Homework 1: Convex Hulls and Grids

Handed out Tuesday, Feb 13. Due at the start of class Thursday, Feb 22. Late homeworks will not be accepted.

**Some hints 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 algorithms text. Nonetheless, be sufficiently complete that all critical are addressed, except for those that are obvious. (See the lecture notes for examples.)

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.” (Paraphrasing a famous mathematician: “Geometry is correct reasoning from poorly drawn figures.”)

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.

**Problem 1.** Computing an approximation to a problem is frequently easier than computing the exact solution. Grid-based algorithms are often efficient when computing approximations, but may be inefficient when computing exact solutions. But grid-based algorithms can be efficient for some exact computational problems, if a little help is provided.

Given a set $P$ of $n$ points (in the plane, for simplicity), define $\delta(P)$ to be the minimum distance between any two points of $P$, that is $\delta(P) = \min_{p_1,p_2 \in P} \|p_1p_2\|$.

Present an algorithm, which given a set $P$ of $n$ points in the plane and an estimate $E$ on the value $\delta(P)$, returns the value of $\delta(P)$ (exactly). You may assume that $\delta(P) \leq E \leq 3\delta(P)$.

Prove that your algorithm runs in $O(n)$ time.

**Problem 2.** The $k$-sum problem is defined as follows: Given $k$ sets $S_1, \ldots, S_k$ of $n$ integer numbers each, is there $a_1 \in S_1, \ldots, a_k \in S_k$ such that $a_1 + \ldots + a_k = 0$? The fastest known algorithms for $k$-sum takes $O(n^{[k/2]})$ for odd $k$, and $O(n^{[k/2]} \log n)$ for even $k$. The objective of this problem is to prove this fact for the cases $k = 4$ and $k = 5$.

(a) Give an algorithm to solve 4-sum in $O(n^2 \log n)$ time. (Or $O(n^3)$ time for partial credit.)
(b) Give an algorithm to solve 5-sum in $O(n^3)$ time. (Or $O(n^3 \log n)$ time for partial credit.)
**Problem 3.** Recall that **GeomBase** is the following problem. Given a set \( P \) of \( n \) points in the plane with the \( x \)-coordinate being an integer, and the \( y \) coordinate being either 0, 1, or 2, is there a non-horizontal line containing three of the points?

Prove that the following problem is 3SUM-hard, by reducing **GeomBase** to it. Given a set \( S \) of \( n \) horizontal line segments, and two other horizontal line segments \( s_1, s_2 \), is there a non-horizontal line segment \( a_1a_2 \) such that \( a_1 \in s_1, a_2 \in s_2 \), and \( a_1a_2 \) does not intersect any segment from \( S \)? (See Figure 1.)

![Figure 1: Example Problem 3.](image1)

**Problem 4.** The objective of this problem is to consider an alternative approach for an \( O(n \log h) \) time algorithm for computing the upper hull of a set of points in the plane.

Given two nonempty planar point sets \( A \) and \( B \), which lie on opposite sides of a vertical line, the *upper tangent* of \( A \) and \( B \) is defined to be a nonvertical line \( T \) that passes through one point \( a \in A \) and one point \( b \in B \) such that no point of \( A \cup B \) lies above this line. (See Figure 2.) It is a fact (which you need not prove) that the upper tangent exists and is unique.

![Figure 2: The upper hull of point sets A and B.](image2)

Consider the following variation of the divide and conquer convex hull algorithm, which given an \( n \)-element planar point set \( P \) computes its upper hull \( H(P) \).

1. If \( |P| \leq 3 \), then compute the upper convex hull by brute force in \( O(1) \) time and return.
2. Otherwise, partition the point set \( P \) into two sets \( A \) and \( B \), where \( A \) consists of \( \lfloor n/2 \rfloor \) the points with the lowest \( x \)-coordinates and \( B \) consists of \( \lceil n/2 \rceil \) of the points with the highest \( x \)-coordinates.
3. Compute the upper tangent \( T \) between \( A \) and \( B \) and let \( a \in A \) and \( b \in B \) be the contact point with these sets. (Note: This is done without knowing the upper hulls.)
4. Remove from \( A \) and \( B \) all the points whose \( x \)-coordinates lies strictly between \( a_x \) and \( b_x \). (Do not remove \( a \) and \( b \).) Let \( A' \) and \( B' \) be the respective subsets of remaining points.
5. Recursively compute the upper hulls \( H(A') = \text{upperHull}(A') \) and \( H(B') = \text{upperHull}(B') \).
6. Merge the two upper convex hulls \( H(A') \) and \( H(B') \) along with the segment \( ab \) to form the final upper hull.
The main difference between this algorithm and the divide-and-conquer algorithm given in class is that this algorithm computes the upper tangent before it computes the two hulls, rather than after. You may assume (without proving) that the computations of Steps (2) and (3) can be performed in $O(n)$ time, where $n = |P|$. (Step (2) follows from the well-known fast median algorithm and we will discuss Step (3) later this semester.) Using these facts, show that the algorithm described here takes $O(n \log h)$ time, where $h$ is the number of points in the final upper convex hull.

**Problem 5.** As mentioned in class, the convex hull is a somewhat non-robust shape descriptor, since if there are any distant outlying points, they will tend to dominate the shape of the hull. A more robust method is based on the following iterative approach. First compute the convex hull of all the points, remove the vertices of the hull. Then compute the convex hull of the remaining points, and again remove the vertices of the hull. Repeat this until no more points remain. The final result is a collection of nested convex polygons (where the last one may degenerate to a single line segment or single point).

![Figure 3: Repeated hulls.](image)

Given a set $P$ of $n$ points in the plane, devise an $O(n^2)$ time algorithm to compute this iterated sequence of hulls. (Fact: $O(n \log n)$ is possible, but very complicated.)

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

Consider the same set-up as in Problem 1. For each point $p \in P$, let $\alpha(p)$ denote the distance between $p$ and the nearest point of $P \setminus \{p\}$.

(a) Present an $O(n)$ time algorithm for the following problem. Given an $n$-element planar point set $P$ and a point $p \in P$, identify a subset $P' \subseteq P$ such that, for all $p' \in P$:

(i) if $\alpha(p') > \alpha(p)$, then $p' \in P'$.

(ii) if $p' \in P'$, then $\alpha(p') > \alpha(p)/3$.

(The constant 3 is (ii) can be replaced, if you wish, by any constant value that works for your algorithm.)

(b) Using the result of part (a), present an $O(n)$ time algorithm that computes $\delta(P)$. (Hint: The approach I know of is based on random sampling. It uses the result of (a) to repeatedly eliminate (in expectation) a constant fraction of the points of $P$ that cannot contribute to the value $\delta(P)$.)
Homework 2: Plane Sweep, Triangulation, and LP

Handed out Thursday, March 1. Due at the start of class Thursday, March 15. Late homeworks will not be accepted.

Note: Whenever you are asked for an \(O(T(n))\) time algorithm, unless otherwise specified you may provide a randomized algorithm whose expected running time is \(O(T(n))\).

