

CMSC 754: Computational Geometry

Fall 2005

<http://www.cs.umd.edu/class/fall2005/cmsc754/>

Introduction: This is an introductory course to computational geometry and its applications. We will discuss techniques needed in designing and analyzing efficient algorithms and data structures for computational problems in discrete geometry, such as convex hulls, geometric intersections, geometric structures such as Voronoi diagrams and Delaunay triangulations, arrangements of lines and hyperplanes, and range searching.

Instructor: Dave Mount. Office: AVW 3373. Email: mount@cs.umd.edu. Office phone: (301) 405-2704. Office hours: (To be announced.) I am also available immediately after class for questions. Feel free to send me email to arrange a time to meet with me.

Class Time: Tue, Thur 9:30-10:45. Room: CSI 1122.

Teaching Assistant: Minkyong Cho. Email: minkcho@cs.umd.edu. Minkyong will not hold regular office hours. If you have a question, please send him email to arrange a time to meet with him.

Text: (Required) *Computational Geometry: Algorithms and Applications* (2nd Edition), M. de Berg, M. van Kreveld, M. Overmars, O. Schwarzkopf, Springer-Verlag, 2000.

Other resources may be used and will be mentioned later in the semester.

Course Structure: This semester, CMSC 754 and CMSC 741 (Geometric and Solid Modeling) will be taught in conjunction with each other. These two courses share a number of topics, and so some of the material in CMSC 754 will be taught by Prof. Leila DeFloriani (and I will teach some of the topics in 741).

Prof. DeFloriani has agreed to allow CMSC 754 students to visit her during her office hours to ask questions regarding the material that she will be covering. (For information on her office hours, please visit the class web page for CMSC 741.) I will be responsible for all other elements of CMSC 754, such as the administration and grading of homeworks and exams.

Prerequisites: CMSC 451 (Algorithms), or its equivalent.

- Knowledge of basic algorithm design techniques, such as divide-and-conquer, greedy algorithms, dynamic programming,
- Knowledge of basic analysis techniques such as asymptotic notation, solving summations and recurrences, basic probability theory.
- Knowledge of basic data structures (e.g. balanced binary trees, heaps, and hashing).

Course Work: Course grades will be based on homeworks (roughly 5 of them) and a midterm and comprehensive final exam. The midterm is tentatively scheduled for Thu, Oct 27. The final exam will be on Fri, Dec 16, from 8:00-10:00am. (There will be no programming projects.) Tentative weights: Homeworks: 25%, Midterm: 30%, Final: 45%.

Normally, late homeworks will *not* be accepted. If you think that you have a legitimate excuse for handing in a homework late, contact me *before* the due date.

Academic Dishonesty: All class work is to be done independently. It is best to try to solve problems on your own, since problem solving is an important component of the course, and exam problems are often based on modifications of homework problems. You are allowed to discuss class material, homework problems, and general solution strategies with your classmates. But, when it comes to formulating or writing solutions you must work alone. You may use free and publically available sources, such as books, journal and conference publications, and web pages, as research material for your answers. (You will not lose points for using external sources.)

You *may not* use any service that involves payment, and you *must* clearly and explicitly cite all outside sources and materials that you made use of. I consider the use of uncited external sources as portraying someone else's work as your own, and as such it is a violation of the University's policies on academic dishonesty. Instances will be dealt with harshly and typically result in a failing course grade.

Topics: The following list of topics is very tentative. Depending on time, some topics may be added or dropped, and the order of topics may change.

Preliminaries: Basic Euclidean geometry.

Shapes: Basics of topology, complexes, manifolds and pseudo-manifolds.

Grids and Hulls: Fixed-radius near neighbors, convex hull algorithms, dominance and applications.

Linear Programming: Half-plane intersection and randomized LP, backwards analysis, applications of low-dimensional LP.

Intersections and Triangulation: Plane-sweep line segment intersection, triangulation of monotone subdivisions, plane-sweep triangulation of simple polygons.

Point Location: Kirkpatrick's method, trapezoidal decompositions and analysis, history DAGs.

Voronoi Diagrams: Basic definitions and properties, Fortune's algorithm.

Geometric Shape Representations: Topological relations in cell and simplicial complexes, edge-based data structures for cell complexes, data structures for general complexes.

Delaunay Triangulations: Point set triangulations, basic definition and properties, randomized incremental algorithm and analysis.

Arrangements and Duality: Point/line duality, incremental construction of arrangements and the zone-theorem, applications.

Geometric Approximation: Dudley's theorem and applications, well-separated pair decompositions and geometric spanners, VC dimension, ϵ -nets and ϵ -approximations,

Shape Simplification: Mesh and curve simplification, level of detail.

Geometric Retrieval: kd-trees, range trees, hereditary segment trees, nearest neighbor searching.

Homework 1: Convex Hulls and Grids

Handed out Tuesday, Sep 20. Due at the start of class Thursday, Sep 29. Late homeworks will not be accepted.

A general note 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 is preferred over technical detail. Nonetheless, be sufficiently complete that all elements are addressed, except for those that are obvious. (See my 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. The objective of this problem is to investigate the choice of grid size in the fixed-radius nearest neighbor problem in the plane. You are given a set of n points P in \mathbb{R}^2 and a radius value $r > 0$. Suppose that (through hashing) you have stored these points in a square grid of side length $g > 0$. (That is, a point with coordinates (x, y) is placed in the grid square with bucket index $b = [b_x, b_y]$ where $b_x = \lfloor x/g \rfloor$ and $b_y = \lfloor y/g \rfloor$.) You should not assume any relationship between r and g , e.g., it may be that $r \gg g$ or $g \gg r$.

- (a) Let b denote the index of some occupied bucket. Which buckets of the grid might contain a point that is within distance r of some point in b ? As a function of r and g , how many such buckets are there?

Hint: Coming up with an accurate answer is quite messy, especially if $g \ll r$. It is sufficient to present a square region that encloses all the buckets that need to be visited in the search.

- (b) Let us assume that your algorithm computes the distances between all pairs of points lying within bucket b and the points of the buckets described in your solution to part (a). Let D denote the total number of such distance calculations over all the points of P . Let k be the actual number of pairs (including duplicates, if you like) of points of P within distance r . Show that D can be bounded above by a linear function of n and k . Indicate clearly how this function depends on the values of g and r .

Hint: Be sure your analysis works if $r < \sqrt{2}g$. (Why does the proof given in class break down in this case?) I found it useful break the grid into a finer subgrid, just for the purposes of the analysis. In this analysis, I found the following fact to be useful.

First, we say that a function $f : \mathbb{R} \rightarrow \mathbb{R}$ is *convex* (or *concave up*) if for any x_1, x_2 , and $0 \leq \lambda \leq 1$,

$$f((1 - \lambda)x_1 + \lambda x_2) \leq (1 - \lambda)f(x_1) + \lambda f(x_2).$$

(For example $f(x) = x^2$ is convex.) Given a convex function f and a set of nonnegative values that sum to N , $x_1 + x_2 + \dots + x_m = N$, the value of $\sum_i f(x_i)$ is minimized when all the values are equal to N/m . This follows from *Jensen's inequality*, which states that for any convex function f ,

$$\frac{\sum_{i=1}^m f(x_i)}{m} \geq f\left(\frac{\sum_{i=1}^m x_i}{m}\right).$$

If you are not familiar with Jensen's inequality, it is well worth memorizing.

