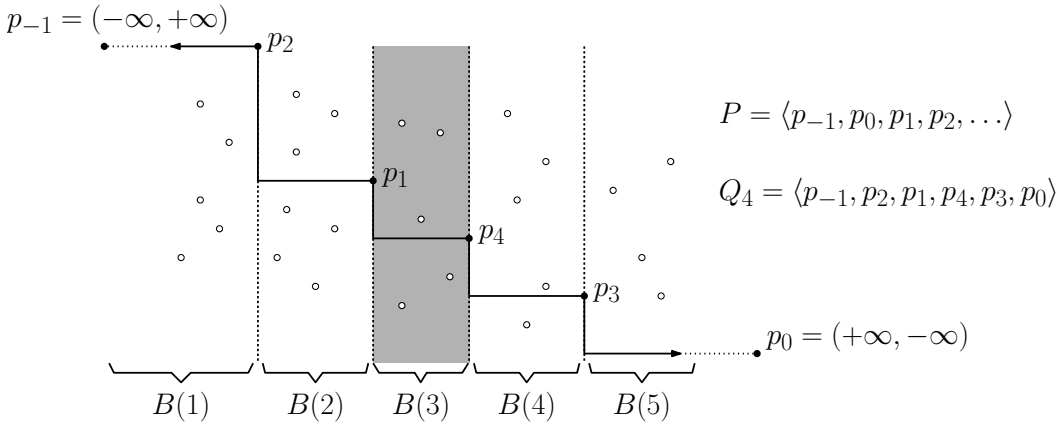

Figure 2: Problem 2: Pareto Set

(a) Using the above structure, explain how to insert the ith point p_i and update the current Pareto set Q_{i-1} to form Q_i. Also explain how to update the associated buckets. (If points are removed from the Pareto set, then their associated bucket sets must be redistributed among the points that are currently in the Pareto set.)

(b) Analyze the running time of your algorithm. Your analysis should include two elements (i) the time needed to update the Pareto set, and (ii) the time needed to update the associated bucket sets.

Problem 3. This is a query variant of problem 1(b). You are given two sets $P = \{p_1, \ldots, p_n\}$ and $Q = \{q_1, \ldots, q_m\}$, which you may assume all lie in the positive x, y-quadrant. The cannon is located at the origin, and its projectile function is restricted so that $a = 0$. In other words given b and c, the projectile travels along the curve $y = bx - cx^2$. The objective is to preprocess the sets P and Q so that the following queries can be answered efficiently. Given a pair (b, c), determine whether the projectile passes above all the points of P and below all the points of Q. If not, your query algorithm should report the leftmost violation, that is, the leftmost point of P that the projectile passes below or the leftmost point of Q that the projectile passes above. (For example, in Fig. 3 the answer would be q_4.)

Let $N = n + m$ be the total input size. The preprocessing time for your data structure should be at most $O(N \log N)$, your data structure should use space $O(N)$, and queries should be answered in time $O(\log N)$. Partial credit will be given if the space or preprocessing time is a bit higher, but the query time must be $O(\log N)$.

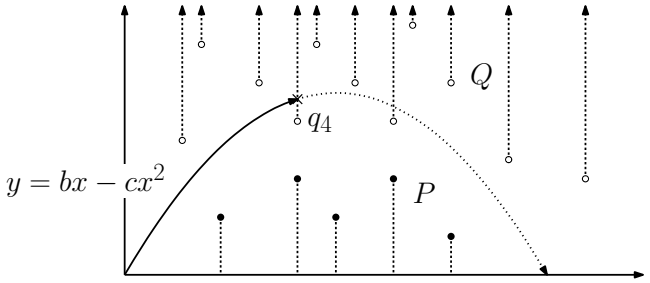

Figure 3: Problem 3: Projectile queries

Problem 4. In Homework 1, we considered a plane-sweep algorithm for determining whether there is any intersection among a collection of n circles in the plane. Here we consider a variant of this problem. The input consists of a collection of n closed circular disks, all having the same radius. (Via scaling, we may assume that they are all unit disks.) Let $C = \{c_1, \ldots, c_n\}$ denote the center points of these disks, and let $\{D_1, \ldots, D_n\}$ denote the actual disks. Thus, D_i consists of the points that lie within unit distance of c_i. Let $U = D_1 \cup \cdots \cup D_n$ denote the union of these disks. The boundary of U may generally consist of multiple parts, each of which consists of a cycle of circular arcs connected by vertices. (In Fig. 4 the boundary consists of three cycles. The vertices are shown as white dots).

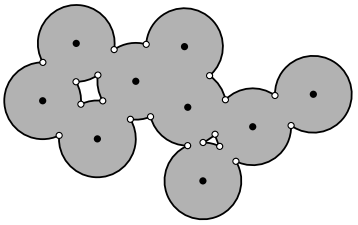

Figure 4: Problem 4: The union of unit disks.

(a) Present an algorithm that reports all the vertices on the boundary of U. (Note that circle intersection points in the interior of the union are explicitly excluded.) Your algorithm should run in time $O(n \log n)$. The order in which the vertices are output is arbitrary. (Hint: Don't try to modify the algorithm from Homework 1. A different approach is needed.)

(b) Prove that the number of vertices reported by your algorithm is $O(n)$.

Challenge Problem. Challenge problems count for extra credit points. These additional points are factored in only after the final cutoffs have been set, and can only increase your final grade.

(a) (Extending problem 4(b)) Prove that the number of vertices on the boundary of the union of n unit disks in the plane is at most $6n$.

(b) Show that the multiplicative factor of 6 in part (a) is essentially tight in the limit by demonstrating for any $\varepsilon > 0$, there exists a value of n (depending on ε) and a configuration of n unit disks in the plane such that the number of vertices on the boundary of their union is at least $(6 - \varepsilon)n$. (Hint: There is a repeating pattern of disks, and as the pattern gets bigger and bigger, the number of vertices gets closer to $6n$.)