Problem 1. Consider a \(d\)-dimensional unit hypercube, which can be defined formally as the polytope consisting of the set of points \((x_1, x_2, \ldots, x_d)\) such that \(0 \leq x_i \leq 1\). For \(0 \leq k \leq d - 1\), let \(f_d(k)\) denote the number of faces of dimension \(k\) on the \(d\)-dimensional unit hypercube. (As discussed in lecture, it may be convenient to extend the definition to include the improper faces of dimension \(-1\) and \(d\).) We all know that a hypercube in dimension \(d\) has \(2^d\) vertices. The objective of this problem is to explore the combinatorial complexity of the other faces of the unit hypercube.

(a) Observe that a \(d\)-dimensional hypercube can be derived from a \((d - 1)\)-dimensional hypercube by “extruding” (that is, stretching) the \((d - 1)\)-dimensional hypercube into the next higher dimension. (See the figure below.) Based on this observation, express the value of \(f_d(k)\) as a function of \(f_{d-1}(k')\) for one or more values of \(k'\).

(b) Prove that \(f_d(k) = 2^{d-k} \binom{d}{k}\). (Hint: Use induction on \(d\) and \(k\)).

![Figure 1: Extruding a hypercube.](image)

Problem 2. You are given a collection of line segments in the plane, which may intersect. The resulting collection subdivides the plane into a planar graph consisting of vertices (at segment endpoints and intersection points), edges, and faces. (This graph is called an arrangement.) One of the faces of such an arrangement is unbounded (extends to infinity) and is called the external face. The degree of a face in an arrangement of segments is the number of edges on the face. (Note that multiple occurrences of the same edge are counted multiply, and the two sides of the same portion of an edge are counted twice.) For example, the shaded face in the figure below has degree 26.) Note that a face in an arrangement of line segments may have islands floating within them. The edges of these islands also are counted in the face’s degree.

Present an algorithm that, given a collection \(n\) line segments in the plane, outputs the maximum degree of any face, including the external face. For example, in the figure below, the shaded face has the highest degree, and so the algorithm would output 26. Your algorithm should run in \(O((n+I) \log n)\) time, where \(I\) is the total number of intersections among the segments.

Clearly explain what information you store in your plane sweep status and event queue. Explain what events your algorithm processes. You may make the usual general position assumptions.

Problem 3. The following problem asks you to solve two basic problems about a simple polygon in linear time. Ideally, your solution should not make use of any fancy data structures.
(a) You are given a cyclic sequence of vertices forming the boundary of a simple \( n \)-sided polygon \( P \), but you are not told whether the sequence has been given in clockwise or counterclockwise order. (More formally, if the polygon’s boundary were to be continuously morphed into circle, what would be the orientation of the circle?) Give an \( O(n) \) time algorithm that determines which is the case. You may assume general position.

(b) You are given a simple \( n \)-sided polygon \( P \) where \( n \geq 4 \). Recall that a diagonal is a line segment joining two vertices such that the interior of the segment lies entirely within the polygon’s interior. Give an \( O(n) \) time algorithm that finds any diagonal in \( P \).

Problem 4. The objective of this problem is to provide the missing ingredient in the \( O(n \log h) \) time convex hull algorithm presented as Problem 4 in the previous homework.

Show that the following problem can be solved in \( O(n) \) time. You are given two sets of points \( A \) and \( B \), which are separated by a vertical line (so that \( A \) lies to the left of \( B \)). The total number of points in these two sets is \( n \). Compute the upper tangent line connecting the two sets. (Recall that the upper tangent is defined to be a nonvertical line \( \ell \) that passes through one point \( a \in A \) and one point \( b \in B \) such that no point of \( A \cup B \) lies above this line.)

(Hint: Use linear programming. Although it is possible to solve the problem in any context, it will be a lot easier to visualize the solution if you assume that all the points of \( A \) lie to the left of the \( y \)-axis and the points of \( B \) lie to the right. Of course, this can be achieved by an \( O(n) \) time translation.)

Problem 5. Consider two point sets \( A \) and \( B \), where all the points of \( A \) lie strictly to the left of the \( y \)-axis and all the points of \( B \) lie strictly to the right of the \( y \)-axis. Let \( n_A = |A| \) and \( n_B = |B| \), and suppose (because it will simplify things) that both \( n_A \) and \( n_B \) are odd. For a given slope \( \theta \), let \( \alpha(\theta) \) be a line of slope \( \theta \) that passes through a point of \( A \) such that at most \( \frac{n_A}{2} \) points of \( A \) lie strictly above this line and at most \( \frac{n_A}{2} \) points of \( A \) lie strictly below this line. Define \( \beta(\theta) \) analogously for the set \( B \).

(a) As \( \theta \) ranges from \(-\infty \) to \(+\infty \), the dual of \( \alpha(\theta) \) traces out a curve in the dual plane. Show (using the properties of duality) that this is a polygonal curve (that is, consisting of straight-line edges) and that this curve is monotonically decreasing.

(b) Show the same for \( \beta(\theta) \), except that it is monotonically increasing.

(c) Using the observations from parts (a) and (b), it follows that these two monotonic curves intersect in at most (and in fact exactly) one point, call it \( q \). Let \( \theta_q \) denote the slope value of this point. Explain (using the properties of duality) what the dual of \( q \) represents in the primal setting.

(d) Given an \( O(n) \) time algorithm, which given \( A \) and \( B \) and any slope value \( \theta \) (but not \( q \)), determines whether \( \theta < \theta_q \), \( \theta = \theta_q \), or \( \theta > \theta_q \).

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.)
In the last homework we saw a solution to the closest pair problem that was based on integer grids. There is also a divide-and-conquer algorithm for this problem (see the book “Introduction to Algorithms” by Cormen, et al. if you are interested.) In this problem, we will consider a solution based on plane sweep. (This is undoubtedly the hardest of the three, but is a good exercise to see that you understand plane sweep.)

You are given a set \( P \) of \( n \) points in the plane. Present an \( O(n \log n) \) time algorithm that computes the closest pair of points of \( P \). (Hint: When a new point \( p \) is encountered by the vertical sweep line, you should only need to access a constant number of points in its general neighborhood. It is up to you to develop an appropriate notion of “general neighborhood” that makes this possible.)
Homework 3: Point Location, Voronoi Diagrams and Delaunay Triangulations

Handed out Tuesday, April 10. Due at the start of class Thursday, April 19. Late homeworks will not be accepted.

Note: Whenever you are asked for an $O(T(n))$ time algorithm, unless otherwise specified you may provide a randomized algorithm whose expected running time is $O(T(n))$.

Problem 1. Let $P$ be a bounded convex polytope in $\mathbb{R}^3$ having $n$ edges. Present a data structure which, given a query point $q \in \mathbb{R}^3$, determines whether $q$ lies within $P$. Your data structure should use $O(n)$ space and should answer queries in $O(\log n)$ time.

(Hint: Reduce to a 2-dimensional point location problem.)

Problem 2. You are given a sequence $P$ of $n$ sites in the plane sorted by $x$-coordinate. You are also given a horizontal line $\ell$, such that all the sites of $P$ lie above $\ell$.

(a) Prove that the intersection of the Voronoi diagram of $P$ with the line $\ell$ partitions $\ell$ into at most $n$ intervals (at most one per site of $P$) such that the order of the intervals is the same as the order of the sites of $P$. (See Fig. 1.)

(b) Give an $O(n)$ time algorithm, which given such a sorted sequence $P$ and $\ell$, computes the intersection of the Voronoi diagram of $P$ with $\ell$. (Provide a very brief justification of your algorithm’s correctness and running time.)

