CMSC 754: Computational Geometry
Fall 2002
`http://www.cs.umd.edu/~mount/754/`

Instructor: Dave Mount. Office: AVW 3209. Email: `mount@cs.umd.edu`. Office phone: (301) 405–2704. Office hours: Mon 3:00–4:00, Wed 3:30–4:30. I am also available immediately after class for questions. Feel free to send me email if you cannot make these times to set up another time.

Teaching Assistant: Justin Wan. Office: AVW 1151. Email: `ycwan@cs.umd.edu`. Office hours: Mon 2:30–3:30, Thu 5:00–6:00.

Class Time: Tue, Thur 2:00-3:15. Room: CSI 1122.

Course Objectives: This is an introductory course to computational geometry and its applications. We will discuss techniques needed in designing and analyzing efficient algorithms for problems in geometry, including convex hulls, geometric intersections, Voronoi diagrams, Delaunay triangulations, geometric data structures, and motion planning.

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

Prerequisites: CMSC 451, 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 and heaps).

Course Work: There will be roughly 5 homework assignments. 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. There will be a midterm and comprehensive final exam. The midterm is tentatively scheduled for Thu, Oct 31. The final exam will be Thu, Dec 19, 10:30am–12:30pm. There will be no programming projects. Approximate weights: Homeworks: 25%, Midterm: 30%, Final: 45%.

Academic Dishonesty: All class work is to be done independently. You are allowed to discuss class material, homework problems, and general solution strategies with your classmates. When it comes to formulating/writing/programming solutions you must work alone. If you make use of other sources in coming up with your answers you *must* site these sources clearly (papers or books in the literature, friends or classmates, information downloaded from the web, whatever).

It is best to try to solve problems on your own, since problem solving is an important component of the course. But I will not deduct points if you make use of outside help, provided that you cite your sources clearly. Representing other people's work as your own, however, is plagiarism and is in violation of university policies. Instances of academic dishonesty will be dealt with harshly.

Topics: The following list of topics is very tentative.

Preliminaries: Basic Euclidean geometry.

Convex Hulls: Graham's scan, QuickHull, Chan's algorithm.

Intersections: Plane-sweep and line segment intersection, representation and intersection of planar subdivisions.

Polygon Triangulation: Art gallery problems, monotone partitions.

Linear Programming: Half-plane intersection, incremental method, applications.

Range Searching: kd-trees, range trees, fractional cascading.

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

Voronoi Diagrams: Post office problem, divide and conquer algorithm, randomized incremental algorithm, history graphs, Fortune's algorithm.

Arrangements and Duality: Point/line duality, incremental construction of arrangements, levels, applications.

Delaunay Triangulations: Triangulations of point sets, properties of the Delaunay triangulation, incremental construction.

Motion planning: Shortest paths, visibility graphs, roadmaps.

Approximation Methods: Dudley's theorem and applications, well-separated pair decompositions, spanners.

CMSC 754:Fall 2002 Dave Mount

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) In the plane-sweep algorithm for line segment intersection, how did we determine which intersection points would be stored in the event queue?

(b) Given an orthogonal range tree in the plane for a set of n points, what is the total size of all the auxiliary trees? Explain briefly.

(c) Given n disjoint segments stored in a segment tree, if only the standard lists are considered (not the hereditary lists), what is the total space needed for the tree in the worst case? Explain briefly.

(d) Given a set of n points distributed uniformly within a circle in the plane, in terms of average case efficiency, which of the following algorithms would be better for computing the convex hull of the points: Graham's scan or Jarvis's march? Explain briefly.

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

Problem 2: (25 points)

(a) (15 points) You are given two x-monotone polygonal chains P and Q with a total of n vertices between then. Prove that P and Q can intersect at most $O(n)$ times.

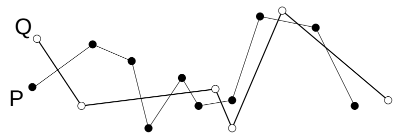


Figure 1: Problem 2.

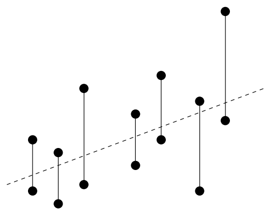
(b) (10 points) Does the bound proved in (a) still apply if the two polygonal chains that are monotone with respect to different directions (e.g., one monotone with respect to x and the other monotone with respect to y)? Justify your answer.