- (c) Derive the overall asymptotic running time of your algorithm, as a function of n , k , r , and g . In order for the algorithm to have a running time that is linear in n , what must be true about the relationship between r and g ? What does your analysis reveal about the best value to assign to g for a given value of r ?

Problem 2. Grid-based algorithms are used quite frequently to compute geometric approximations. The purpose of this problem is to develop an efficient grid-based algorithm for approximating the convex hull. We are given a set P of n points in the plane and an error parameter $\varepsilon > 0$.

The algorithm works as follows. First, it computes the point with the minimum and maximum x -coordinates of the points of P . Let p_{\min} and p_{\max} be these points and let x_{\min} and x_{\max} denote their respective x -coordinates. It then computes approximations to the upper and lower hull. We will describe the approximation to the upper hull only, since the lower hull is analogous. Partition the interval $[x_{\min}, x_{\max}]$ into $1/\varepsilon$ subintervals of equal size. Each subinterval represents a vertical slab. Bucket the points of P among these slabs. (Provide the formula for assigning a point to a bucket index.) Within each nonempty bucket, keep the point having the maximum y -coordinate. Let P' denote these points, together with p_{\min} and p_{\max} . Compute (by Graham's scan, say) the upper hull of P' and return this as the desired approximate upper hull.

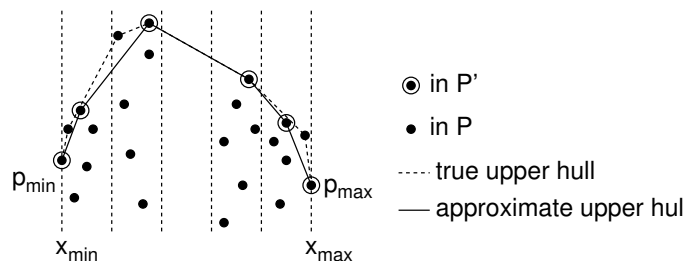


Figure 1: Approximate convex hull algorithm.

- (a) Flesh out the details of this algorithm. In particular, how are points assigned to buckets? What is the overall running time of the algorithm, as a function of n and ε ? (For asymptotic purposes, you should assume that $(1/\varepsilon)^c$ is smaller than n for any constant c , but $1/\varepsilon$ is not a constant.)
- (b) The approximate upper hull produced by this algorithm may be incorrect since some of the points of P may lie above this hull. Let $\Delta(P)$ denote the *diameter* of P , that is, the maximum distance between any two points of P . Prove that any point of P that lies above the approximate hull is within (perpendicular) distance of at most $\varepsilon\Delta(P)$ of the boundary of the approximate hull.

Problem 3. You are given a set of points P in the plane. Each point is tagged as being either red or blue. Let R and B denote the subsets of red and blue points, respectively. Let $n = |P| = |R| + |B|$. Given two points $r = (r_x, r_y) \in R$ and $b = (b_x, b_y) \in B$ we say that b dominates r if $b_x \geq r_x$ and $b_y \geq r_y$.

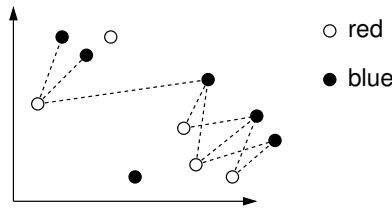


Figure 2: Dominating Pairs.

Give an efficient algorithm that counts all pairs (r, b) , where $r \in R$ and $b \in B$, where b dominates r . For example, in the figure above there are 10 instances of a blue point dominating a red point (indicated with broken lines). Your algorithm should run in time $O(n \log n)$, irrespective of the number of dominating pairs, which may be $\Omega(n^2)$. (Thus, it must count dominating pairs in groups rather than individually.) As always, justify the correctness of your algorithm and derive its running time.

Problem 4. Consider the 2-dimensional domain shown in the figure below left embedded in E^2 and the cell complex for this domain shown on the right. (Note that unshaded 2-cells are *not* part of the complex.) Vertices (0-cells) are assigned numbers, edges (1-cells) are assigned lower-case letters, and faces (2-cells) are assigned upper-case letters.

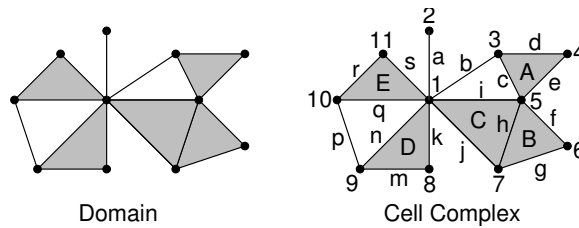


Figure 3: Cell complex.

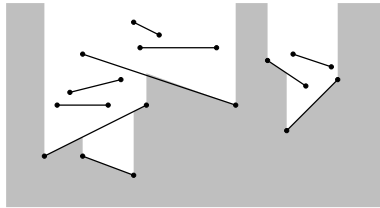
- Which cells (of all dimensions) form the combinatorial boundary of A , that is, what is $B(A)$?
- Which cells (of all dimensions) form the co-boundary (or star) of 5, that is, what is $*5$?
- Which cells (of all dimensions) form the link of 1, that is, what is $Lk(1)$?
- What are the top cells of the complex?
- Is this a simplicial complex? Explain.
- What is the maximum s such that this complex is s -connected? Explain.
- Is this complex regular? Explain.

Homework 2: Plane Sweep, Triangulation, and LP

Handed out Thursday, Sep 29. Due at the start of class Thursday, Oct 13. Late homeworks will not be accepted.

Note: Throughout the semester, when asked to give an $O(X)$ time algorithm, you may provide a randomized algorithm whose expected running time is $O(X)$.

Problem 1. Consider a set of n disjoint line segments in the plane $S = (s_1, s_2, \dots, s_n)$, where each segment s_i is defined by its left and right endpoints p_i and q_i . A point that lies on one of these segments is said to lie on the *lower envelope* if an open vertical ray shot downward from this point does not intersect any other segment. (The portions of the segments touching the shaded region in the figure below form the lower envelope.) Give a plane sweep algorithm that, given S , outputs the sequence of segments (or subsegments) that form the lower envelope. (If you like, you may assume that there is a special horizontal *sentinel segment*, s_0 , of infinite length that lies well above all the segments.)



Your algorithm should run in $O(n \log n)$ time. Be sure to explain the information stored along your sweep line status, the events that are processed, and the data structures used. You may make whatever general-position assumptions you like.

Problem 2. The purpose of this problem is to get some experience analyzing incremental algorithms. Rather than presenting a “real” algorithm, we present a toy example.