(c) Consider a set of n disks (of arbitrary radii) in the plane, with the restriction that no three disks have a common point of intersection. Prove that the number of vertices on the boundary of their union is $O(n)$. (By the way, the $O(n)$ bound holds even without the intersection restriction, but the proof is more complex.)

CMSC 754:Fall 2016 Dave Mount

Homework 3: Arrangements and Approximation

Handed out Saturday, Dec 3. Due at the start of class on Thursday, Dec 8. 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. You are given three sets of points R, G, and B (red, green, and blue) in $\mathbb{R}^2$. A *tricolor strip* is a pair of parallel lines such that the closed region bounded between these two lines contains exactly three points, one from each of R, G, and B. Define the strip's *height* to be the vertical distance between these lines (see Fig. 1).

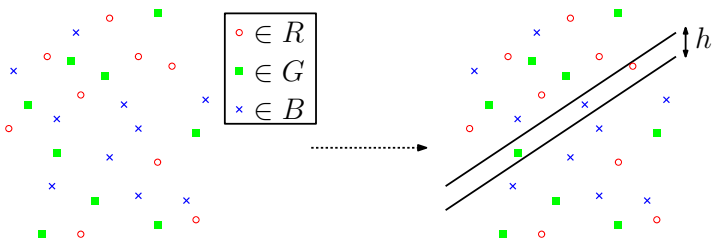

Figure 1: Problem 1: Tricolor strip of height h.

(a) Explain what a tricolor strip of height h corresponds to in the dual plane.

(b) If a tricolor strip is of minimum height, what additional conditions must be satisfied? Explain briefly.

(c) Present an algorithm, which given inputs R, G, and B, computes the minimum height tricolor strip. Your algorithm should run in time $O(n^2)$, where $n = |R| + |G| + |B|$. (If you like, you may express your answer as a plane sweep through a line arrangement running in $O(n^2 \log n)$ time, and then appeal to the fact that it can be easily converted to a topological plane sweep algorithm running in $O(n^2)$ time.)

Problem 2. The maximum complexity of the k-th level of an arrangement of n lines in the plane is an open problem in combinatorial geometry. (The best upper bound is $O(nk^{1/3})$ and the best lower bound is $n \cdot e^{\Omega(\sqrt{\log k})}$.) You have been asked to write a program to compute the complexities of arrangements as part of an empirical study.

Present an algorithm that is given a set $L = \{\ell_1, \ldots, \ell_n\}$ of lines in the plane, and outputs an array $C[1..n]$, such that $C[i]$ is the number of edges on the ith level of the arrangement $\mathcal{A}(L)$. Your algorithm should run in time $O(n^2)$. (As in Problem 1, if you like, you may express your algorithm as a plane-sweep algorithm and appeal to topological plane sweep.)

Problem 3. 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. You are given a set P of n points lying in the unit square in $\mathbb{R}^2$. The objective is to reduce P to a smaller set having roughly the same convex hull. Given any subset $Q \subseteq P$, clearly we have $\text{conv}(Q) \subseteq \text{conv}(P)$. For $\varepsilon > 0$, we say that Q is an ε-sketch of P if every point of P lies within distance at most ε of $\text{conv}(Q)$ (see Fig. 2(a)).

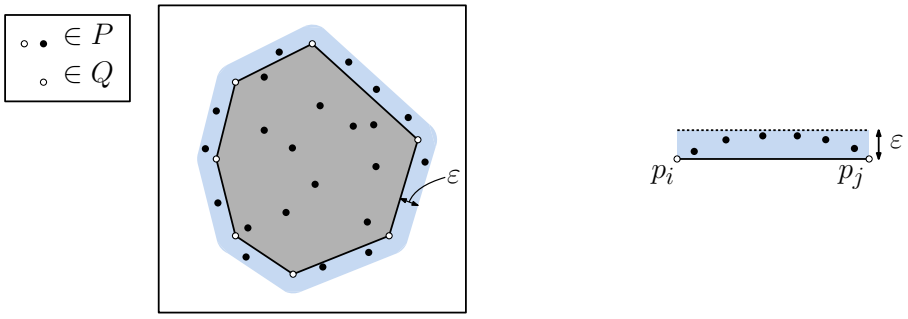

Figure 2: Problem 4: Computing an ε-sketch.

(a) Consider the following simple greedy algorithm for computing an ε-sketch of a planar point set P. First, let $\langle p_0, \ldots, p_{k-1} \rangle$ denote the vertices of $\text{conv}(P)$ listed in counterclockwise order. Put p_0 in Q and set $i \leftarrow 0$. Find the largest index j, where $i < j \leq k$ such that all the points $\{p_{i+1}, \ldots, p_{j-1}\}$ lie within distance ε of the line segment $\overline{p_i p_j}$ (see Fig. 2(b)). (Indices are taken modulo k, so $p_k = p_0$.) Add if $j = k$, then stop. Otherwise, add p_j to Q, set $i \leftarrow j$, and repeat.

Show that this procedure correctly produces an ε-sketch of P.

(b) Let $m(P)$ be the minimum number of points on any ε-sketch of P. Assume that the points of P are in convex position (that is, they all lie on $\text{conv}(P)$). Show that this procedure produces a sketch of size at most $m(P) + 1$. (Hint: First, show that the points in any minimum sketch can

be assumed to be taken from the vertices of the convex hull of P. Second, consider the points of the minimum sketch in cyclic order about P's convex hull and ask how many points generated by the greedy algorithm can be created between any two points of the minimum sketch.)

(c) Prove that no matter how many vertices there are on P's convex hull, $m(P) = O(1/\varepsilon)$.

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 1. Modify your algorithm for Problem 1, so that instead of minimizing the vertical width between the segments, it minimizes the *perpendicular width* between the segments. The running time should still be $O(n^2)$.

Challenge Problem 2. Modify your solution to Problem 3(c), so that instead of computing an ε-approximation to the average bichromatic distance, it computes an ε-approximation to the *standard deviation* of the bichromatic distances. The running time should still be $O(n \log n + n/\varepsilon^d)$.