Problem 3: (15 points) For this problem, assume for simplicity that n is a power of 2. Consider a kd-tree for a set of n points in 3-space, which is built according the standard splitting method described in class:

- The cutting dimension *cycles* among x, y, and z.
- Each time a node is split, its points are partitioned *evenly* into sets of equal sizes.

Consider a line that is parallel to the x-axis. What is the maximum number of leaf cells that can be *stabbed* (that is, intersected) by such a line. Express your answer using O-notation, as a function of n. Show how you derived your answer.

Problem 4: (15 points) You are given a set of n vertical line segments in the plane. Present an efficient an algorithm to determine whether there exists a line that intersects all of these segments. An example is shown in the figure below. (Hint: $O(n)$ time is possible.) Justify your algorithm's correctness and derive its running time.


Problem 4

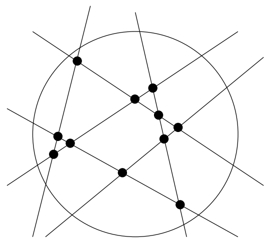

Problem 5

Figure 2: Problems 4 and 5.

Problem 5: (20 points) You are given a set of n lines that intersect a circle C. Describe an $O(n \log n)$ time algorithm that counts all the pairs of lines that intersect within the circle. You may assume that the point at which a line intersects a circle can be computed in $O(1)$ time. Justify your algorithm's correctness and derive its running time. (If you cannot do this, for half credit, show how to report the intersections in $O(k + n \log n)$ time, where k is the number of intersections.)

CMSC 754: Final Exam

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

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

Problem 1: (20 points) Short answer questions. No explanations are needed, but may be given for partial credit. Asymptotic quantities may be given for partial credit.

(a) As an (exact) function of n, what is the minimum number of guards needed to guard any n-sided simple polygon?

(b) What is the (exact) maximum number of intersections between the edges of two convex polygons, where each polygon has n vertices?

(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?

Problem 2: (20 points) More short answer questions. Give short explanations where requested.

(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 an 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.

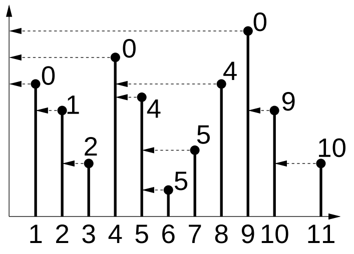
(iv) The largest angle in a DT cannot exceed 90 degrees.

(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: (10 points) 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.)

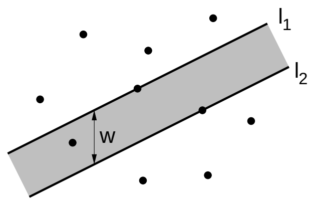
The input is a sequence of top endpoints of each segment $p_i = (x_i, y_i)$, for $1 \leq i \leq n$, which are *sorted in increasing order* by x-coordinate. The output is the sequence of indices, one for each segment.

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



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

Problem 4: (20 points) Given a set of n points P in the plane, a *3-corridor* for P is a pair of parallel lines such that there are 3 points of P lying on or between these parallel lines. Define the *vertical width* of a strip to be the vertical distance between the lines. Throughout, assume that no three points of P are collinear.



(a) Suppose that ℓ_1 and ℓ_2 define a 3-corridor in the (primal) plane for P whose vertical width is w. What is the corresponding structure in the dual plane? Briefly justify your answer.

(b) Prove that if ℓ_1 and ℓ_2 define the 3-corridor for P of *minimum* vertical width, then one of the lines ℓ_1 or ℓ_2 passes through two points of P. (You may prove your result in either the primal or dual plane and may assume general position.)

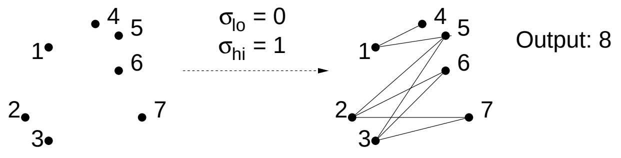
(c) Describe an $O(n^2)$ time algorithm to compute the 3-corridor of minimum vertical width for P.

(d) Define the *perpendicular width* of a corridor to be the perpendicular distances between the lines. True or false: Does the assertion of (b) (either line passes through two points) apply in this case as well? Explain briefly.