The randomized incremental linear programming (LP) algorithm given in class reduced the 2-dimensional LP to a 1-dimensional LP, which was then solved by a simple $O(n)$ time deterministic algorithm. However, had we solved the 1-d problem by applying the same incremental process, we would obtain something like the following (totally ridiculous) algorithm for producing the maximum of a set of numbers. Let us assume we are given an $A = \{a_1, a_2, \dots, a_n\}$ of real numbers.

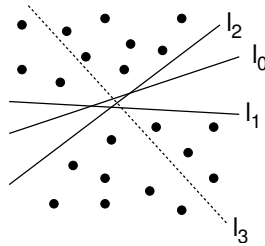
MadMax(A, n):

- (1) If $n = 1$ return a_1 .
- (2) Otherwise
 - (a) Select a random element $x \in A$.
 - (b) $x' \leftarrow \text{MadMax}(A - \{x\}, n - 1)$.
 - (c) If $x \leq x'$ return x' . (Clearly x is not the maximum of A .)
 - (d) Otherwise
 - (i) We “know” that x must be maximum of A , but just to be absolutely sure of this, compare x to the other $n - 1$ elements of A .
 - (ii) Return x .

Answer each of the following questions, and (especially in (b)) give a proof of your answer. You may assume that the elements of A are distinct.

- (a) What is the worst-case running time of MadMax?
- (b) What is the expected-case running time of MadMax, where the expectation is taken over the random choices made in step (2a)?

Problem 3. You are given a set of n points in the plane $P = \{p_1, p_2, \dots, p_n\}$ and a line $\ell_0 : y = a_0x - b_0$ that does not pass through any point of P . The line ℓ_0 partitions the points of P into those that lie above ℓ and those that lie below ℓ . We say that an arbitrary line $\ell : y = ax - b$ is *affinely equivalent* to ℓ_0 relative to P if it partitions the points of P in exactly the same way as ℓ_0 . For example, in the figure below, the lines ℓ_1 and ℓ_2 are affinely equivalent to ℓ_0 , whereas ℓ_3 is not. The objective of this problem is to design an efficient data structure that allows us to determine whether a given query line ℓ is affinely equivalent to ℓ_0 relative to P .



- (a) Given P and ℓ_0 , consider the region formed by the dual images of all the lines ℓ that are affinely equivalent to ℓ_0 relative to P . Describe (in terms of the duals of the points of P and the dual of ℓ_0) what this region looks like. (For example: Is it a polygon? Is it convex? Is it bounded or unbounded? What are its vertices? What are its edges?) Prove your assertions by appealing to the basic properties of point-line duality.

To simplify your answer, you may assume that no vertical line is affinely equivalent to ℓ_0 . (Although you might consider how removing this assumption would affect your answer.)

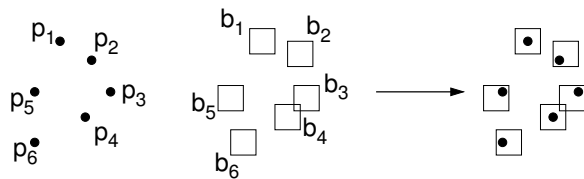
- (b) Given P and ℓ_0 , describe a data structure which, given any line $\ell : ax - b$, determines in $O(\log n)$ time whether ℓ is affinely equivalent to ℓ_0 . (Note that this is a static data structure in the sense that P and ℓ_0 remain constant throughout the lifetime of the data structure. Only the query line changes.) Show that your data structure has $O(n)$ size and can be built in $O(n \log n)$ time.

Hint: No fancy 2-dimensional data structures are needed. In fact, I needed nothing fancier than a 1-dimensional array.

Problem 4. The following problem arises in point pattern matching and image registration in computer vision. You are given a sequence of n points $P = \langle p_1, p_2, \dots, p_n \rangle$ in the plane and sequence of n closed, axis-aligned rectangles $B = \langle B_1, B_2, \dots, B_n \rangle$. Describe $O(n)$ time algorithms to solve each of the following problems.

(Hint: Reduce this to determining the feasibility of an LP problem in a space of constant dimension.)

- (a) Does there exist a translation vector, $t = (t_x, t_y)$ such that for $1 \leq i \leq n$, the translated point $p_i + t$ lies within the corresponding rectangle B_i ? (You may assume that the rectangles B_i are represented in any manner that is convenient to your solution.)



(b) An *affine transformation* T is given by six coefficients, a_{11} , a_{12} , etc. and maps a point $p = (p_x, p_y)$ to the point:

$$T(p) = (a_{11}p_x + a_{12}p_y + a_{13}, a_{21}p_x + a_{22}p_y + a_{23}).$$

(Note that a translation is just a special case where $a_{11} = a_{22} = 1$ and $a_{12} = a_{21} = 0$.) Generalize your solution to (a), to determine whether there exist an affine transformation that maps each point p_i of P to lie within the corresponding rectangle B_i .

In both cases, it is important that the correspondences between points and boxes is given in advance. (Otherwise, the problem is much harder.)

Homework 3: Point Location and Voronoi Diagrams

Handed out Thu, Oct 13. Due at the start of class Tue, Oct 25. Late homeworks will not be accepted.

Note: Throughout the semester, when asked to give an $O(X)$ time algorithm, you may provide a randomized algorithm whose expected running time is $O(X)$.

Problem 1. We stated in class that the maximum number of parabolic arcs on the beach line is $2n - 1$. Present an example of n points in the plane such that at some point in the sweeping process the beach line contains exactly $2n - 1$ arcs. Your example should be sufficiently general that it is clear how to extend it to arbitrarily large values of n .

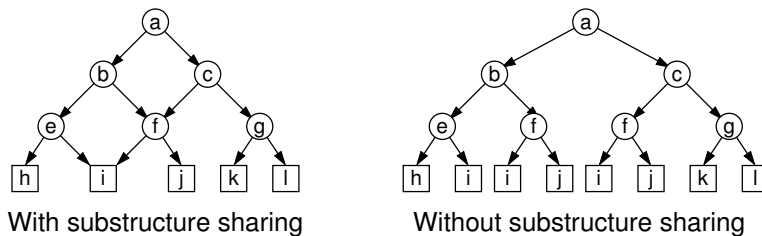
Problem 2. Let P be a set of n points in the plane. Give an $O(n \log n)$ time algorithm that computes, for each point $p \in P$ the closest point to p among all the other points of P .

Problem 3. In the early 1970's the Ford Pinto was an infamous automobile that had a high risk of exploding in rear-end collisions. Consider a set of n Pintos driving to the east along a straight road. Let p_i denote the starting point of the i th Pinto at time $t = 0$, and let v_i denote its speed and assume that $v_i > 0$. At time t this Pinto is located at position $p_i + t \cdot v_i$. If two Pinto's collide, the one with the lower velocity is hit from behind and explodes and the other survives and continues without any change in its velocity.