Figure 1: Problem 2.

Problem 3. Computing geometric properties of a union of spheres is important to many applications in computational biology. The following problem is a 2-dimensional simplification of one of these problems.

You are given a set $P$ of atoms representing a protein molecule, which for our purposes will be represented by a collection of circles in the plane, all of equal radius $r_a$. Such a protein lives in a solution of water. We will model a molecule of water by a circle of radius $r_b > r_a$. A water molecule cannot intersect the interior of any protein atom, but it can be tangent to one.

We say that an atom $a \in P$ is solvent-accessible if there exists a placement of a water molecule that is tangent to $a$, and the water molecule does not intersect any of the other atoms of $P$. In Fig. 1, all atoms are solvent-accessible except for those that are shaded. Given a protein molecule $P$ of $n$ atoms, devise an $O(n \log n)$ time algorithm to determine all solvent-inaccessible molecules of $P$.

Problem 4. The Delaunay triangulation of a set of points in the plane has a number of nice properties. The objective of this problem is to demonstrate two properties that it does not have.
Problem 5. In class we showed that the minimum spanning tree of a set of point sites in the plane is a subgraph of the Delaunay graph. (The Delaunay graph is a graph whose edges consist of the edges of the Delaunay triangulation.) The following problem generalizes this observation to two other graphs. Throughout, let $P$ be a set of $n$ point sites in the plane. (You may make the usual general position assumptions.) We define two undirected graphs $G(P)$ and $R(P)$, whose vertex set consists of the sites of $P$ and whose edges are defined below.

(a) For $p, q \in P$, the edge $\{p, q\}$ is in $G(P)$ if and only if the circle having $pq$ as its diameter contains no other sites of $P$. Prove that $G(P)$ is a subgraph of $DT(P)$.

(b) For $p, q \in P$, the edge $\{p, q\}$ is in $R(P)$ if and only if there is no other site that is simultaneously closer to $p$ and $q$ than they are to each other. That is, there exists no $r \in P$ such that $\max(||rp||, ||rq||) < ||pq||$. Prove that $R(P)$ is a subgraph of $G(P)$. (Note: By part (a) it follows therefore that $R(P)$ is a subgraph of $DT(P)$.)

(c) Give an example of a planar point set $P$ such that the three graphs $G(P)$, $R(P)$ and $DT(P)$ are all distinct. (Hint: This can be done with just 4 points.) Explain your construction.

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.)

Given a finite set $P$ of points in the plane, we say that a triangulation $T$ of $P$ is acute if every angle of every triangle of $T$ is less than 90 degrees.

(a) Prove that any acute triangulation is Delaunay. That is, if $T$ is an acute triangulation of $P$ and $DT(P)$ is the Delaunay triangulation of $P$, then $T = DT(P)$. (As usual, you may assume the points are in general position. Note that it follows that if $P$ has an acute triangulation, it is unique.)

(b) Let $T$ be an acute triangulation and let $\theta_{\text{max}} < \pi/2$ be the largest angle of this triangulation. Prove that there exists a parameter $t > 1$, which may depend on $\theta_{\text{max}}$, such that $T$ is a $t$-spanner. That is, letting $P$ denote the vertices of $T$, show that for any $p, q \in P$, there exists a path in $T$ whose length is at most $t||pq||$. (Note: It is possible to establish a value on $t$ that does not depend
on $\theta_{\text{max}}$, but in my solution the value of $t$ increases with $\theta_{\text{max}}$ and tends to infinity in the limit as $\theta_{\text{max}} \to \pi/2$.

Proving part (b) seems to be harder than I had originally thought. Here are some hints. You may prove all of them, some of them, or simply give an independent proof for (b) based on your own ideas. (I think that (b.3) is technically the most interesting of the three.) Throughout, let $T$ be an acute triangulation, and let $P$ denote $T$’s vertex set.

(b.1) Prove that for any edge $\{a, b\} \in T$, there does not exist a point $c \in P$ such that $\angle acb \geq \pi/2$. (Note: This clearly true for the vertices of the (up to) two triangles that are incident to $\{a, b\}$. The objective is to show that this is also true for every point of $P$. (See Fig. 3(a).)

(b.2) Assuming (b.1), prove that for every pair $p, q \in P$ if $\{p, q\}$ is not edge of $T$, then there exists a point $r \in P$ such that the edge $\{p, r\}$ is in $T$ and $\angle rpq < \pi/4, \angle rqp < \pi/2$. (Note that this is equivalent to saying that there exists a point of $P$ lying within the shaded right triangle shown in the Fig. 3(b).)

(b.3) Assuming (b.2), prove the spanner property (b) given above. (Hint: The proof is by induction on the distance $\|pq\|$ between the source and destination points. It involves only basic high-school trigonometry and the triangle inequality, which states that for any three points $a, b, c$, we have $\|ac\| \leq \|ab\| + \|bc\|$.)
Handed out Tuesday, April 24. Due at the start of class Tuesday, May 1. Late homeworks will not be accepted.

Note: Whenever you are asked for an $O(T(n))$ time algorithm, unless otherwise specified you may provide a randomized algorithm whose expected running time is $O(T(n))$.

Problem 1. In class we have sometimes talked about computing with infinite or infinitesimal quantities. The problem is that you cannot set a variable to $+\infty$ and perform a floating-point computation. The objective of this problem is to explore exactly how this can be done in through a couple of examples. For each of the following problems, assume that $\varepsilon > 0$ is a small value and explain how to evaluate each predicate in the limit as $\varepsilon \to 0$. To answer each problem, plug the appropriate value of $\varepsilon$ into your formula and (symbolically) evaluate the formula the limit as $\varepsilon \to 0$. Your final answer should not involve the use of limits or any infinite or infinitesimal quantities.

(a) When performing plane sweep through a line arrangement, we need to test whether one line $l_1 : y = a_1 x + b_1$ intersects the sweep line at a lower point than another line $l_2 : y = a_2 x + b_2$. Assuming that the sweep line is at $x = x_0$, this is evaluated using the predicate:

$$a_1 x_0 + b_1 < a_2 x_0 + b_2.$$

When we start the plane sweep we start at $x_0 = -\infty$. To simulate this, consider the above predicate for $x_0 = -1/\varepsilon$, in the limit as $\varepsilon \to 0$. How would you evaluate the above inequality in the limit? More generally, if you want to sort a set of lines (say, from bottom to top) according to their intersection with the vertical line at $x_0 = -\infty$ how would sort them? How about at $x_0 = +\infty$. (Hint: If this seems easy, it is because it is intended to be a warm-up for part (b).)

(b) Recall the inCircle test used in the computation of Delaunay triangulations. It tests whether the point $d$ lies within the circle $a$, $b$, $c$, which are assumed to be given in counterclockwise order.

$$\text{inCircle}(a, b, c, d) \equiv \det \begin{pmatrix} a_x & a_y & a_x^2 + a_y^2 & 1 \\ b_x & b_y & b_x^2 + b_y^2 & 1 \\ c_x & c_y & c_x^2 + c_y^2 & 1 \\ d_x & d_y & d_x^2 + d_y^2 & 1 \end{pmatrix} > 0.$$  

Suppose that $b$, $c$, and $d$ are points with finite coordinates, but that $a$ has the coordinates $(a_x, a_y) = (1/\varepsilon, -1/\varepsilon)$. Explain how to evaluate this predicate in the limit as $\varepsilon \to 0$. Express your answer in terms of (smaller) determinants involving the coordinates $b$, $c$, and $d$ only.