Problem 5: (10 points) You are given a set $P = \{p_1, \ldots, p_n\}$ of n distinct points in the plane. You may assume no two points have the same x-coordinate. Each pair of distinct points p_i and p_j defines a line whose slope is $\sigma_{i,j} = (p_{j,y} - p_{i,y})/(p_{j,x} - p_{i,x})$. Describe an efficient algorithm, which given P and two values σ_{lo} and σ_{hi}, returns the *number* of pairs $\{i, j\}$, such that

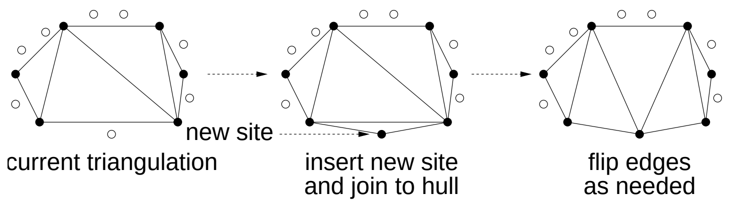
$$\sigma_{lo} \le \sigma_{i,j} \le \sigma_{hi}.$$

For example, in the figure below, we show the pairs whose slopes are between $\sigma_{lo} = 0$ and $\sigma_{hi} = 1$. It is possible to solve this problem in $O(n \log n)$ time.



Problem 6: (20 points) Let $P = \langle p_1, p_2, \ldots, p_n \rangle$ 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.)



(a) Give a backwards analysis to show that the expected number of edge-flips performed with each insertion is $O(1)$.

(b) 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.)

CMSC 754:Fall 2002 Dave Mount

Homework 1: Convex Hulls and Neighbors

Handed out Thursday, Sep 12. Due at the start of class Tuesday, Oct 1. 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, a justification of its correctness, and a derivation of its running time. Giving careful proofs can be quite cumbersome in geometry, and so you are encouraged to use intuition and give illustrations whenever appropriate. However, beware that a poorly drawn figure can make certain erroneous hypotheses seem "intuitively correct," and can lead you to ignore special cases that may not be apparent in the figure. Please do not give detailed C++ or Java code. Only provide sufficient detail to convince the grader and me that your method is correct and complete. Try to keep your answers concise.

Unless otherwise stated, you may assume that 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. Generalize the bucket-based fixed-radius near neighbor problem to the plane. That is, given a set of points P in the plane, and a distance $r > 0$, report the set of all pairs (p, q) for $p, q \in P$ such that the Euclidean distance from p to q is at most r. In particular do the following:

(a) Present an algorithm for this problem, focusing in particular on the size of the grid, how points are bucketed, and which buckets are visited in the search. (You may assume that any needed data structures for the hashing and list manipulation have been provided to you.)

(b) Establish that the number D of distance computations performed by your algorithm is within a constant factor of the number k of pairs reported. Derive an specific constant bound (e.g., $D \leq 23k$). Based on this, show that the running time of your algorithm is $O(n + k)$.

Problem 2. The objective of this problem is to investigate one approach for an $O(m+n)$ time algorithm to determine whether one convex polygon P lies inside of another convex polygon Q. Here m and n denote the number of vertices of P and Q, respectively. (There are other algorithms for doing this, but I would like you to consider this particular approach.)

(a) Given the leftmost vertex v_1 of Q, show how to use orientation tests to compute the counterclockwise point of tangency of P relative to v_1, denoted t_1. (More formally, P lies on and to the left of the directed line $\overline{v_1 t_1}$. Such a line is called a *supporting line* for P. See Fig. 1.) Show that your method runs in $O(m)$ time.

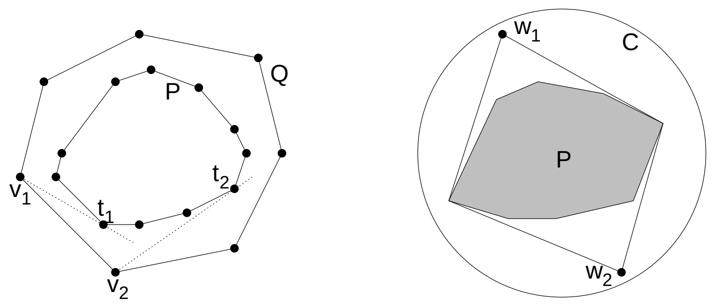
(b) Assuming that the CCW tangent point t_{i-1} has already been computed for vertex v_{i-1} of Q, show how to compute the tangent point t_i for the next vertex v_i in CCW order about Q.

(c) Prove that as the tangent points are being computed, it is possible to determine whether P is not contained Q.

(d) Assuming that P is contained within Q, show that all of these tangent points can be computed in $O(m+n)$ time.