Challenge Problem 3. Strengthen 4(c) to show that there exists an ε-sketch of size $O(1/\sqrt{\varepsilon})$.

(Hint: Traverse the vertices of the hull in counterclockwise order starting at the bottommost vertex of the hull. Each vertex visited can be associated with two real numbers: (1) the distance traveled along the boundary of the hull to this vertex, and (2) the angle formed between the x-axis and the edge directed out from this vertex. Both values increase monotonically throughout the traversal. First, show that if both of these values for consecutively selected points differ by at most $c\sqrt{\varepsilon}$, for an appropriately chosen constant c, the resulting points will form an ε-sketch. Next, show that it is possible to select $O(1/\sqrt{\varepsilon})$ points satisfying this property.)

Challenge Problem 4. Consider problem 4(b) in the case where the points of P are not in convex position. Note that the optimum sketch is allowed to use points that are not on $\text{conv}(P)$, but the greedy algorithm is still required to take points only from $\text{conv}(P)$. What can be said about the number of points generated by the greedy algorithm compared to the optimum $m(P)$? (Is it $m(P) + 1$? $m(P) + O(1)$? $c \cdot m(P)$ for some constant c?)

Mini-Homework 4: Range Spaces and Motion Planning

Handed out Thursday, Dec 1. Due at the start of class on Thursday, Dec 8. 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.

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.

In the following parts, let τ denote the triangle whose vertices are $(0, 0)$, $(1, 0)$, and $(0, 1)$ (see Fig. 1(a)).

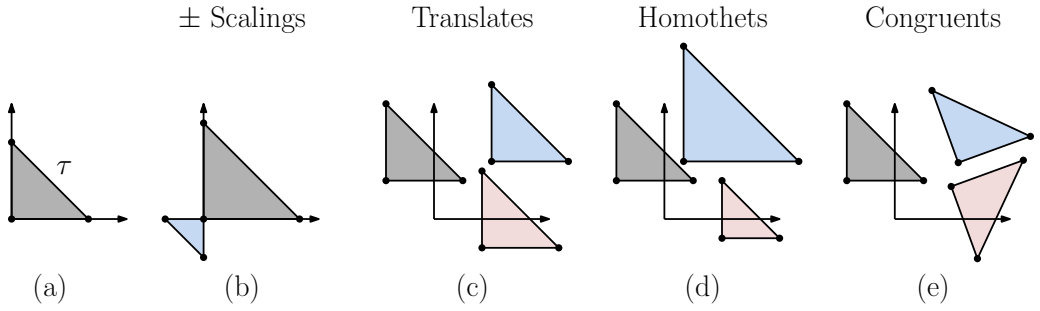

Figure 1: Problem 1: VC-dimension.

(a) A *uniform ±-scaling* of τ is any triangle with vertices $(0, 0)$, $(\alpha, 0)$ and $(0, \alpha)$ for any $\alpha \in \mathbb{R}$, possibly negative (see Fig. 1(b)). What is the VC-dimension of $\Sigma_1 = (\mathbb{R}^2, \mathcal{S})$, where $\mathcal{S}$ is the set of all uniform ±-scalings of τ?

(b) What is the VC-dimension of $\Sigma_2 = (\mathbb{R}^2, \mathcal{T})$, where $\mathcal{T}$ is the set of all translates of τ (see Fig. 1(c))?

(c) A shape τ' is a *homothet* of τ if τ' can be formed by performing a uniform scaling of τ by a any positive scale factor followed by any translation (see Fig. 1(d)). What is the VC-dimension of $\Sigma_3 = (\mathbb{R}^2, \mathcal{H})$, where $\mathcal{H}$ is the set of all homothets of τ?

(d) **Challenge Problem:** A shape τ' is *congruent* to τ if τ' can be formed by performing a rotation, translation, and reflection of τ, but no scaling (see Fig. 1(e)). What is the VC-dimension of $\Sigma_4 = (\mathbb{R}^2, \mathcal{C})$, where $\mathcal{C}$ is the set of all shapes that are congruent to τ? (Hint: If you cannot obtain the exact dimension, I would be satisfied with upper and/or lower bounds.)

Problem 2. Let us consider a motion planning problem in the plane, where the ground is x-axis. Consider a robotic crane, whose base is anchored at the origin. The crane can stretch vertically up to any height $h \geq 0$ above the x-axis, and it can extend horizontally at its highest point to the right of the y-axis by any distance $w \geq 0$. There is a hook dangling down at distance 1 from the tip of the crane. Defining the points $p_0 = (0, 0)$, $p_1 = (0, h)$, $p_2 = (w, h)$ and $p_3 = (w, h - 1)$, we require that none of the three line segments $\overline{p_0p_1}$, $\overline{p_1p_2}$, $\overline{p_2p_3}$ intersects any obstacle in the robot's workspace.

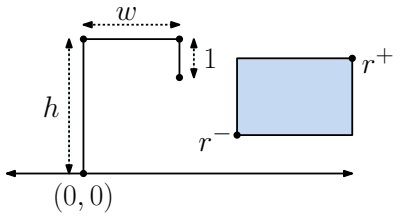

Figure 2: Problem 2: Robotic crane.

Suppose that you are given a workspace consisting of n disjoint axis-parallel rectangular obstacles $\mathcal{R} = \{R_1, \ldots, R_n\}$, where the ith rectangle is defined by its lower-left corner r_i^- and its upper right corner r_i^+. You may assume that all these rectangles lie above the x-axis, but they may lie on either side of (or overlap) the y-axis.

(a) Given a rectangle $r^- = (x^-, y^-)$ and $r^+ = (x^+, y^+)$, describe the shape of the resulting C-obstacle in the (w, h) configuration space of the crane. (Hint: There will be a few cases depending on whether the rectangle lies to the left, right, or overlaps the y-axis.)