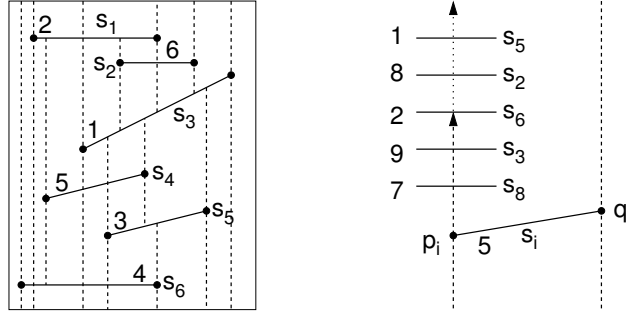
Show how to build a data structure of $O(n)$ space in $O(n \log n)$ time that can answer the following queries in $O(\log n)$ time each. Given a query point at position q_0 along the road at some time t_0 , which is the first Pinto (if any) that will pass position q_0 at some future time, and at what time will this event occur?

Hint: Reduce this to a 2-dimensional point location problem in in the plane. Show how to construct a planar subdivision consisting of $O(n)$ line segments in $O(n \log n)$ time. Prove the correctness of your reduction.

Problem 4. Recall that the history DAG used in the planar point location algorithm given in class used substructure-sharing to keep the space down to $O(n)$. The objective of this problem is to consider what the space requirements would be had we not used substructure sharing. Consider the following alternative data structure. Start with the initial history DAG, but every time a substructure is shared by two or more nodes, create separate copies of the substructures and store pointers to these individual structures. Repeat this process recursively so that the *history DAG* is converted into a *history tree*. (This is illustrated in the figure below.)



We will show that the expected space needed for the history tree is $O(n \log n)$, averaged over all random permutations π . Let $S = \{s_1, \dots, s_n\}$ be a set of n segments, and let π be a random permutation of $\{1, \dots, n\}$. Suppose that these segments are inserted into a trapezoidal map in the order $\langle s_{\pi_1}, s_{\pi_2}, \dots, s_{\pi_n} \rangle$. We first show that the leaves of the history tree correspond to a refinement of the standard trapezoidal map. Define the *refined trapezoidal map* for S and π , denoted $\mathcal{T}(S, \pi)$, as follows. For each line segment s_{π_i} , shoot a bullet path vertically both upwards and downwards until it first hits a segment s_{π_j} , such that $\pi_j < \pi_i$. (See the figure below. The integer label associated with each segment is its position in the insertion order.)



Refined trapezoidal map
Insertion Order: (s3, s1, s5, s6, s4, s2)

- Show that if the segments are inserted in the order π then there is a 1–1 correspondence between the leaf nodes of the history tree and the trapezoids of the refined trapezoidal map $\mathcal{T}(S, \pi)$.
- For each segment s_i , let p_i and q_i denote its left and right endpoints. Let $U(p_i)$ denote the set of segment of S that intersect the ray shot vertically upwards from p_i , and let $D(p_i)$ be the analogous set for the bullet path shot downwards from p_i . (For example, in the above figure on the right $U(p_i) = \{s_8, s_3, s_6, s_2, s_5\}$.) Let $U(p_i, \pi)$ denote the subset $U(p_i)$ that are fragmented by the upward vertical bullet path from p_i until first hitting a segment of lower index than p_i in π . (For example, in the figure, s_i has index of 5, fragments s_8 and s_3 before stopping at s_6 , whose index is 2, and so $U(p_i, \pi) = \{s_8, s_3\}$.) Define $D(p_i, \pi)$ analogously for the downward bullet path. Define $U(q_i, \pi)$ and $D(q_i, \pi)$ analogously for q_i . Define

$$C(S, \pi) = \sum_{i=1}^n (|U(p_i, \pi)| + |D(p_i, \pi)| + |U(q_i, \pi)| + |D(q_i, \pi)|),$$

to be the total number of fragmentations caused by all the bullet paths. Prove that the total number of trapezoids of $\mathcal{T}(S, \pi)$ is at most $C(S, \pi) + O(n)$.

- Next, let us consider the probability that a bullet path fragments a given segment. Let p_i be the endpoint of a segment s_i of S (left or right) and let $s_j \in U(p_i)$. Define $u(p_i, s_j)$ to be number of segments that lie vertically between p_i and s_j . (For example, in the figure above, $u(p_i, s_2) = 3$ since it includes the segments s_8, s_3 , and s_6 .) Define $d(p_i, s_j)$ analogously for the downward ray and define $u(q_i, s_j)$ and $d(q_i, s_j)$ analogously for the other endpoint of s_i . Show that, for a random permutation π , the probability that the bullet path shot from p_i fragments the segment s_j is at most $1/(2 + u(p_i, s_j))$.

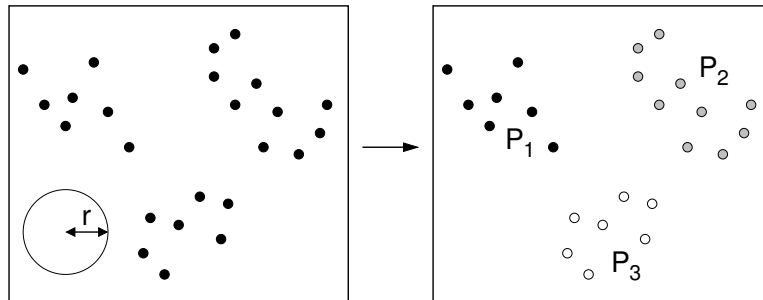
- (d) Using (c), prove that the expected number of segments fragmented by an arbitrary bullet path (either upwards or downwards) from any segment endpoint (left or right), is $O(\log n)$. (Hint: Use the Harmonic series.)
- (e) Combining all the results from parts (a)–(d), prove that, for a random permutation π , the expected number of leaf nodes in the history tree, is $O(n \log n)$.

Homework 4: Delaunay Triangulations and Arrangements

Handed out Fri, Nov 11. Due at the start of class Thu, Dec 1. Late homeworks will not be accepted. **Note:** I have pushed the due date back for this homework to accommodate the holiday. The last homework will be handed out Nov 29 and due Dec 13, so please plan accordingly.

Problem 1. For this problem, let P denote a set of n points in the plane. Recall that a *partition* of P is a collection of pairwise disjoint subsets whose union is P .

- (a) Consider any partition of P into two or more subsets and let p_i and p_j be the pair of points of different subsets that are of minimum distance. Show that the segment $\overline{p_i p_j}$ is an edge of the Delaunay triangulation of P .
- (b) Describe an $O(n \log n)$ time algorithm for the following problem. Given the planar point set P and a distance threshold $r > 0$, partition P into a maximum number of disjoint subsets P_1, P_2, \dots, P_m such that for all $1 \leq i < j \leq m$ the minimum distance between any point of P_i and P_j is strictly greater than r . (See the figure below.)



Problem 2. You are given a set of n line segments $S = \{s_1, \dots, s_n\}$ in the plane, where $s_i = \overline{p_i q_i}$. Present an $O(n^2)$ time and space algorithm that computes a line ℓ that intersects the greatest number of segments of S . (You may make the usual general position assumptions.)

Problem 3. The objective of this problem is to consider a two-dimensional version of a common optimization problem in the areas of classification and support-vector machines in learning theory, where we want to find a line that best separates two sets of points in the plane.