You may design your algorithms in (a) and (b) so that they terminate if it is discovered that P is not contained in Q.

Problem 3. Given a convex polygon P and a point w that is external to P, we say that a point q on the boundary of P is *visible* to w if the open line segment $\overline{wq}$ does not intersect P. Let P denote an n-sided convex polygon enclosed within a closed circular disk C. Present an $O(n)$ time algorithm that, given P and C, determines whether there exist two points w_1 and w_2 lying within C but outside of P, such that every point on the boundary of P is visible to either w_1 or w_2 (or both). If so your algorithm should output such a pair of points. (See the Fig. 1.)



Problem 2. Problem 3

Figure 1: Problems 2 and 3.

(a) First show that if two such points exist, then they may be assumed to lie on the boundary of C.

(b) Explain how to subdivide the boundary of C into circular arc intervals such that all the points within each interval "see" the same edges of P. Show that this can be done in $O(n)$ time.

(c) Using (a) and (b) present an $O(n)$ time for the problem.

Extra Credit Problem:

Note: Extra credit points are not considered when grade cutoffs are determined, and hence there is no penalty for not working them. However, if your final grade is near to a cutoff point, I may take extra credit points into consideration to increase the final letter grade to the next higher grade.

Given a set P of n distinct points in the plane, devise an $O(n)$ expected time algorithm which computes the minimum distance $r_{\min}$ between any two points of P, and reports the two points that achieve this distance.

You can do this by using grids and buckets and solving the following problems.

(a) Show that if you are given a candidate distance r, then in $O(n)$ expected time it is possible either to compute $r_{\min}$ exactly in $O(n)$ time or, failing this, to determine which one of the following two cases holds:

  (i) $r < r_{\min}$,

  (ii) $r > r_{\min}$.

(b) Show that in case (ii), there is a simple test that allows you to determine a subset of point of P whose closest point is farther away than $r_{\min}$. These points may be eliminated from further consideration.

(c) (Hard) Combine parts (a) and (b) to devise an algorithm that searches for the value of $r_{\min}$. The algorithm first chooses a candidate distance r. The test of part (a) is applied to determine whether case (ii) applies. If so, it uses (b) to eliminate some subset of the remaining points.

(d) Assuming each execution of parts (a) and (b) takes time proportional to the number of points remaining, how many points must be eliminated in each stage in order for the overall running time to be $O(n)$? (You can answer this even if you don't get part (c).)

You may not make any assumption about the number of bits of precision used to store the coordinates, but you may assume that all arithmetic operations on points run in $O(1)$ time. (If you assume constant precision, then there is an easy $O(n)$ time algorithm.)

Homework 2: Polygons and Triangulations

Handed out Tuesday, Oct 1. Due at the start of class Thursday, Oct 17. Late homeworks will not be accepted.

Problem 1. In computer graphics, triangulations are used to represent surfaces. Graphics libraries like OpenGL support a special type of triangulation called a *strip*, which is defined as follows. For $n \geq 3$, given a sequence of vertices $V = \langle v_1, v_2, \ldots, v_n \rangle$, the strip of V consists of the $n-2$ triangles $\langle v_i, v_{i+1}, v_{i+2} \rangle$, for $1 \leq i \leq n-2$. (See Fig. 1.)

Assume that you are given an n-sided, x-monotone, simple polygon P as a sequence of vertices in counterclockwise order. Present an algorithm that determines whether P can be triangulated as a strip of the above form, such that its leftmost vertex is v_1, and its rightmost vertex is v_n. Justify the correctness and derive the running time of your algorithm. (Warning: I found a simple algorithm, but its proof of correctness was quite involved.)

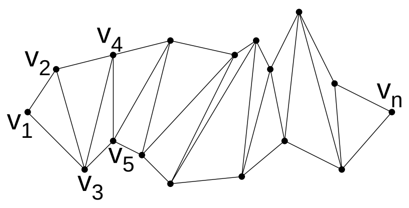


Figure 1: Problem 1.

Problem 2. Sometimes rather than computing a path that has minimum length, it is desirable to compute a polygonal path with the minimum number of segments. Consider the following situation. We are given an x-monotone n-sided polygon P. The lower chain is a convex chain. The upper chain is an arbitrary x-monotone polygon (see Fig. 2). Let s and t be the leftmost and rightmost vertices of P. The objective is to find a polygonal chain from s to t that lies within P that consists of the fewest number segments.

(a) Show that such a path may be assumed to be x-monotone.