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

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

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

(b) Develop a rectangle range counting data structure of size $O(n^2)$ that can be used to compute the number of points of $P$ lying within any canonical rectangle in $O(1)$ time. (Hint: The answer to the query will involve both addition and subtraction.) Because the rectangle query is canonical, you should not assume general position. Your procedure should allow the option of either including or excluding points on the boundary of the rectangle.

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

Problem 3. An important source of geometric problems in mobile applications involves objects that are moving in time. We will consider such an example in this problem.

Consider a set of $n$ cars moving with constant velocities on a straight road. We may model these cars as a set of $n$ points $P = \{p_1, p_2, \ldots, p_n\}$, where the position of point $p_i$ at time $t$, denoted $p_i(t)$, is equal to $a_it + b_i$. Above these cars a helicopter is flying. The helicopter has a camera that points straight down and has a fixed field of view of angle $0 < \theta < \pi$. (See Fig. 2(a).) The helicopter pilot is afraid of heights (bad career choice!), and would like to fly as low as possible subject to the constraint that the camera must keep all the cars in its field of view at all times.

(a) Given a finite time interval $T = [t_0, t_1]$, compute a function $h(t) = (h_x(t), h_y(t))$ for all $t \in T$, which gives the position of the helicopter at any time, such that it has minimum height $h_y(t) \geq 0$ and can see all the cars in its camera. (Hint: Your function traces out a, possibly self-intersecting, polygonal curve in the plane. How many segments might this curve have? It may help to visualize this problem in terms of an arrangement of lines in the plane.)

Justify the correctness of your algorithm and derive its running time and an upper bound on the complexity of the polygonal curve output by your program.
Suppose there is a maximum speed limit for the cars, denoted $a_{\text{max}}$. That is, $|a_i| \leq a_{\text{max}}$ for $1 \leq i \leq n$. As a function of $a_{\text{max}}$ and $\theta$, how fast might the helicopter need to fly in order to achieve its objective?

Let us change the problem. Suppose now that the helicopter pilot is given a number $k$, where $2 \leq k \leq n$, and the pilot is only required to see $k$ cars in the camera at any time. (See Fig. 2(b).) It is the pilot’s choice which cars to see, and the pilot may change the set of cars under surveillance at any time (if it results in a lower height). As before, the pilot wishes to minimize the height that the helicopter needs to fly to accomplish this.

(c) Prove that for some value of $k$ the helicopter cannot achieve its objective unless it is allowed to fly infinitely fast. (This can be stated more formally as follows. Prove that for any choice of $0 < \theta < \pi$ and any speed limit $a_{\text{max}} > 0$, and any upper bound $v_{\text{max}} > 0$ on the helicopter’s speed, there exists a valid pattern of car movements (satisfying the speed limit) so that at some time the helicopter must move faster than $v_{\text{max}}$ if it is to minimize the surveillance height.)

(d) Suppose that the helicopter has been outfitted with a magic transporter, which allows the pilot to instantly jump from its current position to any point in space. Now, solve the problem of part (a), in this new context, where only $k$ of the $n$ cars need to be observed at any time, and the height is to be minimized. Your algorithm should run in $O(n^3 \log n)$ time and use $O(n)$ space. (Hint: Beware. Although you might be inclined to think at first that critical events only occur when two cars share the same location, there are additional events that need to be taken into consideration.)

Problem 4. In class we introduced the concept of a level in arrangement of $n$ lines in the plane. Assuming general position, recall that an edge of the arrangement is on the $k$th level, denoted $L_k$, if there are $k-1$ lines strictly above it and $n-k$ lines strictly below it. Generally, the number of edges on a level may exceed $O(n)$. The objective of this problem is to establish a (very clever) upper bound of $O(n \sqrt{k})$ on the combinatorial complexity, that is, the number of edges, on the $k$th level of a planar arrangement of $n$ lines.

We classify the vertices of the level into two groups. A valley vertex is one where the level moves from a line of lower slope to one of higher slope and a peak vertex is one where the level transitions from a line of higher slope to one of lower slope. (See Fig. 3(a).) We will first prove that the number of valley vertices is $O(n \sqrt{k})$, and we will bound the number of peak vertices later.

First, sort the edges by increasing order of slope, and assign the $i$th line from the top the label $i$. For any edge $e$ of the arrangement, define a potential function $\phi(e)$, to be the sum of the labels of the line containing $e$ and all the lines above this edge. (See Fig. 3(b).)

(a) Let $m$ denote the number of edges of $L_k$. Consider the sequence of edges $(e_1, e_2, \ldots, e_m)$ along the $k$th level of the arrangement, and let $\Phi = (\phi_1, \phi_2, \ldots, \phi_m)$ denote the associated sequence of potential values, that is $\phi_i = \phi(e_i)$. Prove that for all $2 \leq i \leq m$, if $e_{i-1}$ and $e_i$ are joined by a
valley vertex then the potential value increases strictly, and if they are joined by a peak vertex, the potential is unchanged.

(b) Prove that the total change in potential $\phi_m - \phi_1$ is $O(nk)$.

Since the potential increases strictly for each valley vertex of $L_k$, observe that part (b) immediately provides an upper bound of $O(nk)$ on the number of valley vertices on $L_k$. But our objective is to obtain a tighter bound by considering the amount by which the potential increases with each edge.

For $1 \leq i \leq m$, define the $i$th potential change to be $\delta_i = \phi_i - \phi_{i-1}$. (And define $\phi_0 = 0$, for convenience.) Let $\Delta = \langle \delta_1, \ldots, \delta_m \rangle$ be the sequence of all these potential changes. Let us classify the elements of $\Delta$ into two groups. We say that a change is large if $\delta_i > \sqrt{k}$, and it is small if $\delta_i \leq \sqrt{k}$.

(c) Prove that the number large potential changes of $\Delta$ is at most $O(n\sqrt{k})$.

(d) Prove that the number of small potential changes of $\Delta$ is $O(n\sqrt{k})$. (Hint: How many pairs of lines could generate a potential change of 1? of 2? of $j$?)

(e) Combine the results of parts (c) and (d) to show that the total number of valley vertices of $L_k$ is $O(n\sqrt{k})$.

(f) Complete the proof by showing that the number of peak vertices is asymptotically the same as the number of valley vertices, thus completing the proof that the total complexity of $L_k$ is $O(n\sqrt{k})$.

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.)

The objective problem is to develop an efficient data structure for range counting over a set of $n$ points, where the ranges are halfplanes.

(a) Given a set of $n$ points $P$ in the plane, show that it is possible to partition this set (assuming general position) into four disjoint regions defined by two straight lines, such that each region has at most $\lceil n/4 \rceil$ points.

(b) Use this observation to devise a data structure of size $O(n)$ for answering halfplane range searching. Assuming that the construction of (a) can be performed in $O(n)$ time, prove that the time to compute your data structure is $O(n \log n)$.