Let B be a set of n blue points and let R be a set of n red points in the plane. Given a nonvertical line ℓ , let ℓ^+ and ℓ^- denote the halfplanes lying above and below ℓ , respectively. We define ℓ 's *separation defect* to be

$$\delta(\ell) = \min (|B \cap \ell^+| + |R \cap \ell^-|, |B \cap \ell^-| + |R \cap \ell^+|).$$

Given two planar point sets B and R , explain how to compute a line ℓ that has the minimum separation defect in $O(n^2)$ time and space. (To avoid dealing with special cases in which the line passes through the a point of R or B , you may restrict your algorithm to considering lines that do not pass through any point.)

Problem 4. The objective of this problem is to present an alternative (and, I think, cuter) proof of the zone theorem. The proof is based on a *bead counting game*. Let L be a set of n lines, let $\mathcal{A}(L)$ denote its arrangement. Let ℓ be any other line, and let $Z_{\mathcal{A}}(\ell)$ be the zone of ℓ in $\mathcal{A}(L)$. Remember that the complexity of the zone is defined to be the sum of all the edges of all its faces of the arrangement that intersect ℓ . (We do not consider ℓ to be part of the arrangement, and so the fact that it appears to split some of these edges is not counted. Note however that an edge of the arrangement that intersects ℓ should be counted twice, once for the left face and once for the right face.)

Let us assume that ℓ is horizontal. Consider for now just the left-bounding edges of $Z_{\mathcal{A}}(\ell)$ that lie above ℓ . (See Lecture 15 for definitions.) Place a single bead on each such edge. Here are the rules of the game, and they are illustrated in the figure below.

Rule 1: If this edge intersects ℓ , move the bead to the point at which the edge intersects ℓ .

Rule 2: If the edge does not intersect ℓ , then let ℓ' denote the line of L supporting this edge, and let ℓ'' be the line whose intersection with ℓ' forms the lower vertex of this edge. Move the bead to the intersection of ℓ'' with ℓ .

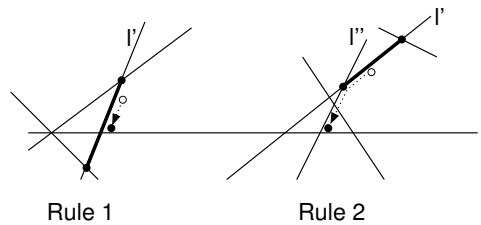


Figure 1: Bead moving rules.

- Show that after applying Rules 1 and 2 (just to the left-bounding edges lying in the portion of the zone above ℓ) the total number of beads that can reside at the intersection of ℓ with any other line of the arrangement is at most 2.
- Using (a), show that the total number of edges in the zone of ℓ is at most $8n$.
- Explain how to tighten the argument to obtain an upper bound of $6n$ on the total number of edges of the zone. (Hint: Some edges are counted multiple times in (b).)
- Give an example to show that, for arbitrarily large n , the number of edges on the portion of the zone lying above ℓ can be as high as $4n - O(1)$. (As a challenge, try to get $6n - O(1)$ total edges on the arrangement, that is, counting the edges of the zone lying both above and below ℓ . I must admit that I do not see how to do it, although I can get $5n - O(1)$.)

Homework 5: WSPDs, Spanners, and Simplification

(Revised version, updated Dec 4.) Handed out Tue, Nov 29. Due at the start of class Tue, Dec 13. Late homeworks will not be accepted.

Problem 1. The following problem is intended to address a question raised in class. Suppose that we expressed the well separated decomposition explicitly, by listing all the elements of each set of the WSPD. Would the total size of this representation still be $O(n)$?

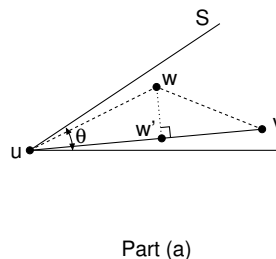
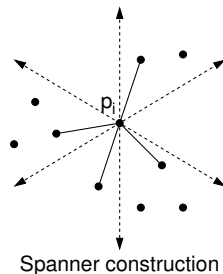
- (a) Show that for arbitrarily large n and for any fixed separation factor s , there exists a set of n points P in the plane such that the following property holds: Let $\{\{A_1, B_1\}, \{A_2, B_2\}, \dots, \{A_m, B_m\}\}$ be the s -well separated pairs generated for P by the algorithm given in class. For each point $p \in P$, let $m(p)$ denote the number of pairs $\{A_i, B_i\}$ in which p is involved, that is, $p \in A_i \cup B_i$. Then the average value of $m(p)$ over all point of P is $\Omega(\log n)$. That is

$$\frac{1}{n} \sum_{p \in P} m(p) = \Omega(\log n),$$

where the hidden constant factor depends on s and d . (Hint: In my example, every point of p is involved in $\Omega(\log n)$ pairs.) (Note: I do not know whether $\Omega(n \log n)$ is the worst case. If you can come up with an example that has a higher average value, please let me know.)

- (b) Based on your result from part (a), show that if we had represented the WSPD by explicitly enumerating the individual elements of the sets A_i and B_i , for $1 \leq i \leq m$, the total space requirements would be $\Omega(n \log n)$ rather than $O(n)$, where the constant factor depends on s and d .

Problem 2. The goal of this problem is to describe an alternative way of constructing a spanner. Let P be a set of n points in the plane and let $t > 1$ be the desired stretch factor. Let $k = \lceil 2\pi/\theta \rceil$, where θ is chosen such that $1/(\cos \theta - \sin \theta) = t$. Surround each point p_i of P with a collection of k sectors, where the angle of each sector is at most θ , and for each of these k sectors, create an edge from p_i to its closest point within this sector. Let $G(t)$ be the resulting graph. (See the figure below on the left for one point p_i .)



- (a) To analyze the spanner properties of this graph, we first prove a lemma. Let u be a point, and let S be a sector of angle θ (defined above) whose apex is at u . Let v and w be points within S , such that w is closer to u than v (that is, $\|uw\| \leq \|wv\|$). Prove that

$$\|uw\| + t\|wv\| \leq t\|uv\|.$$

(Hint: This reduces to basic trigonometry. It may help to consider the point w' , which is the orthogonal projection of w onto the segment uv . See the figure.)

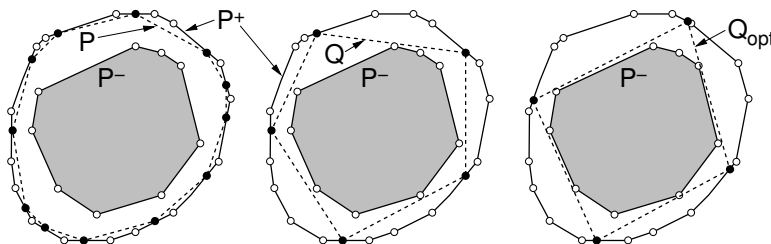
- (b) Prove by induction on the length of paths that $G(t)$ is a t -spanner for P . As part of your solution, explain how to construct a spanner path between any two points of P in $G(t)$. (Hint: Use part (a) to prove the spanner bound.)
- (c) Show that for all $t > 1$, the number of edges of $G(t)$ is $O(n)$. Assuming that $t = 1 + \varepsilon$, for some small $\varepsilon > 0$, explain how the hidden constant varies as a function of ε .