(b) Given the set $\mathcal{R}$ of rectangles, and given starting and target configurations $s = (w_s, h_s)$ and $t = (w_t, h_t)$, sketch an algorithm for determining whether there is a collision-free motion of the crane between these configurations. (Hint: A high-level sketch of the algorithm is sufficient. To make your life simpler, you may assume that you are given a procedure that will input the C-obstacles from part (a), compute the union of these C-obstacles, and returns a convenient decomposition of free-space (e.g., as a trapezoidal map).)

Sample Problems for the Midterm Exam

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.

The exam will be this Tuesday, Nov 8 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.

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

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

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

(e) Any triangulation of any n-sided simple polygon has exactly $n - 2$ triangles. Suppose that the polygon has h polygonal holes each having k sides. (In Fig. 1, $n = 10$, $h = 2$, and $k = 4$). As a function of n, h and k, how many triangles will such a triangulation have? Explain briefly.

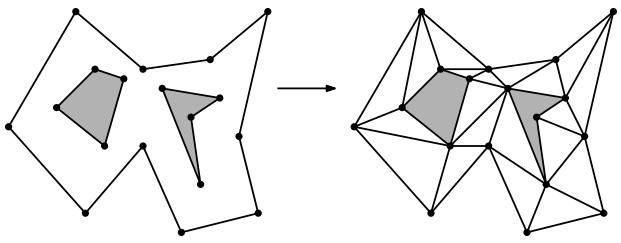

Figure 1: Problem 1(e).

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

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

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

(i) The DT is a t-spanner, for some constant t.

(ii) The Euclidean minimum spanning tree of P is a subgraph of the DT.

(iii) Among all triangulations of P, the DT maximizes the minimum angle.

(iv) Among all triangulations of P, the DT minimizes the maximum angle.

(v) Among all triangulations of P, the DT minimizes the total sum of edge lengths.

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)).

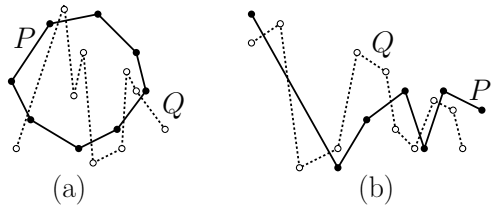

Figure 2: Problem 2.

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

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

(1) Let $P = \{p_1, p_2, \ldots, p_n\}$ be a random permutation of the points.

(2) Let $\text{low}[x] = \text{high}[x] = p_1[x]$. 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 4. You are given a set of n vertical line segments in the plane. Present an efficient an algorithm to determine whether there exists a line that intersects all of these segments (see Fig. 3). Such a line is called a *transversal*. (Hint: $O(n)$ time is possible.) Justify your algorithm's correctness and derive its running time.

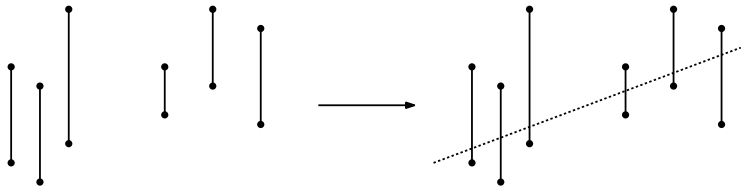

Figure 3: Problem 5.

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

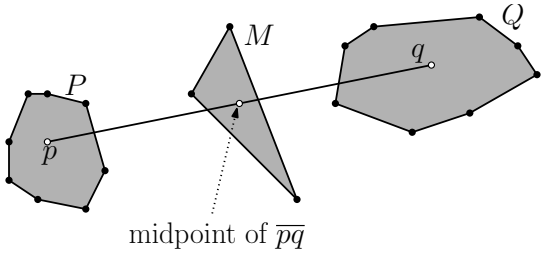

Figure 4: Problem 5.

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

Problem 7. You are given two sets of points in the plane, the red set R containing n_r points and the blue set B containing n_b points. The total number of points in both sets is $n = n_r + n_b$.

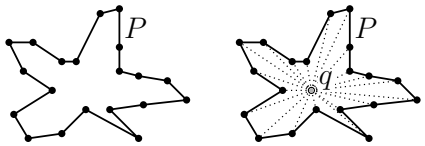

Figure 5: Problem 6.

Give an $O(n)$ time algorithm to determine whether the convex hull of the red set intersects the convex hull of the blue set. If one hull is nested within the other, then we consider them to intersect.

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

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

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

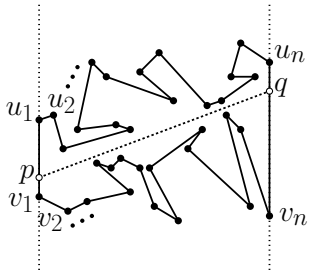

Figure 6: Problem 8.

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

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

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

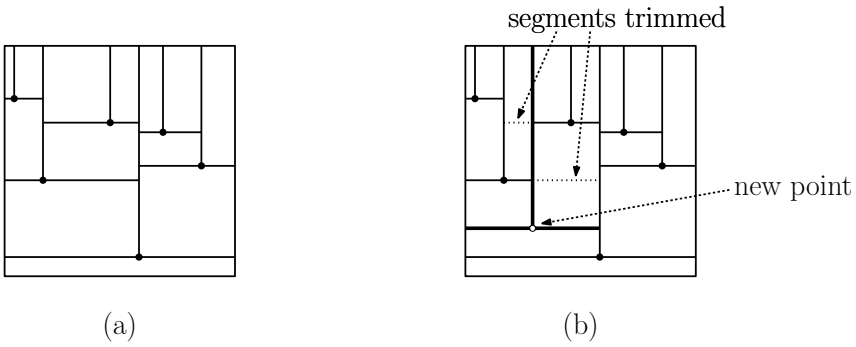

Figure 7: Problem 9.

(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)).

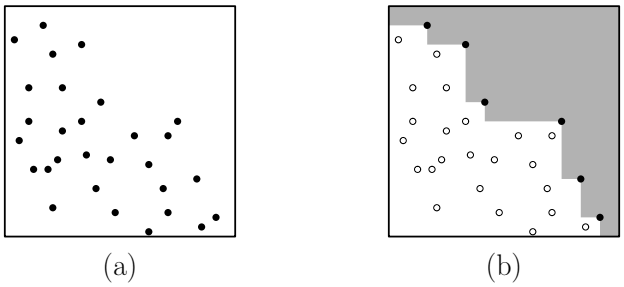

Figure 8: Problem 10.