(b) A point $p \in P$ is said to be *tangential* if the two tangents drawn from p to the lower chain of P lie entirely within P. (In Fig. 2 right, p is tangential, but q does not.) Prove that the set of points within P that are tangential forms a monotone polygon.

(c) Give an $O(n)$ time algorithm to compute the polygonal region of all tangential points within P. (Hint: It may be easier to do this in two stages, one for left-tangential points and one for right-tangential points. Use plane sweep.)

(d) Using the results of (c), give an $O(n)$ time algorithm which computes the polygonal path from s to t having the fewest number of segments. (Hint: First prove that such a path should need never pass through a nontangential point.)

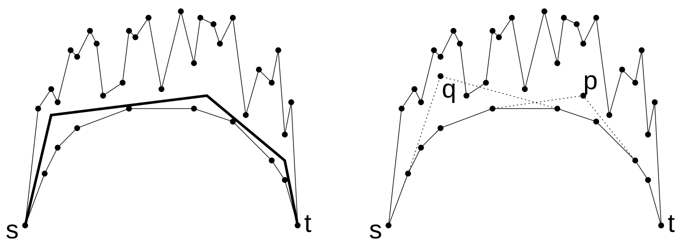


Figure 2: Problem 2.

Problem 3. You are given a pair of vertical lines $x = x_0$ and $x = x_1$, which define a vertical strip in the plane. You are also given a set L of n nonvertical lines, $\ell_i : y = a_i x + b_i$. Present an $O(n \log n)$ time algorithm that *counts* the number of intersections of the lines of L within this strip. (Thus, the output consists of a single number.) You may make whatever general position assumptions are convenient.

Extra Credit Problem: Determining whether a general graph has a spanning tree of degree at most three is known to be NP-complete. In this problem, we will consider this problem for planar triangulations. Given a connected graph $G = (V, E)$, a *spanning subgraph* is a connected subgraph $G' = (V', E')$ such that $V' = V$ and $E' \subseteq E$. A *spanning tree* is an acyclic, spanning subgraph.

Recall that a *triangulation* of a point set V is a PSLG whose vertex set is V, whose external face consists of the edges of the convex hull of V, and whose internal faces are triangles. Prove that any triangulation of a set of points in the plane (viewed as a planar graph) has a spanning tree of degree at most three. (See Fig. 3, left.)

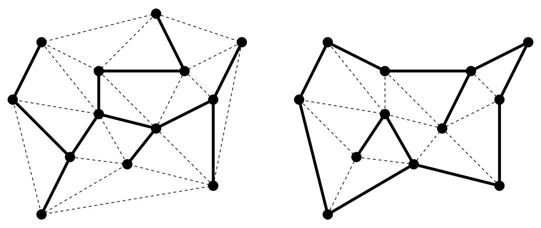


Figure 3: Extra Credit Problem.

Hint: First prove the following by induction on the number of vertices. Consider any PSLG triangulation whose external face is homeomorphic to a circle. Such a graph is characterized by three conditions: its internal faces are triangles, there are at least three vertices on its external face, and every vertex on its external face has exactly two neighbors on the external face. Prove that such a PSLG has a spanning subgraph of degree at most three such that every edge of the external face is in the spanning subgraph. (See Fig. 3, right.)

CMSC 754:Fall 2002 Dave Mount

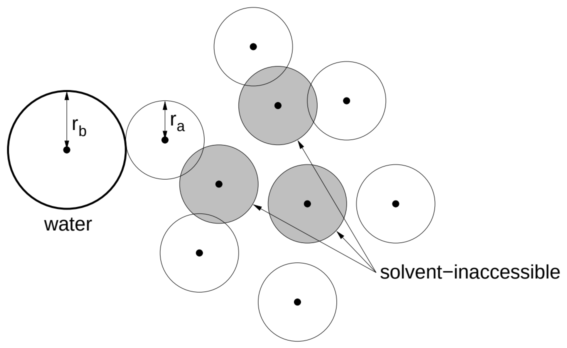
Homework 3: Voronoi Diagrams and Delaunay Triangulations

Handed out Thursday, Nov 14. Due at the start of class Tuesday, Dec 5. (Note that there will be a short Homework 4, which will be handed out Dec 3 and due Dec 12, so plan accordingly.) Late homeworks will not be accepted.