Problem 3. A desirable property of spanners is that every vertex should have constant degree.

- (a) Show that there exists a set of points in the plane such that the WSPD-based spanner construction given in class results in at least one vertex of degree $\Omega(n)$. That is, the degree is at least cn for some $c > 0$, which might depend on the dimension and separation factor, but not on n . (Note: The algorithm for constructing the quadtree did not specify how representatives are chosen in the quadtree, and for this problem you are free to select the representatives for each WSPD in any manner you like.)
- (b) Show that it is possible to modify the WSPD-based spanner construction so that the number of edges is the same, but each vertex has constant degree (where the constant depends on the stretch factor). For this, you may make use of the following simplifying assumptions: (1) every internal node of the quadtree has at least two nonempty children (that is, there are two children whose cells contain at least one point), and (2) the side length of the child's cell is half the side length of the parent's cell (that is, we did not need to apply compression). (Hint: Show that in such a tree it is possible to distribute representatives so that each point occurs as a representative for at most a constant number of internal nodes.)

Problem 4. A natural question that arises in mesh simplification is whether the mesh that results after simplification is in some sense “minimal,” that is, for a given error tolerance, this mesh has the fewest number of simplices. This problem is very hard to answer even for 2-dimensional meshes, and many problems in this area are open. In the 1-dimensional case (polygonal curves) it is possible to prove minimality results for many formulations. We will consider one.

For this problem we consider the following simple situation. Consider two convex polygons P^- and P^+ , called the *inner* and *outer* polygons, respectively, where P^- is contained within P^+ . Suppose we are given a convex polygon P that is nested between these two polygons (lying outside interior P^- and inside the closure of P^+), such that every vertex of P lies on the boundary of P^+ . (See the figure below, left.) Our goal is to simplify P to form a convex polygon, using a subset of the vertices of P , such that the simplified polygon is nested between P^- and P^+ .



We use the following simple heuristic, which simplifies P in stages. If P has three vertices or fewer we stop. Otherwise, we select an arbitrary vertex v_i of P . We consider the result if we were to remove

v_i and join its neighbors v_{i-1} and v_{i+1} by an edge. (Indices wrap around cyclically.) If this edge lies outside the interior of P^- then v_i is removed, and otherwise the operation is ignored, and we go on to another vertex. If this operation cannot be performed on any of the remaining vertices, the entire process terminates. Let Q denote some polygon that results through the application of this process, and let k denote the number of sides of Q . Clearly Q is nested between P^- and P^+ .

Let Q_{opt} denote a convex polygon having a minimum number of sides that is nested between P^- and P^+ , and let k_{opt} denote the number of sides it has. Unlike Q , the vertices of Q_{opt} need not be taken from the vertices of P , and need not lie on the boundary of P^+ . Show that the above algorithm produces nearly a factor-2 approximation to the optimum by proving that $\lfloor k/2 \rfloor \leq k_{opt}$. (Hint: Show that every two consecutive vertices of Q can be charged to a single vertex of Q_{opt} , so that no vertex of Q_{opt} is charged more than once.)

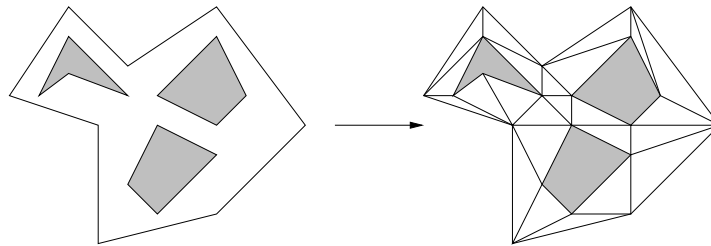
Sample Problems for the Midterm Exam

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 (Leila's lectures and order- k Voronoi diagrams, for example) and are not reflected in this list.

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 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.

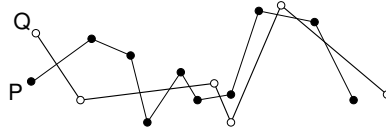
- In the plane-sweep algorithm for line segment intersection, how did we determine which intersection points would be stored in the event queue?
- In the analysis of the randomized incremental point location we argued that the expected depth of a random query point is at most $12 \ln n$. Based on your knowledge of the proof, where does the factor 12 arise? (Hint: It arises from two factors. You can explain either for partial credit.)
- 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.
- Recall the orientation primitive $\text{Orient}(a, b, c)$, 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.)
- 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?



- 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.

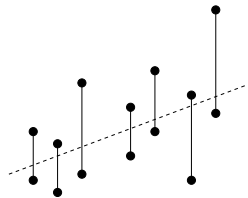
Problem 2:

- (a) 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.

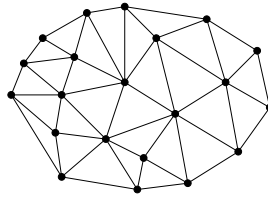


- (b) Does the bound proved in (a) still apply if the two polygonal chains that are monotone with respect to different directions (e.g., one monotone with respect to x and the other monotone with respect to y)? Justify your answer.

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. 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 5. 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.

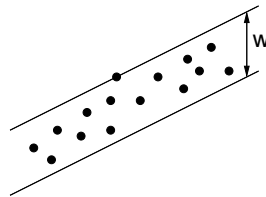
Problem 6. Consider the following randomized incremental algorithm, which computes the smallest rectangle (with sides parallel to the axes) bounding a set of points in the plane. This rectangle is represented by its lower-left point Lo and the upper-right point Hi .

- (1) Let $P = \{p_1, p_2, \dots, p_n\}$ be a random permutation of the points.
- (2) Let $Lo[x] = Hi[x] = p_1[x]$. Let $Lo[y] = Hi[y] = p_1[y]$.
- (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.)

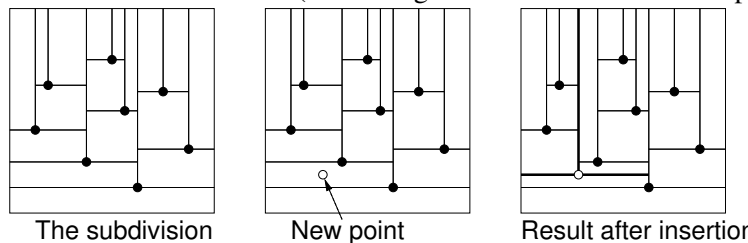
Problem 7. Define a *strip* to be the region bounded by two (nonvertical) parallel lines. The *width* of a strip is the vertical distance between the two lines.

- (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 8. 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 $O(1)$.

CMSC 754: Midterm Exam

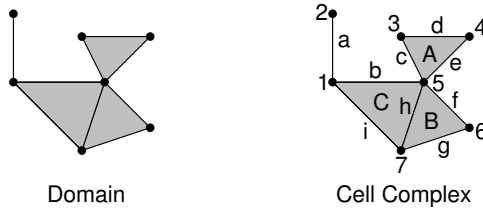
This exam is closed-book and closed-notes. You may use one sheet of notes, front and back. Write all answers in the exam booklet. If you have a question, either raise your hand or come to the front of class. Total point value is 100 points. Good luck!