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

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

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

Problem 11. Consider an n-sided simple polygon P in the plane. Let us suppose that the leftmost edge of P is vertical (see Fig. 9(a)). Let e denote this edge. Explain how to construct a data structure to answer the following queries in $O(\log n)$ time with $O(n)$ space. Given a ray r whose origin lies on e and which is directed into the interior of P, find the first edge of P that this ray hits. For example, in the figure below the query for ray r should report edge f. (Hint: Reduce this to a point location query in an appropriate planar subdivision.)

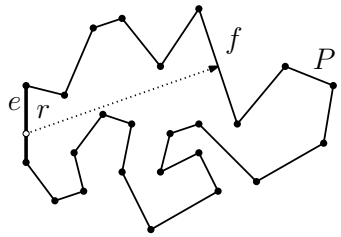

Figure 9: Problem 11.

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

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

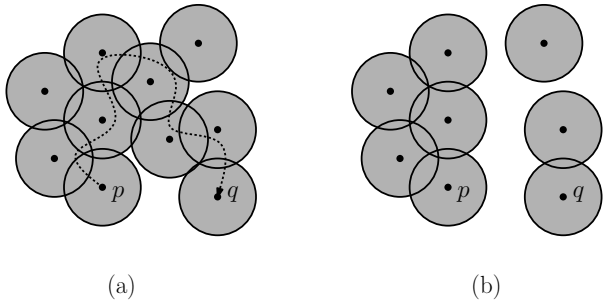

Figure 10: Problem 12.

Problem 13. In class we argued that the number of parabolic arcs along the beach line in Fortune's algorithm is at most $2n - 1$. The goal of this problem is to prove this result in a somewhat more general setting.

Consider the beach line at some stage of the computation, and let $\{p_1, \ldots, p_n\}$ denote the sites that have been processed up to this point in time. Label each arc of the beach line with its the associated site. Reading the labels from left to right defines a string. (In Fig. 11 below the string would be $p_2p_1p_2p_5p_7p_9p_{10}$.)

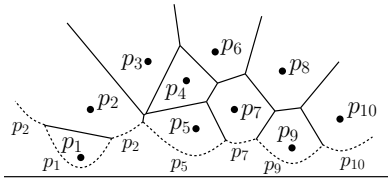

Figure 11: Problem 13.

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

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

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

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

CMSC 754: Midterm Exam

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

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

Problem 1. (25 points; 5–8 points each) Give a short answer (a few sentences) to each question.

(a) You are given two point sets P_1 and P_2, where the points of P_1 lie strictly to the left of the y-axis and the points of P_2 lie strictly to the right of the y-axis. Let ℓ_1 and ℓ_2 denote the upper and lower tangents of the convex hulls of these two point sets (see Fig. 1). Assuming the dual transformation given in class (that maps point (a, b) to the line $y = ax - b$), what do the duals of the lines ℓ_1^* and ℓ_2^* represent with respect to the duals of the point sets P_1^* and P_2^*? (A drawing may help illustrate your answer.)

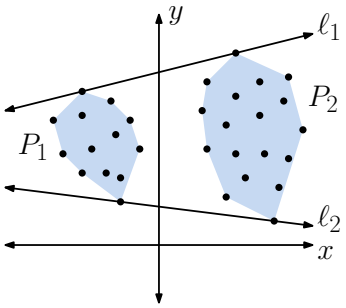

Figure 1: Problem 1(a).

(b) A trapezoidal map of n segments has roughly $6n$ vertices and roughly $3n$ trapezoids. Explain (e.g., via a charging argument) where the numbers 6 and 3 come from.

(c) True or False: There exists a set of line segments and a (very nasty) insertion order for these segments so that the depth of the point-location data structure described in class is $\Omega(n)$ (Explain briefly.)

(d) Give a short justification of the following claim: Given a set of n points in the plane $P = \langle p_1, \ldots, p_n \rangle$ that have been sorted by their x-coordinates, the maximum slope determined by any two points of the set will be achieved by a pair of consecutive points.

Problem 2. (25 points) 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. 2). 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. 2 the maximum depth is 3, as realized by the small triangular face in the middle.)

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 $O(n \log n + I \log n)$ time plane-sweep algorithm to determine the maximum depth.

Answer each of the following questions. Keep your answers *brief* (a few sentences each is sufficient).

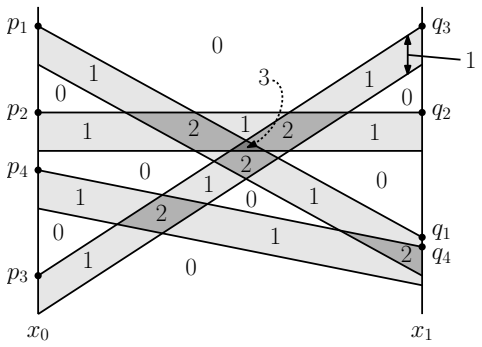

Figure 2: Problem 2: The numbers indicate the depth values.

(a) (5 points) What information is stored in the sweep-line status?

(b) (5 points) How is the sweep-line status initialized (for $x = x_0$)?

(c) (5 points) What is the initial set of events? How are these initial events computed? How are events stored?

(d) (5 points) How is each event processed?

(e) (5 points) Briefly justify the $O(n \log n + I \log n)$ running time.

Problem 3. (25 points) You are given a set U of n_1 upper halfplanes and a set L of n_2 lower halfplanes (see Fig. 3(a)). Let P denote their (possibly unbounded) intersection (see Fig. 3(b)). Let's assume that the ith element of U is given by the inequality $y \geq a_i x + b_i$ and the jth element of L is given by $y \leq c_j x + d_j$. Define the *maximum enclosed square*, $\text{MES}(U, L)$, to be the *axis-aligned* square of maximum side length that is enclosed within P (see Fig. 3(c)).