(c) Explain how halfplane counting queries are answered using your data structure. Prove that a query can be answered in time $O(n^\alpha)$ time, for some constant $0 < \alpha < 1$. What is the value of $\alpha$?

(Hint: It may simplify your descriptions to assume that $n$ is a power of 4.)
Homework 5: Range Searching and Approximation

Haded out Tuesday, May 1. Due at the start of class Thursday, May 10. Late homeworks will not be accepted.

Note: Whenever you are asked for an $O(T(n))$ time algorithm, unless otherwise specified you may provide a randomized algorithm whose expected running time is $O(T(n))$.

» Work any four of these five problems. If you attempt more than four, please indicate which problem you do not want to have graded. «

Problem 1. In Homework 1, we saw how to compute a sequence of layers of convex hulls for a point set $S$. Use this structure to develop a data structure for answering halfplane range reporting queries for the set $S$. Your data structure should be able to answer a query in $O((k+1)\log n)$ time, where $k$ is the number of points inside the query range. (Hint: Begin by showing that given a $n$-sided convex polygon $P$, it is possible to preprocess it to answer the following queries in in $O(\log n)$ time. Given a halfplane $h$ determine that $P$ does not intersect $h$, or if it does return any vertex of $P$ that lies within $h$.)

Problem 2. The following problem can be called shallow halfplane range searching. You are given a set of $n$ points $P$ in the plane and a fixed constant $k$ (independent of $n$). Describe a data structure to answer the following queries efficiently. The query is given a halfplane $h$ and the answer to the query is “yes” if there are at most $k$ points of $P$ lying within $h$ and “no” otherwise. (If a point of $P$ lies on the boundary $h$, it may either counted or not, your choice.) Your data structure should be able to answer queries in $O(\log n)$ time and should use $O(n)$ space. Preprocessing time should be $O(n^2)$.

Figure 1: An example for Problem 2, for $k = 3$.

Problem 3. The objective of this problem is to investigate the VC-dimension of some range spaces. We begin with some definitions. A range space $\Sigma$ is a pair $(X,R)$, where $X$ is a (finite or infinite) set, called points, and $R$ is a (finite or infinite) family of subsets of $X$, called ranges. For any $Y \subseteq X$, define $P_R(Y)$, called the projection of $R$ on $Y$, to be $\{r \cap Y \mid r \in R\}$. If $P_R(Y)$ contains all the subsets of $Y$ (that is, if $Y$ is finite, we have $|P_R(Y)| = 2^{|Y|}$) then we say that $Y$ is shattered by $R$. The Vapnik-Chervonenkis dimension (or VC-dimension) of $\Sigma$, denoted $VC(\Sigma)$, is the maximum cardinality of a shattered subset of $X$. If there are arbitrarily large shattered subsets then the VC-dimension is defined to be $\infty$.

For each of the following range spaces, determine 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.)

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 $y \leq y_0$. Prove that $VC(\Sigma) = 2$.

$VC(\Sigma) \geq 2$: 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 2 that is shattered.
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.

For simplicity, you may assume that points are in general position.

(a) \(\Sigma_R = (\mathbb{R}^2, R)\), where \(R\) is the set of all closed axis-aligned rectangles.

(b) \(\Sigma_S = (\mathbb{R}^2, S)\), where \(S\) is the set of all closed axis-aligned squares.

(c) \(\Sigma_D = (\mathbb{R}^2, D)\), where \(D\) is the set of all closed circular disks in the plane.

**Problem 4.** In this problem you may assume the following rather remarkable result, which we will not prove.

Theorem: For any set \(P\) of \(n\) points in the plane, there exists a spanning tree \(T\) of \(P\), whose edges are straight line segments, such that any line in the plane, crosses at most \(O(\sqrt{n})\) edges of \(T\).

(a) Prove that the above theorem is tight in the sense that there exists a set of \(n\) points in the plane such that for any spanning tree \(T\) on these points, there exists a line (depending on \(T\)) that intersects at least \(\Omega(\sqrt{n})\) edges of \(T\). (Hint: For my proof, I arranged the points in a grid, and only needed to consider horizontal and vertical lines.)

(b) (Using the above theorem) prove that there exists a path \(\pi\) consisting of straight line segments that spans all the points of \(P\), such that any line intersects at most \(O(\sqrt{n})\) edges of \(\pi\).

(c) Assuming that \(n\) is even, (using the above theorem) prove that there exists a matching \(M\) of the points of \(P\), such that any line intersects at most \(O(\sqrt{n})\) edges of \(M\). (Recall that a matching is a collection of line segments joining pairs of points of \(P\) such that each point \(P\) is incident to exactly one segment of the collection.)

The objective of this problem is to study how evenly we can color a set of points with two colors. To motivate this concept, suppose that we color all the inhabitants of the world with two colors (e.g., male and female) in as even a manner as possible. We would expect that the number of men and women residing in any region should be roughly equal.

To make this concept of an even partition more formal, let \(P\) denote a set of \(n\) points in the plane. Let \((R, B)\) be a partition of this set into two subsets, which are called the red and blue points, respectively. Define the halfplane red-blue discrepancy of \((R, B)\), denoted \(\Delta_{(R,B)}(P)\) to be the maximum (or more formally the supremum) over all halfplanes \(h\) of the absolute difference in the number of red points and blue points in \(h\). That is,

\[
\Delta_{(R,B)}(P) = \sup_h | |h \cap R| - |h \cap B| | .
\]

The halfplane red-blue discrepancy of \(P\), denoted \(\Delta(P)\) is defined to be the minimum over all red-blue partitions of \(\Delta_{(R,B)}(P)\).

(d) Use the above results to prove that the halfplane red-blue discrepancy of any planar point set \(P\) of size \(n\) is \(O(\sqrt{n})\). (Note: This is not tight. A more sophisticated argument, which is based on results from probability theory that we have not covered, can be applied to show that the red-blue halfplane discrepancy of any such point set is \(O(n^{1/4} \log n)\).)

**Problem 5.** The objective of this problem is to establish an important theorem on approximating convex bodies. To make things as simple as possible (but still retain the essence of the full proof) we will limit ourselves to a a very restricted situation in the plane.

Let \(K\) denote a convex body in the plane bounded above by an \(x\)-monotone curve \(f(x)\) for \(0 \leq x \leq 1\), where \(f(0) = f(1) = 0\), and below by the \(x\)-axis. (See Fig. 2(a).) Let us assume that \(f'(x)\) is defined
for $0 \leq x \leq 1$, and that the absolute value of the slope of $f(x)$ is bounded, that is, there is a constant $c$ such that $|f'(x)| \leq c$, for $0 \leq x \leq 1$. Given some parameter $\varepsilon > 0$, our objective is to approximate $f(x)$ by a convex polygon, such that the vertical distance between $f(x)$ and the polygon is at most $\varepsilon$. We will show that this can be done by a polygon of complexity $O(1/\sqrt{\varepsilon})$.

The key to the construction is to generate a set of points along the upper boundary, so that between consecutive points the horizontal distance and change in slope is bounded by $\sqrt{\varepsilon}$.

- First we sample points so that the horizontal distances are bounded. Let $k_1 = \lceil 1/\sqrt{\varepsilon} \rceil$. Let $P_1$ consist of the set of $k_1 + 1$ points $(x_i, f(x_i))$, where $x_i = i/k_1$ for $i = 0, 1, 2, \ldots, k_1$. (See Fig. 2(b).)