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. Give a short justification for each answer.

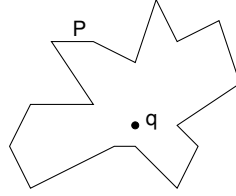
- (a) You are given a convex polygon in the plane having n_c sides and an x -monotone polygon having n_m sides. What is the maximum number of intersections that might occur between the edges of these two polygons? (You may use O -notation.)
- (b) Recall that a winged-edge data structure requires 1 pointer per vertex, 8 pointers per edge, and 1 pointer per face. Assume that this data structure is used for representing a triangulation in the plane having V vertices, E edges, and F faces. Show that the maximum number of pointers used by this is at most $c \cdot V$, for some constant c . What is the value of c ? Explain how you derived your answer.
- (c) 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. (15 points) Consider the 2-dimensional domain shown in the figure below left embedded in the plane and the cell complex for this domain shown on the right.



- (a) Which cells (of all dimensions) form the combinatorial boundary of C , that is, what is $B(C)$?
- (b) Which cells (of all dimensions) form the co-boundary (or star) of 1, that is, what is $*1$?
- (c) Which cells (of all dimensions) form the link of 5, that is, what is $Lk(5)$?

Problem 3: (15 points) 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 the figure below.) Suppose that P is given as a counterclockwise sequence of its vertices $\langle v_1, v_2, \dots, v_n \rangle$. Show that it is possible to determine in linear time whether P is star-shaped in $O(n)$ time. Prove the correctness of your algorithm.



Problem 4: (25 points) You are given a set of n lines in the plane $L = \{\ell_1, \ell_2, \dots, \ell_n\}$. Each line ℓ_i is represented by its slope a_i and y -intercept b_i , that is, it satisfies the equation $y = a_i x + b_i$. In class we showed that it is possible to compute the lower envelope of these lines in $O(n \log n)$ time.

- (a) Suppose that the lines of L have been given in increasing order of slope ($a_1 < a_2 < \dots < a_n$). Show that it is possible to compute the lower envelope of L in $O(n)$ time.
- (b) Suppose that the lines of L have been given in increasing order of y -intercept ($b_1 < b_2 < \dots < b_n$). Is it possible to compute the lower envelope of L in $O(n)$ time? Explain your answer.

Problem 5: (20 points) Let $P = \{p_1, p_2, \dots, p_n\}$ be a set of points in the plane. Let $\delta(P)$ denote the distance between the two closest points of P , that is, $\delta(P) = \min_{i \neq j} \|p_i p_j\|$. Computing $\delta(P)$ in $O(n)$ time is a challenging problem, but your task here is simpler. Given P and a positive target value $s > 0$, give an $O(n)$ time algorithm that determines whether s is less than, equal to, or greater than $\delta(P)$. (There is no partial credit for an $O(n \log n)$ solution. For partial credit, show how to test for one of these three conditions in $O(n)$ time.)

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 were not covered in previous semesters (Leila's lectures, order- k Voronoi diagrams, WSPDs and spanners, and VC-dimension, for example) and are not reflected in this list.

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) What is the (exact) maximum number of intersections between the edges of two convex polygons, where each polygon has n vertices?
- (b) 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.
- (c) What is the (exact) maximum number of arcs that can appear on the beach line in Fortune's Voronoi diagram algorithm for n sites?
- (d) What is the (asymptotic) maximum number of faces of a convex hull of n points in dimension d ?
- (e) Given n points in 3-space, if we preprocess these points using a *kd-tree*, how quickly can we count the number of points that lie within a given axis-parallel 3-dimensional rectangle? How much space does the data structure need?

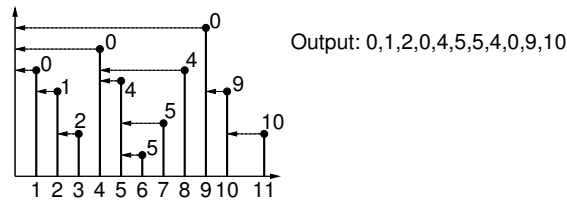
Problem 2. More short answer questions.

- (a) Given a k - d tree with n points in the plane, what is the (asymptotic) maximum number of cells that might be stabbed by a line that is not axis-parallel? Explain briefly.
- (b) What is a *zone* in an arrangement? Given an n -line arrangement A in the plane and given an arbitrary line ℓ , what is the (asymptotic) maximum complexity (number of edges) of the zone of ℓ relative to A ? (*No explanation needed.*)
- (c) Which of the following statements regarding the Delaunay triangulation (DT) of a set of points in the plane are true? (*No explanation needed.*)
 - (i) Among all triangulations, the DT minimizes the maximum angle.
 - (ii) Among all triangulations, the DT maximizes the minimum angle.
 - (iii) Among all triangulations, the DT has the minimum total edge length.
 - (iv) The largest angle in a DT cannot exceed 90 degrees.
- (d) Let P be a set of n points in the plane, and let h denote the number of points of P that lie on the convex hull of P . As a function of n and h , how many edges and triangles are there in the Delaunay triangulation of P ? Show how you derived your answer.

Problem 3. 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.)

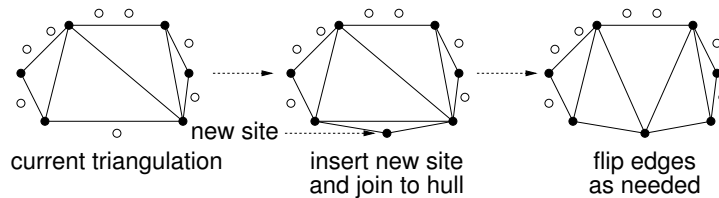
The input is a sequence of top endpoints of each segment $p_i = (x_i, y_i)$, for $1 \leq i \leq n$, which are sorted in increasing order by x -coordinate. The output is the sequence of indices, one for each segment.

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



Problem 4. Let $P = \langle p_1, p_2, \dots, p_n \rangle$ be the vertices of a convex polygon, given in counterclockwise order. The purpose of this problem is to modify the randomized incremental Delaunay triangulation algorithm to run in $O(n)$ expected time.

Unlike the algorithm presented in class, we *do not* put all the sites within a large bounding triangle. We start with the triangle defined by three random sites from P . The remaining sites are inserted in random order, each new site is connected to the convex hull, and edge flips are performed until the circumcircle condition is satisfied. (See the figure below.)



- Give a backwards analysis to show that the expected number of edge-flips performed with each insertion is $O(1)$.
- Whenever a new site is added, we need to determine where along the current convex hull it is to be added. Explain how to do this is $O(1)$ expected time. You are allowed preprocess the sites in $O(n)$ time. (Hint: The result of Problem 3 may be useful.)

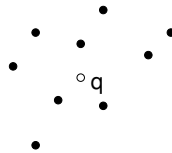
Problem 5. The Euclidean metric is but one way to measure distances. As mentioned in class, one of the others is the L_∞ metric, which is defined

$$dist_\infty(p, q) = \max(|p_x - q_x|, |p_y - q_y|).$$

- Given a point q , describe the set of points that are at L_∞ distance w from q .