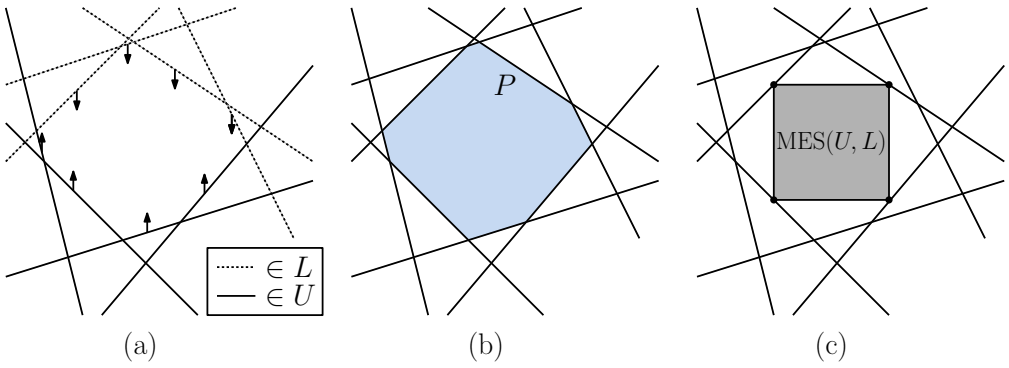

Figure 3: Problem 3: Maximum enclosed square.

(a) (10 points) Show that problem of computing $\text{MES}(U, L)$ can be expressed as a linear programming problem in a space of constant dimension.

What are the variables? What are the constraints? What is the objective function? (Hint: Any axis-aligned square can be defined by three numbers, the (x, y) coordinates of its lower left corner, and its side length s.)

(b) (2 points) Letting $n = n_1 + n_2$ denote the total number of halfplanes, what is the running time needed to solve this LP?

(c) (3 points) Can your LP be infeasible? Unbounded? Explain briefly why or why not.

(d) (10 points) Assuming *general position* and that the solution is *feasible* (neither infeasible nor unbounded), what can be said about the number of vertices of MES(U, L) that lie on P's boundary? (Only one? Two? Three? All four?) Briefly justify your answer. (I don't require a formal proof.)

Problem 4. (25 points) This problem arises from computational biology. You are given a protein molecule P that consists of n atoms. Each atom is represented by a circular disk in the plane of radius r (see Fig. 4(b)). Let $P = \{p_1, \ldots, p_n\}$ denote the centers of these atoms.

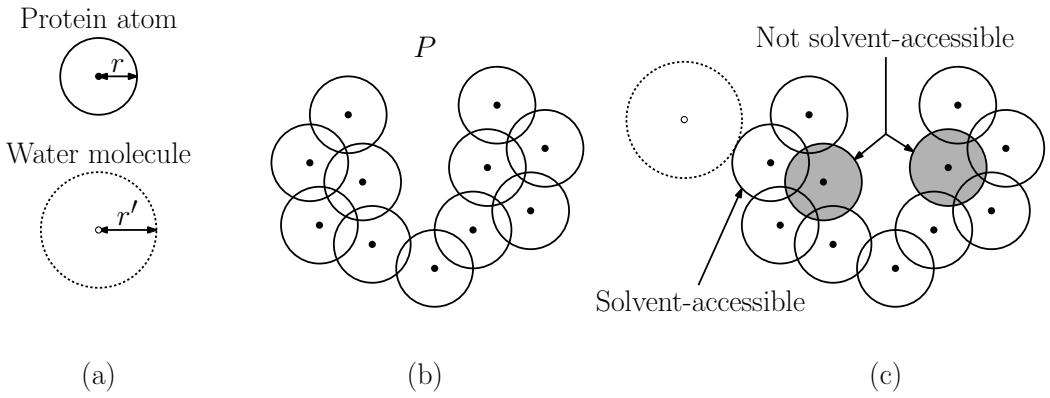

Figure 4: Problem 4: Solvent-accessibility.

The protein molecule lives in a solution of water. A water molecule is represented by a circular disk of radius $r' > r$ (see Fig. 4(a)). Water molecule and protein molecules *cannot* overlap. We say that an atom of P is *solvent-accessible* if there exists a placement of a water molecule that touches this atom, but does *not* intersect any of the other atoms of P. (In Fig. 4(c) only the two shaded atoms are *not* solvent-accessible.)

(a) (10 points) Consider the Voronoi diagram of the center points of the atoms of P. Show that an atom $p_i \in P$ is solvent-accessible if and only if there exists a point within p_i's Voronoi cell that is at distance at least $r + r'$ from p_i.

(b) (5 points) Show that in any (bounded) Voronoi cell, the point of the cell that is *farthest* from the cell's site is a vertex of this cell.

(c) (10 points) Using (a) and (b), show that it is possible to determine all the atoms that are solvent-accessible in $O(n \log n)$ time.

Sample Problems for the Final Exam

The final exam will be **Sat, Dec 17, 10:30am-12:30pm**. The exam will be closed-book and closed-notes. You may use *two* sheets of notes (front and back). The following problems have been collected from old homeworks and exams. They do not necessarily reflect the actual difficulty or coverage of questions on the final exam. The final will be comprehensive, but will emphasize material since the midterm.

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

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

(a) A *dodecahedron* is a convex polyhedron that has 12 faces, each of which is a 5-sided pentagon. 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.

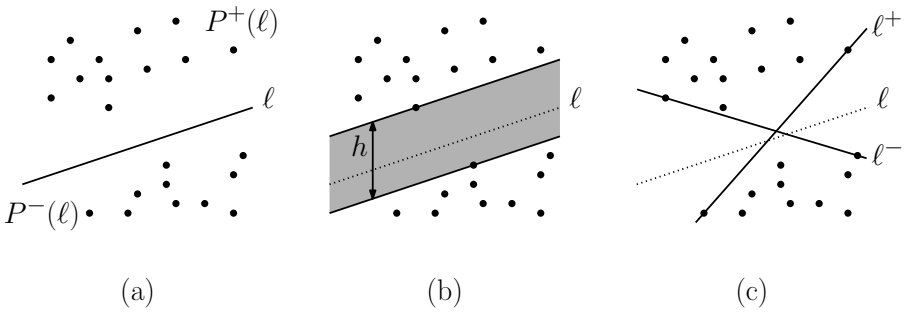

Figure 1: Problem 2: Query problem.