- Next we sample points so that the differences in slopes are bounded. Let $s_0 = f'(0)$ and let $s_1 = f'(1)$. Let $\Delta = s_1 - s_0$ be the total range of slopes. Let $k_2 = \lceil 2c/\sqrt{\varepsilon} \rceil$. Consider the set of slopes $t_i = s_0 + i\Delta/k_2$ for $i = 0, 1, 2, \ldots, k_2$. Let $z_i$ be the point such that $f'(z_i) = t_i$. Let $P_2$ consist of the points $(z_i, f(z_i))$ for $i = 0, 1, 2, \ldots, k_2$.

(a) Let $P = P_1 \cup P_2$. Show that the number of points of $P$ is $O(1/\sqrt{\varepsilon})$ (assuming $c$ is a constant). Show that the horizontal distance between each consecutive pair of points of $P$ is at most $\sqrt{\varepsilon}$. Show that the difference in slope of $f$ between consecutive pairs of points of $P$ is at most $\sqrt{\varepsilon}$. (See Fig. 2(d).)

(b) Let $H$ be the convex hull of $P$. Using the observations of (a), prove that for all $0 \leq x \leq 1$, the vertical distance between the upper boundary of $K$ and the upper hull of $H$ is at most $a \cdot \varepsilon$, for some constant $a$. (See Fig. 2(d).)

Note: The result generalizes to all dimensions. In particular, it can be shown that given any bounded convex body $K$ in $d$-dimensional space, it is possible to approximate $K$ by a convex polytope $Q$ of complexity $(1/\varepsilon)^{(d-1)/2}$, so that the distance between any point of $Q$ to its closest point on $K$ (and vice versa) is $\varepsilon \Delta(K)$, where $\Delta(K)$ denotes the diameter of $K$.

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

Modify your solution to Problem 1, so the running time is $O(k + \log n)$ with $O(n)$ total space. (Hint: Use fractional cascading.)
Sample Problems for the Midterm Exam

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 midterm exam. (They certainly do not reflect the length of the exam!) Note that some topics we covered this semester were not covered in previous semesters. (In particular, the upper bound theorem and Chan’s algorithm are new this semester.)

The exam will be closed-book and closed-notes. You may use one sheet of notes (front and back). In all problems, unless otherwise stated, you may assume general position, and you may use 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.

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

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

(c) Recall the orientation primitive Orient(a, b, c), which given three points in the plane, returns a positive value if the points are ordered counterclockwise, zero if they are collinear, and negative if clockwise. Show how to use this primitive (one or more times) to determine whether a point d lies within the interior of the triangle defined by the points a, b, and c. (You may assume that a, b, and c are oriented counterclockwise.)

(d) In class we showed that 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 the figure below n = 8, h = 3, and k = 4). As a function of n, h and k, how many triangles will such a triangulation have?

(e) In Fortune’s algorithm for planar Voronoi diagrams, given n sites, what is the maximum number of distinct arcs that a single site can contribute to the beach line? (You may use O-notation.)

Problem 2.
(a) You are given a convex polygon in the plane having \( n_c \) sides and an \( x \)-monotone polygon having \( n_m \) sides. What is the maximum number of intersections that might occur between the edges of these two polygons? (You may use \( O \)-notation.)

(b) You are given two \( x \)-monotone polygonal chains \( P \) and \( Q \) with a total of \( n \) vertices between them. Prove that \( P \) and \( Q \) can intersect at most \( O(n) \) times.

(c) Does the bound proved in (b) still apply if the two polygonal chains that are monotone with respect to different directions (e.g., one monotone with respect to \( x \) and the other monotone with respect to \( y \))? Justify your answer.

**Problem 3.** You are given a set of \( n \) vertical line segments in the plane. Present an efficient algorithm to determine whether there exists a line that intersects all of these segments. An example is shown in the figure below. (Hint: \( O(n) \) time is possible.) Justify your algorithm’s correctness and derive its running time.

**Problem 4.** Consider the segments shown in the figure below. Consider the insertion order \( \langle s_1, s_2 \rangle \).

(a) Show the trapezoidal map after the insertion of \( s_1 \).

(b) Show the trapezoidal map after the insertion of both segments. (Label the trapezoids for part (c).)

(c) Show the final history DAG for this insertion order.

**Problem 5.** Given a set of \( n \) points in the plane, a triangulation of these points is a planar straight line graph whose outer face is the convex hull of the point set, and each of whose internal faces is a triangle. There are many possible triangulations of a set of points. Throughout this problem you may assume that no three points are collinear.

(a) Among the \( n \) points, suppose that \( h \) lie on the convex hull of the point set. As a function of \( n \) and \( h \), what is the number of triangles (internal faces) \( t \) in the triangulation. Show
how you derived your answer. (It is a fact, which you do not need to prove, that the number of triangles depends only on $n$ and $h$.) You may give an asymptotic answer for partial credit.

(b) Describe an $O(n \log n)$ algorithm for constructing any triangulation (your choice) of a set of $n$ points in the plane. Explain your algorithm and analyze its running time. You may assume that the result is stored in any reasonable representation. (Hint: There is a simple plane-sweep algorithm.)

**Problem 6.** You are given two sets of points in the plane, the red set $R$ containing $n_r$ points and the blue set $B$ containing $n_b$ points. The total number of points in both sets is $n = n_r + n_b$. Give an $O(n)$ time algorithm to determine whether the convex hull of the red set intersects the convex hull of the blue set. If one hull is nested within the other, then we consider them to intersect.

**Problem 7.** 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 $qp$ lies entirely within the interior of $P$. (See the figure below.) 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 in linear time whether $P$ is star-shaped in $O(n)$ time. (Note: You are not given the point $q$.) Prove the correctness of your algorithm.

**Problem 8.** 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 $Lo$ and the upper-right point $Hi$.

1. Let $P = \{p_1, p_2, \ldots, p_n\}$ be a random permutation of the points.
3. For $i = 2$ through $n$ do:
   - (a) if $p_i[x] < Lo[x]$ then (*) $Lo[x] = p_i[x]$.
   - (b) if $p_i[y] < Lo[y]$ then (*) $Lo[y] = p_i[y]$.
   - (c) if $p_i[x] > Hi[x]$ then (*) $Hi[x] = p_i[x]$.
   - (d) if $p_i[y] > Hi[y]$ then (*) $Hi[y] = p_i[y]$.
Clearly this algorithm runs in $O(n)$ time. Prove that the total number of times that “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. (Hint: Use backwards analysis.)

**Problem 9.** Define a *strip* to be the region bounded by two (nonvertical) parallel lines. The *width* of a strip is the vertical distance between the two lines.

(a) Suppose that a strip of vertical width $w$ contains a set of $n$ points in the primal plane. Dualize the points and the two lines. Describe (in words and pictures) the resulting configuration in the dual plane. Assume the usual dual transformation that maps point $(a, b)$ to the line $y = ax - b$, and vice versa.

(b) Give an $O(n)$ time algorithm, which given a set of $n$ points in the plane, finds the nonvertical strip of minimum width that encloses all of these points.

**Problem 10.** 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 the figure below left for an example.)

(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 were inserted in increasing order of $y$-coordinate. (See the figure above center and right.)