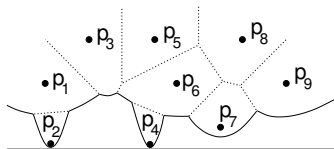
- (b) Given two points distinct p and q , describe the set of points that are equidistant from p and q in the L_∞ distance. (Hint: The bisector is a polygonal curve. As a function of the coordinates of p and q , indicate how many vertices this curve has and where these vertices are located. In contrast usual practice, I would like to have the exact formula, and not a high-level description.)
- (c) Argue that the Voronoi diagram for n sites using the L_∞ metric has $O(n)$ edges and vertices. (Hint: First show that each cell of the Voronoi diagram is connected. Take care in your application of Euler's formula, since some vertices may have degree two.)
- (d) Assuming that the L_∞ Voronoi diagram is given, describe a method to compute the largest empty axis-aligned square centered at a query point q . Your method should involve $O(n)$ space and $O(\log n)$ query time. (You are not required to show how to build the data structure, only to argue its existence.)
- (e) A Voronoi cell is *unbounded* if it extends to infinity. Which points of P have unbounded L_∞ Voronoi cells? Explain.
- (f) Recall that the beach line in Fortune's algorithm consists of parabolic segments. Suppose we were to modify Fortune's algorithm to construct the L_∞ Voronoi diagram. Describe the structure of the beach line in this case: What is its shape? How many segments does it contain in the worst-case (as a function of n)? Is it monotone with respect to the x -axis?

Problem 6. 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 shown in the figure below 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.)



- (a) Show that for all (sufficiently large) n there exists a set of n points that has 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 7. 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 $\{p_1, p_2, \dots, p_n\}$ 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 the figure below the string would be $p_1 p_2 p_1 p_3 p_6 p_4 p_6 p_7 p_9$.)



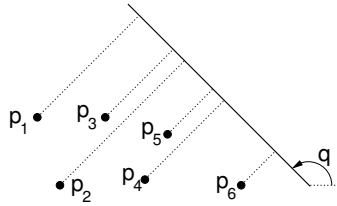
- (a) Prove that for any i, j , the following alternating subsequence cannot appear anywhere within such a string:

$$\dots p_i \dots p_j \dots p_i \dots p_j \dots$$

- (b) Prove that any string of n distinct symbols that does not contain any repeated symbols ($\dots p_i p_i \dots$) 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*.

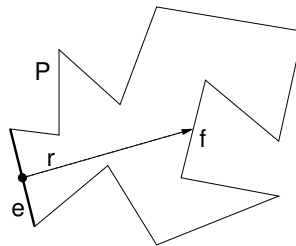
Problem 8. In class we showed (assuming general position) that an arrangement of n lines in the plane has $\binom{n}{2}$ vertices, n^2 edges and $\binom{n}{2} + n + 1$ faces. Generalize this to give the number of vertices, edges, faces, and 3-dimensional cells in an arrangement of n planes in 3-space. (Assume general position.)

Problem 9. Given a set P of n points in the plane, and given any slope θ , project the points orthogonally onto a line whose slope is θ . 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 the figure below, for the slope θ the permutation would be $\langle p_1, p_3, p_2, p_5, p_4, p_6 \rangle$.



- (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 10. 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: Use duality to reduce this to a point location query.)



CMSC 754: Final Exam

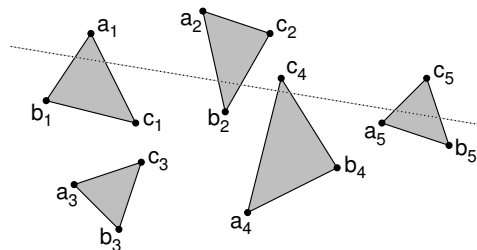
This exam is closed-book and closed-notes. You may use two sheets of notes, front and back. Write all answers in the exam booklet. If you have a question, either raise your hand or come to the front of class. Total point value is 150 points. Good luck!

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 $O(T(n))$.

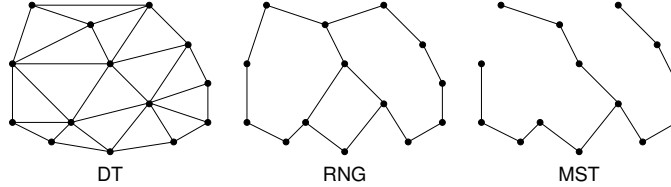
Problem 1: (30 points; 5-10 points each) Give a short answer (a few sentences) to each question.

- (a) 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.
- (b) Recall that the minimum weight triangulation for a set of points is a triangulation that minimizes the sum of the edge lengths. Give an example of a planar point set in general position such that the minimum weight triangulation is not equal to the Delaunay triangulation. Explain briefly. (Hint: This can be done with four points that are nearly cocircular.)
- (c) 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.
- (d) The *Hausdorff distance* between one point set P and another point set Q is defined in terms of (select one):
 - (i) the **maximum** of point-to-point distances
 - (ii) the **sum** of point-to-point distances
 - (iii) the **sum of squares** of point-to-point distances
 - (iv) the **square root of the sum of squares** of point-to-point distances

Problem 2. (15 points) You are given a set of n triangles in the plane $\mathcal{T} = \{T_1, \dots, T_n\}$, where triangle T_i has vertices $\langle a_i, b_i, c_i \rangle$. Present an algorithm that computes a line ℓ that stabs the greatest number of triangles of \mathcal{T} . (You may assume general position.) For example, in the figure below 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 3. (30 points) The Delaunay triangulation and minimum spanning tree (MST) are two examples of a straight-line planar graphs defined on a set P of n points in the plane. Another example is the *relative neighborhood graph* (or RNG). Its vertices are the points of P . There is an edge joining two points p_i and p_j if there is no point $p_k \in P$ that is simultaneously closer to p_i and p_j than they are to each other. That is, there is no p_k such that $\max(\|p_i p_k\|, \|p_j p_k\|) < \|p_i p_j\|$.



- (a) (15 points) Prove that the RNG of P is a subgraph of the Delaunay triangulation of P .
- (b) (15 points) Prove that the MST of P is a subgraph of the RNG of P .

Problem 4. (30 points) 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) (5 points) 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) (10 points) 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. (Hint: Consider two cases. First, when the convex hull contains 3 points and when all 4 points are on the convex hull. In each case indicate which subset(s) cannot be generated.)
- (c) (15 points) 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. (Hint: This can be done by applying a data structure that we have given in class.) 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 5. (25 points) Consider a set P of point sites in the plane in general position.

- (a) (10 points) Prove that a site $p \in P$ has a nonempty cell in the *farthest-point Voronoi diagram* if and only if p is on the convex hull of P .
- (b) (15 points) Suppose that P has n sites and that h of these lie on P 's convex hull. As a function of n and h , what is the maximum number of vertices, faces, and edges in the farthest point Voronoi diagram of P ? (Give three answers.) You should consider that the unbounded edges are all connected to a single vertex at infinity. Explain how you got your answers.

Problem 6. (20 points) 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 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.