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)).

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.)

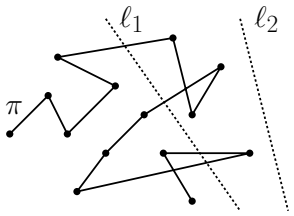

Figure 2: Problem 3: Path crossing queries.

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. 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.

(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. 3(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. 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.)

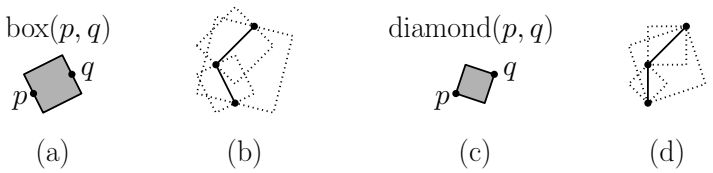

Figure 3: Problem 4: The box and diamond graphs.

Problem 5. Consider the range space (P, R) where P is a set of n points in the plane, and R is the set of all ranges arising by intersecting P with a closed halfplane.

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

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

Problem 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)).

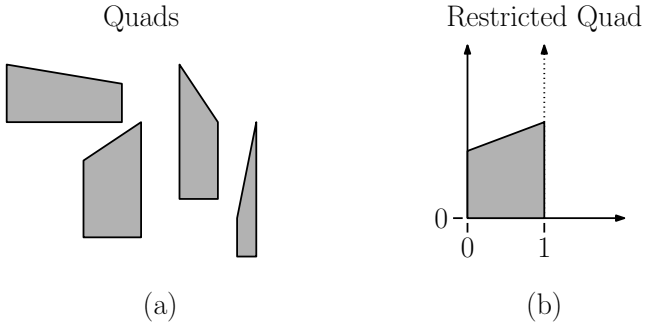

Figure 4: Problem 6: Quads and restricted quads.

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

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

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

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

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

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

Problem 8. 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 9. The purpose of this problem is to consider an approximation algorithm to an important problem that arises in clustering. You are given n points P in the plane. Given any integer k, $1 \leq k \leq n$, define $b_k(P)$ to be the Euclidean ball of minimum radius that encloses at least k points of P (see Fig. 5). (Note that the center of $b_k(P)$ may be anywhere in $\mathbb{R}^2$, not necessarily at a point of P). Let $r_k(P)$ denote the radius of this ball. The objective of this problem is to derive an algorithm that computes a factor-2 approximation, that is, it computes a ball of radius at most $2r_k(P)$ that contains at least k points of P.

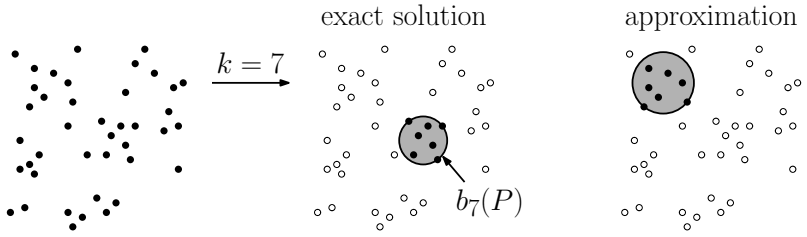

Figure 5: Problem 9: Smallest k-disk approximation.

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

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

order. Finally, let $Q = X \times Y$, be the set of points that result by taking any x-coordinate from X and any y-coordinate from Y.

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

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

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

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. 6). 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 $x^+ > x^- + 1$.)

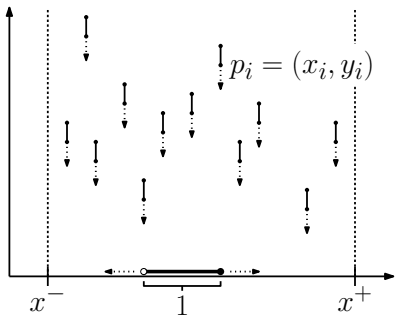

Figure 6: Problem 10: Robot motion planning.

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 in the exam booklet. If you have a question, either raise your hand or come to the front of class. Total point value is 100 points. Good luck!

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

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

(a) Given a simple polygon with n sides, what can be said about the number of triangles in any triangulation? Either state the exact number as a function of n, or provide upper and lower bounds. (No explanation needed.)

(b) In class we showed that the trapezoidal map of n nonintersecting line segments has at most $6n+4$ vertices and at most $3n+1$ trapezoids. Suppose instead that there are k instances where two segments intersect. (In Fig. 1 there are $k = 2$ segment intersection points.) At each intersection point, we shoot two bullet paths, up and down. As a function of n and k, what is the number of vertices and trapezoids in the resulting trapezoidal map? (Explain briefly.)

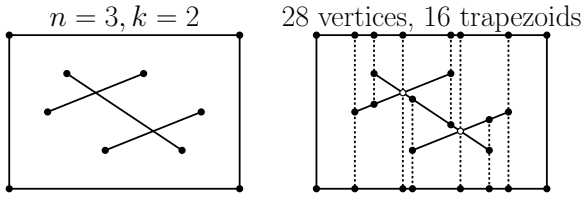

Figure 1: Trapezoidal maps with $k = 2$ intersection points.

(c) In our backward analysis of randomized incremental algorithms, we were careful to argue that the structure being analyzed does *not* depend on the order of insertion. What aspect of the analysis would fail if the structure did depend on the order of insertion? (Explain briefly.)

(d) You are writing a motion planning algorithm for a flying quadcopter drone. It can translate to any point in a 3-dimensional workspace, and it can rotate to any angle about its vertical axis. (It is *not* allowed to tilt, however.) The drone must avoid a collection of obstacles, each is a 3-dimensional Euclidean ball.