Problem 1. 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, 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 other atoms P. In the figure below, all atoms are solvent-accessible except for three (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 2. In this problem we will investigate the properties of a variant of the Voronoi diagram, in which we replace the notion of "nearest" with "farthest." Given a set $P = \{p_1, p_2, \ldots, p_n\}$ of point sites and site $p_i \in P$, recall that we defined $\text{Vor}_P(p_i)$ to be the set of all points in the plane that are strictly closer to p_i than to any other site of P. Define $\text{Far}_P(p_i)$ to be the set of all points in the plane that are strictly farther from p_i than to any other site of P. As we did with Voronoi diagrams, we define the *farthest-point Voronoi diagram* to be the union of the boundaries of $\text{Far}_P(p_i)$ for all points in P. Let *NVD*(P) denote the *nearest-point Voronoi diagram* of P and let *FVD*(P) denote the *farthest-point Voronoi diagram* of P.

(a) Show that $\text{Far}_P(p_i)$ is a convex polygon (possibly unbounded).

(b) Unlike the nearest-point Voronoi diagram, a site may not contribute a region to the farthest point diagram. Show that $\text{Far}_P(p_i)$ is nonempty if and only if p_i is a vertex of the convex hull of P. (Hint: Recall that a site p_i is on the boundary of P's convex hull if and only if there is a halfplane containing P whose boundary line passes through p_i.)

(c) Draw a picture of three points in the plane. Show (in different colors) *NVD*(P) and *FVD*(P). Label each region according to which sites are closest and farthest.

(d) Repeat (c), but this time for four points, such that only three of the four points are on the convex hull.

(e) Repeat (d), but this time for all four points on the convex hull. (Try to avoid degeneracies, such as four cocircular points or parallel sides.)

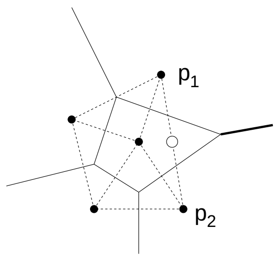
(f) Recall the minimum enclosing disk problem from Lecture 10. Show that it is possible to solve this problem in deterministic $O(n \log n)$ time, by appealing to the structure of *FVD*(P). (Hint: Recall that there were two possible cases for the minimum enclosing disk.) You may assume that *FVD*(P) can be computed in $O(n \log n)$ time, and that each element of the diagram (face, vertex, edge) is labeled with its farthest sites.

Problem 3. We start with two (seemingly unrelated) definitions. A triangulation is *acute* if the angles of all its triangles are less than 90 degrees. A Voronoi diagram is said to be *medial* if each edge of the diagram contains in its interior the midpoint of the two defining sites for the edge. For example, the diagram in the figure below left fails to be medial because the Voronoi edge between p_1 and p_2 (indicated with a heavy line) fails to contain the midpoint between its two sites (the white circle).

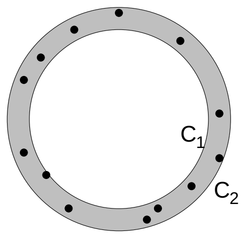
(a) Prove that an acute triangulation of a set of points is always Delaunay, that is, it satisfies the empty circumcircle property.

(b) Prove that if a Voronoi diagram is medial, then the corresponding Delaunay triangulation is acute.

(Hint: It may be useful to review the facts about chords and angles mentioned in Lecture 18.)



Problem 3



Challenge Problem

Challenge Problem: Given a set P of n points in the plane, the objective of this problem is to design an algorithm that determines the pair of concentric circles C_1 and C_2 such that every point of P lies between these circles, and the area of the region between these two circles is as small as possible. (Note: There is a variant of this problem in which the goal is to minimize the width of this region. I have explicitly chosen area rather than width because it is easier prove the following properties.)

Consider the subdivision S formed by overlaying *NVD*(P) and *FVD*(P). The vertices of S consist of the union of the vertices of the two Voronoi diagrams together with the points at which two edges, one from each diagram, intersect each other.

(a) Prove that the common center of the optimum concentric circles cannot lie in the interior of a face of S.

(b) Prove that the common center of the optimum concentric circles cannot lie in the interior of an edge of S.

(c) Using (a) and (b), give an $O(k + n \log n)$ algorithm for solving this problem, where k is the number of vertices of S. (You may assume that *NVD*(P) and *FVD*(P) can be computed in $O(n \log n)$ time each.)