(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 $O(1)$. 

4
Problem 11. You are given a collection of vertical line segments in the first quadrant of the $x, y$ plane. Each line segment has one endpoint on the $x$-axis and the other endpoint has a positive $y$-coordinate. Imagine that from the top of each segment a horizontal bullet is shot to the left. The problem is to determine the index of the segment that is first hit by each of these bullet paths. If no segment is hit, return the index 0. (See the figure below.)

The input is a sequence of top endpoints of each segment $p_i = (x_i, y_i)$, for $1 \leq i \leq n$, which are sorted in increasing order by $x$-coordinate. The output is the sequence of indices, one for each segment.

Present an $O(n)$ time algorithm to solve this problem. Explain how your algorithm works and justify its running time.

Output: 0,1,2,0,4,5,4,0,9,10

Problem 12. Given an $n$-vertex simple polygon $P$ and an edge $e$ of $P$, show how to construct a data structure to answer the following queries in $O(\log n)$ time and $O(n)$ 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 $f$. You may assume that $e$ is rotated into a convenient position, if it helps to simplify things. (Hint: Reduce this to a point location query.)
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) Give a short answer (a few sentences) to each question.

(a) There is a lower bound of $\Omega(n \log n)$ (from the element-uniqueness problem) on computing the closest pair of points in the plane. Nonetheless, we saw a grid-based algorithm that runs in $O(n)$ time. Why is this not a contradiction?

(b) How many edges (1-faces) does a $d$-dimensional simplex have? How many edges are incident to a single vertex in a $d$-dimensional simplex? Explain briefly.

(c) You are given a set of $n$ disjoint line segments in the plane. How many trapezoids are there in a trapezoidal decomposition of these segments? Explain briefly. (Give an exact, not asymptotic, answer.)

(d) Suppose the line segments of part (c) are not disjoint, but have $I$ intersection points. Each intersection point results in two bullet paths. As a function of $n$ and $I$, how many faces are there now? Explain briefly.

(e) What are the two types of events that arise in Fortune’s plane sweep algorithm for Voronoi diagrams. Briefly explain what each event means. (You do not need to explain how each event is processed.)

**Problem 2.** (10 points)

Let $P_1$ and $P_2$ denote two polygonal chains that share the same start and end points (see the figure to the right). They are supposed to form the upper and lower parts of a horizontally monotone polygon, which we are to triangulate. However, we do not trust the integrity of the input. Explain how to modify the monotone polygon triangulation algorithm given in class so that as it runs it also tests the following two conditions:

1. The chains $P_1$ and $P_2$ are indeed horizontally monotone.
2. The polygon is simple, meaning that it does not self intersect.

Your modified algorithm should run in $O(n)$ time and can terminate as soon as either error is detected. Briefly justify the correctness and running time of your algorithm. (It is not necessary to give the entire triangulation algorithm, just explain the modifications.)

Note: You can get half credit by presenting any $O(n)$ time algorithm, even if it is not based on the monotone triangulation algorithm given in class.
Problem 3. (20 points)

You are given two sets of points in the plane, $P_1$ and $P_2$. Let $n$ denote the total number of points in $P_1 \cup P_2$. A partial classifier is a pair of parallel lines $\ell_1$ and $\ell_2$, such that all the points of $P_1$ lie on or above $\ell_1$ and all the points of $P_2$ lie on or below $\ell_2$. The cost of the partial classifier is the vertical distance between these lines.

(a) Assuming the standard dual transformation, which maps the point $(a, b)$ to the dual line $y = ax - b$ (and vice versa), give a geometric explanation of what a partial classifier means in the dual plane.

What is the meaning of cost in the dual setting?

(b) Give an algorithm which, given point sets $P_1$ and $P_2$, computes a partial classifier of minimum cost. $O(n)$ time is possible. Briefly justify the correctness and running time of your algorithm.

Problem 4. (20 points) Given two points $p_1 = (x_1, y_1)$ and $p_2 = (x_2, y_2)$ in the plane, we say that $p_2$ dominates $p_1$ if $x_1 \leq x_2$ and $y_1 \leq y_2$. Given a set of points $P = \{p_1, p_2, \ldots, p_n\}$, a point $p_i$ is said to be maximal if it is not dominated by any other point of $P$. (See the figure below.)

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 5. (25 points) Recall that given an $n$ element point set in the plane, Chan’s convex hull algorithm iterates over values $t = 1, 2, \ldots$. For each value of $t$, it partitions the points into subsets of size $m \leftarrow \min(2^t, n)$. For each partition it invokes a procedure PartialHull($P, m$), which runs in $O(n \log m)$ time, but fails if $m < h$, where $h$ is the number of points on the final convex hull. In summary:

- For $t \leftarrow 1, 2, 3, \ldots$ do:
  - Let $m \leftarrow \min(2^t, n)$.
  - Let $L \leftarrow $ PartialHull($P, m$).
  - If $L \neq \text{"Fail"}$ then return $L$.

In each of the following cases, indicate what the running time of Chan’s algorithm would be had we used a different progression for $m$. (Use $O$-notation as a function of $n$ and $h$.) Briefly explain each answer. For full credit, attempt to express each running time in simplest terms.

(a) What would the running time be if we chose $m \leftarrow \min(t, n)$?

(b) What would the running time be if we chose $m \leftarrow \min(2^t, n)$?

(c) What would the running time be if we chose $m \leftarrow \min(2^{2^t}, n)$?

Hint: The following identities may be useful. $\sum_{i=1}^{N} \log i = \Theta(N \log N)$, $\sum_{i=1}^{N} i^c = \Theta(N^{c+1})$, $\sum_{i=1}^{N} f(i) = \Theta(f(N))$, where $f(i)$ is any function that grows at an exponential rate or higher.
Sample Problems for the Final Exam

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. Note that some topics we covered this semester (upper bound theorem, Chan’s convex hull algorithm, VC-dimension) were not covered in previous semesters. The final will be comprehensive, but will emphasize material since the midterm.

The exam will be closed-book and closed-notes. You may use two sheets of notes (front and back). 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.

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

(b) A kd-tree of \(n\) points in the plane defines a subdivision of the plane into \(n\) cells, each of which is a rectangle. Is this subdivision a cell complex? Explain briefly.

(c) Given a \(k\)-\(d\) tree with \(n\) points in the plane, what is the (asymptotic) maximum number of cells that might be stabbed by an line that is not axis-parallel? Explain briefly.

(d) What is a zone in an arrangement? Given an \(n\)-line arrangement \(A\) in the plane and given an arbitrary line \(\ell\), what is the (asymptotic) maximum complexity (number of edges) of the zone of \(\ell\) relative to \(A\)? (No explanation needed.)

(e) Which of the following statements regarding the Delaunay triangulation (DT) of a set of points in the plane are true? (No explanation needed.) Among all triangulations...

(i) ...the DT minimizes the maximum angle.

(ii) ...the DT maximizes the minimum angle.

(iii) ...the DT has the minimum total edge length.

(iv) The largest angle in a DT cannot exceed 90 degrees.

(f) 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 partial credit.) Explain briefly.

Problem 2. 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 \(\{p_1, p_2, \ldots, p_n\}\) 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. 1 below the string would be \(p_1p_2p_1p_3p_6p_4p_6p_7p_9\).)