If you were model this drone as a point in a configuration space, what would the dimension of the configuration space be? Explain the meaning of each coordinate of your space. (I am *not* asking what the C-obstacles would look like.)

(e) Suppose you have a workspace with n disjoint obstacles, each of which is a k-sided convex polygon, and you are given a robot that translates in this workspace, which is modeled as an m-sided convex polygon.

(i) As a function of n, k, and m, give an upper bound on the total number of vertices in all the C-obstacles? (You may express your answer using big-Oh notation.) Explain briefly.

(ii) The C-obstacles may overlap. As a function of n, k, and m, give an upper bound on the total number of vertices in the *union* of the C-obstacles? (Again, you can use big-Oh notation.) Explain briefly.

Problem 2. (30 points) Let P be a set of points in $\mathbb{R}^2$ that all lie in the first quadrant. (That is, all the coordinates are positive.) Define a *slope set* to be the subset of points of P that lie on or below a line of *strictly negative* slope. (In Fig. 2, the left and center illustrate slope sets as black points. The right set is *not* a slope set.) Let $\Sigma = (P, \mathcal{R})$ denote a range space where $\mathcal{R}$ is the set of all slope sets of P.

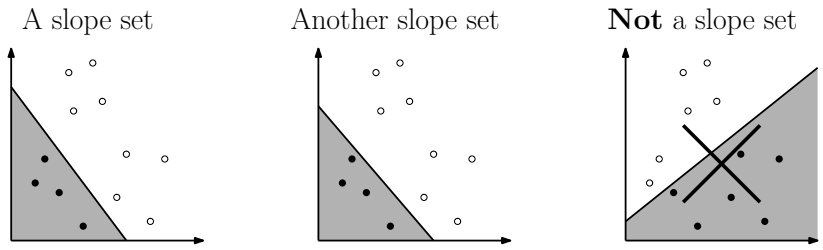

Figure 2: Slope sets.

(a) Show that there is a set consisting of 2 points (both in the first quadrant) that is *shattered* by Σ.

(b) Prove that *no* set of three points (all in the first quadrant) is shattered by Σ. (Hint: Consider the line defined by the leftmost and rightmost points, and consider cases depending on whether the middle point lies above or below this line.)

(c) What (if anything) can you infer from (a) and (b) about the *VC-dimension* of Σ?

(d) Prove from first principles that $|\mathcal{R}| = O(n^2)$, where $n = |P|$. You are *not* allowed to appeal to Sauer's lemma.

Problem 3. (25 points) Consider the same setup as Problem 3. P is a set of points in $\mathbb{R}^2$ that all lie in the first quadrant, and a *slope set* is any subset of P that lies on or below a line of strictly negative slope.

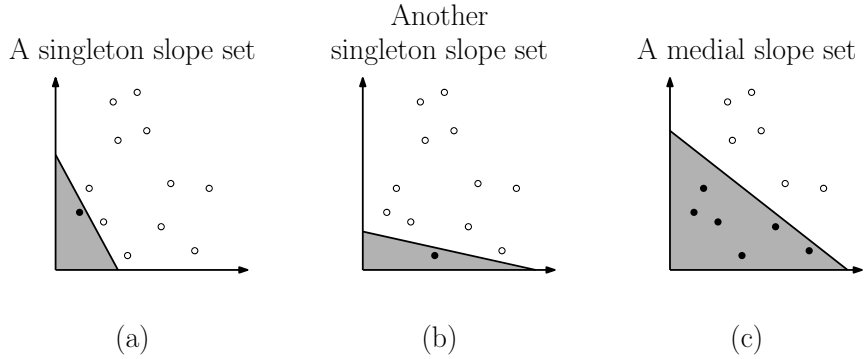

Figure 3: Singleton and Medial slope sets.

(a) A slope set is a *singleton* if it contains just a single point of P (see Fig. 3(a) and (b)). Present an efficient algorithm that, given P, computes the number of singleton slope sets in P. (For example, the set P shown in Fig. 3(a) and (b) has the two singleton slope sets shown.) Justify your algorithm's correctness and derive its running time. (Hint: For full credit, your algorithm should run in $O(n \log n)$ time.)

(b) A slope set is *medial* if it contains exactly $\lfloor n/2 \rfloor$ points of P, where $n = |P|$ (see Fig. 3(c)). Present an efficient algorithm that, given P, computes the number of medial slope sets in P. Justify your algorithm's correctness and derive its running time. (Hint: $O(n^2)$ time is possible.)

Problem 4. (15 points) You are given a set P of n points in $\mathbb{R}^d$ and a collection of m Euclidean balls in $\mathbb{R}^d$, where every ball has the same radius δ. Let $Q = \{q_1, \ldots, q_m\}$ denote the center points of these balls. The *hitting number* of a ball is the number of points of P that lie within it. The *total hitting number* is the sum of the hitting numbers over all the balls of Q. The hitting number can obviously be computed in $O(nm)$ time, but we want a more efficient solution.

Given an error parameter $\varepsilon > 0$, we want to compute the total number of pairs (p, q) for $p \in P$ and $q \in Q$ such that the point p lies "approximately" within the ball centered at q. In particular, if $\|pq\| < \delta/(1+\varepsilon)$ then this pair *must* be counted and if $\|pq\| > \delta(1+\varepsilon)$, then this pair must *not* be counted. Otherwise, your algorithm may either count or not count the pair, at its discretion.

Present an efficient algorithm that, given P, Q, δ, and ε, computes the total hitting number in this approximate setting. Let $N = n + m$. For full credit, your algorithm should run in time $O(N \log N + N/\varepsilon^d)$ time. Justify your algorithm's correctness and derive its running time. (Hint: You may assume that you are given an algorithm that for any $s > 0$, computes a bichromatic s-well separated pair decomposition, as described in Homework 3. The algorithm produces $O(Ns^d)$ bichromatic pairs and runs in time $O(N \log N + Ns^d)$.)