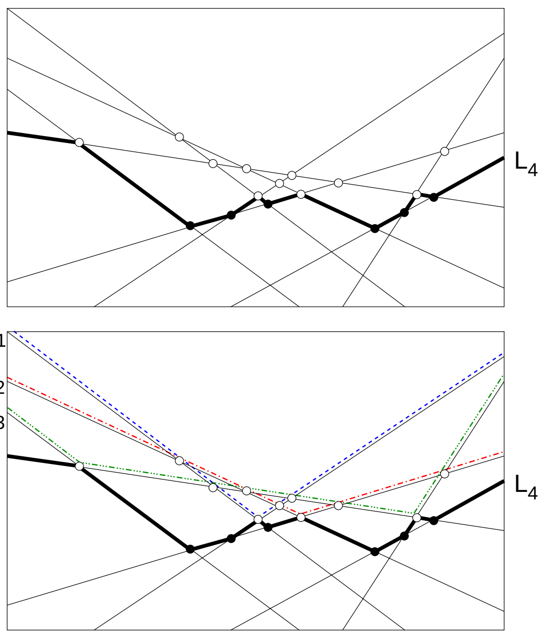
CMSC 754:Fall 2002 Dave Mount

Homework 4: Arrangements

Handed out Tuesday, Dec 3. Due at the start of class Thursday, Dec 12. Late homeworks will not be accepted.

Problem 1. Consider an arrangement $A(L)$ of a set L of lines in the plane. Recall that for $1 \leq k \leq n$, the kth level, denoted $\mathcal{L}_k$ is the set of points such that there are at most $k-1$ lines strictly above this point and at most $n-k$ lines strictly below this point. (See the figure below.)

An important combinatorial question is how many vertices of the arrangement lie on or above the kth level. In this problem we will prove that the number of such vertices is $O(nk^2)$. (This is far from tight. The best known bound to date is $O(nk^{1/3})$.) For example, in the top part of the figure below we consider the case $k = 3$. The edges of the first three levels lie strictly above level $\mathcal{L}_4$. There are 11 vertices shown as white circles belonging to levels 1 through 3 (there are a couple of others that occur outside the figure boundary).



The proof is based on showing the existence of k unbounded convex polygons, $c_1, c_2, \ldots, c_k$ that cover all the edges of the first k levels. Throughout we assume that the lines are in general position. For $1 \leq i \leq k$, the polygon c_i is defined as follows. Start at $x = -\infty$ with the line having the ith smallest slope. Trace this line from left to right until hitting a vertex of level $\mathcal{L}_{k+1}$. At this vertex make a left turn. We continue in this manner until reaching $x = +\infty$. This results in a collection of k x-monotone curves. (This is shown in the bottom part of the figure above. The figure is a bit misleading, because c_3 has a smaller slope than c_2, and so these labels should be reversed.) Prove the following facts about these curves.

(a) Each curve c_i is the boundary of an (unbounded) convex polygon.

(b) No two curves c_i and c_j share a common edge.

(c) The union $c_1 \cup c_2 \cup \ldots \cup c_k$ covers all the edges and vertices of levels 1 through k.

(d) It follows from (b) and (c) that the number of vertices of levels 1 through k is equal to the total number of vertices and intersection points between the k convex curves $c_1, c_2, \ldots, c_k$. Prove that this number is at most $O(nk^2)$.

Problem 2. 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 line $\ell : y = ax - b$. The answer to the query is "yes" if there are at most k points of P lying either above or below ℓ, and "no" otherwise. You may assume for simplicity that ℓ does not pass through any points of P. Your data structure should be able to answer queries in $O(\log n)$ time and should use $O(n)$ space. Preprocessing time should be $O(n^2)$. (Hint: Use duality and the results of Problem 1(d).)

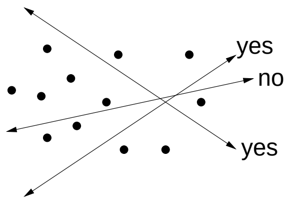


Figure 1: An example for Problem 2, for $k = 3$.

Challenge Problem: Consider the following two problems:

Zero-sum: Given three sets A, B, and C, each containing n numbers, determine whether there are three numbers, $a \in A$, $b \in B$, and $c \in C$ such that $a + b + c = 0$.

Collinear triple: Given a three sets of n points in the plane P, Q, and R, do there exist three points $p \in P$, $q \in Q$, $r \in R$, such that p, q, and r are collinear?

(a) Present an $O(n^2)$ time algorithm for the Zero-sum problem. (It is believed that $O(n^2)$ is optimal in the algebraic computation model.)

(b) Present an $O(n^2)$ time algorithm for Collinear triple problem. (You may *not* assume in general position. For example, three or more points of P may be collinear.)