![Figure 1: Problem 2.](image-url)
(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 of the type given in part (a) cannot be of length greater than $2n - 1$. (Hint: Use induction on $n$.) These are important sequences in combinatorics, known as Davenport-Schinzel sequences.

**Problem 3.** Consider a set $P$ of $n$ points in the plane. For $k \leq \lfloor n/2 \rfloor$, point $q$ (which may or may not be in $P$) is called a $k$-splitter if every line $L$ passing through $q$ has at least $k$ points of $P$ lying on or above it and at least $k$ points on or below it. (For example the point $q$ in Fig. 2 is a 3-splitter, since every line passing through $q$ has at least 3 points of $P$ lying on either side. But it is not a 4-splitter since a horizontal line through $q$ has only 3 points below it.)

![Figure 2: Problem 3.](image)

(a) Show that for all (sufficiently large) $n$ there exists a set of $n$ points with no $\lfloor n/2 \rfloor$-splitter.

(b) Prove that there exists a $k$-splitter if and only if in the dual line arrangement, levels $\mathcal{L}_k$ and $\mathcal{L}_{n-k+1}$ can be separated by a line.

(c) Prove that any set of $n$ points in the plane has a $\lfloor n/3 \rfloor$-splitter.

(d) Describe an $O(n^2)$ algorithm for computing a $\lfloor n/3 \rfloor$-splitter for point set $P$.

**Problem 4.** Given a set $P$ of $n$ points in the plane, and given any slope $\theta$, project the points orthogonally onto a line whose slope is $\theta$. The order (say from top to bottom) of the projections is called an allowable permutation. (If two points project to the same location, then order them arbitrarily.) For example, in Fig. 3 for the slope $\theta$ the permutation would be $(p_1, p_3, p_2, p_5, p_4, p_6)$.

![Figure 3: Problem 4.](image)

(a) Prove that there are $O(n^2)$ distinct allowable permutations. (Hint: What does an allowable permutation correspond to in the dual plane?)

(b) Give an $O(n^3)$ algorithm which lists all the allowable permutations for a given $n$-point set. ($O(n^2)$ time to find the permutations and $O(n)$ time to list each one.)

**Problem 5.** You are given a set of $n$ triangles in the plane $T = \{T_1, \ldots, T_n\}$, where triangle $T_i$ has vertices $(a_i, b_i, c_i)$. Present an algorithm that computes a line $\ell$ that stabs the greatest number of triangles of $T$. (You may assume general position.) For example, in Fig. 4 there exists a line that intersects 4 of the 5 triangles. Your algorithms should run in $O(n^2)$ time and use at most $O(n^2)$ space.
Problem 6. This problem involves a 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 3 by giving an example of a set \(Q\) of 3 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 3, by proving that no 4-element set can be shattered by halfplane ranges.

(c) Present a data structure for halfplane range counting queries in the plane that uses \(O(n^2)\) space and can answer queries in \(O(\log n)\) time. Recall that the objective is to preprocess a set \(P\) of \(n\) points in the plane, so that given any query halfplane \(h\), we can return a count of the number of points of \(P \cap h\) in \(O(\log n)\) time.

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 \(P\). Given an approximation factor \(\varepsilon > 0\), we say that a point \(p' \in P\) is an \(\varepsilon\)-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 \(\varepsilon\)-approximate nearest neighbor for every point of \(P\). Justify the correctness of your algorithm.
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.** (20 points; 3-6 points each) Give a short answer (a few sentences) to each question.

(a) Is Jarvis’s March ever faster than Graham’s scan (in terms of its asymptotic running time)? If so, indicate under what circumstances.

(b) What are the three possible outcomes of any linear programming instance? Explain each briefly.

(c) When the $i$th segment is added to a trapezoidal map . . .

   (i) . . . what is the worst case number of structural changes to the map?

   (ii) . . . assuming a random insertion order, what is the expected number of structural changes?

   No explanation required. Asymptotic answers are fine.

(d) Assuming $d$ is a constant:

   (i) What is the worst-case complexity of the Delaunay triangulation of $n$ points in $d$-space?

   (ii) What is the number of vertices in an arrangement of $n$ hyperplanes in $d$-space?

   (No explanation needed.)

(e) Let $P$ be a set of $n$ points in the plane, where each point is associated with a numerical weight.

   Given any axis-aligned rectangle the query problem is to return the minimum weight among the points in this rectangle. **True or false:** It is possible to answer such queries using a kd-tree in $O(\sqrt{n})$ time. (Give a very brief explanation.)

**Problem 2.** (15 points)

Define a semi-trapezoid to be the closed region bounded on the left and right by vertical lines and above by a nonvertical line. (Two examples are shown in the figure to the right. Note that the range extends to $-\infty$ in the $y$-direction.) What is the VC-dimension of the range space of closed semi-trapezoids in the plane? Provide a brief proof.

**Problem 3.** (15 points) Consider a set $P$ of $n$ points in the plane. The distances between each pair of distinct points defines a set of $\binom{n}{2}$ interpoint distances. Present an efficient algorithm to compute an approximation the second largest interpoint distance.

More formally, your algorithm is given a set $P$ of $n$ points in the plane and a constant approximation parameter $\varepsilon > 0$. Let $\Delta$ denote the true second largest interpoint distance among the points of $P$.

Your algorithm may output any value $\Delta'$ where

$$\frac{\Delta}{1 + \varepsilon} \leq \Delta' \leq (1 + \varepsilon)\Delta.$$ 

(Hint: Use WSPDs. First give the algorithm for an arbitrary separation parameter $s$, then derive a value of $s$ that provides the approximation bound.)

For full credit, you should derive the value of $s$, and not just restate a bound that we proved in class.
Problem 4. (15 points) 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. 1(a) are connected, and the disks of Fig. 1(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 1: Problem 4.](image)

Problem 5. (15 points) You are given a set $P = \{p_1, p_2, \ldots, p_n\}$ of $n$ points in the plane. Consider all the $\binom{n}{2}$ lines passing through each pair of distinct points $p_i, p_j \in P$, and let $\ell_{\text{max}}$ to be the line of this set with the maximum slope. We are interested in computing $\ell_{\text{max}}$.

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

(b) Give an $O(n \log n)$ algorithm that computes $\ell_{\text{max}}$. Justify your algorithm’s correctness.

Problem 6. (20 points) You are given a set $P$ of $n$ points in the plane and a path $\pi$ that visits each point exactly once. (This path may self-intersect.) (See Fig. 2(a) and (b)).

(a) (10 points) Explain how to build a data structure from $P$ and $\pi$ of space $O(n)$ so that given any query line $\ell$, it is possible to determine in $O(\log n)$ time whether $\ell$ intersects the path. (For example, in Fig. 2(c) the answer for $\ell_1$ is “no,” and the answer for $\ell_2$ is “yes.”)

(b) (10 points) This is a generalization to part (a). Explain how to build a data structure from $P$ and $\pi$ so that given any line $\ell$, it is possible to report all the segments of $\pi$ that intersect $\ell$. The space should be at most $O(n \log n)$ and the query time should be at most $O(k \log^2 n)$, where $k$ is the number of segments reported. (Hint: Even if you did not solve (a), you can solve this by applying the data structure of part (a) as a “black box.”)