(b) Show how to reduce the Zero-sum problem to the Collinear triple problem in $O(n)$ time. In particular, given three sets of numbers (A, B, C) for the Zero-sum problem, in $O(n)$ time produce three sets of points (P, Q, R) for the Collinear triple problem, such that (A, B, C) has a zero-sum triple if and only if (P, Q, R) has a collinear triple. This implies that if there were a faster algorithm for the Collinear triple problem then there would be a faster algorithm for the zero-sum problem.

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 (fixed-radius near neighbors and hereditary segment trees) 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. Short answer questions.

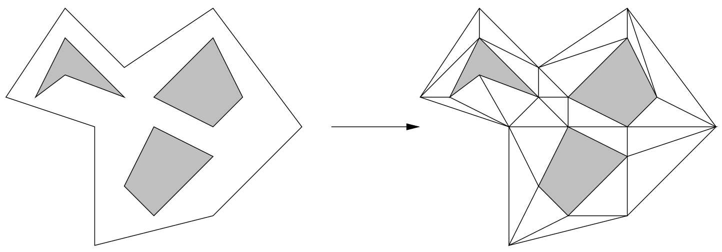
(a) Draw a picture of a simple polygon and a set of guards, such that the guards can see every point on every edge of the polygon, but the guards cannot see every point in the interior of the polygon.

(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 $\mathit{Orient}(a, b, c)$, 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) Recall that Chan's algorithm partitioned the point subsets of sizes $m = 2^{2^t}$ for $t = 1, 2, \ldots$. Suppose that instead we iterate through values of the form $m = 2^t$. What would the running time of the algorithm be in this case?

(e) 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?

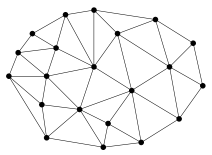


(f) The worst-case running time for the randomized LP algorithm is $O(n^2)$. Suppose you are given a fixed set of n halfplanes and an objective function. Is there always a way to order these halfplanes so that the algorithm requires at least $\Omega(n^2)$ time? If so, explain how to construct such an ordering. If not, then give an example (for arbitrary n) in which all orderings lead to a lower running time.

(g) 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. 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 as a DCEL, but you do *not* have to give the details of how the DCEL is constructed. (Hint: There is a simple plane sweep algorithm.)

Problem 3. 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 $O(n)$ 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.

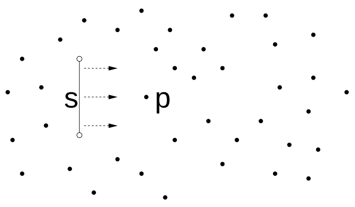
Problem 4. Consider the following randomized incremental algorithm which computes the smallest rectangle (with sides parallel to the axes) bounding a set of points in the plane. This rectangle is represented by its lower-left point *Lo* and the upper-right point *Hi*.

```
(1) Let $P = \{p_1, p_2, \ldots, p_n\}$ be a random permutation of the points.
(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 5. We showed that a kd-tree could answer orthogonal range queries for a set of n points in the plane in $O(\sqrt{n} + k)$ time, where k was the number of points reported. Generalize this to show that in dimension 3, the query time is $O(n^{2/3} + k)$.

Problem 6. You are given a set P of n points in the plane, which you may preprocess into any data structure in order to answer the following query efficiently: Given any *vertical* line segment, determine the first point of P (if any) that would be hit if we were to translate the line segment from its initial position horizontally to the right. (For example, in the figure below, if s were the query segment, then p would be the answer.) Explain your data structure and the query processing. Analyze its space, preprocessing time and query time.



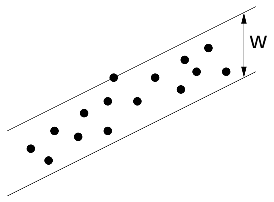
Problem 7. Given two points $p_1 = (x_1, y_1)$ and $p_2 = (x_2, y_2)$ in the plane, we say that p_2 *dominates* p_1 if $x_1 \leq x_2$ and $y_1 \leq y_2$. Given a set of points $S = \{p_1, p_2, \ldots, p_n\}$, a point p_i is said to be *maximal* if it is not dominated by any other point of S.

(a) Let $(x_1, y_1), (x_2, y_2), \ldots, (x_k, y_k)$ be a list of the maximal points in S, such that $x1 < x_2 < \ldots < x_k$. Prove that $y_1 > y_2 > \ldots > y_k$.

(b) Give an $O(n \log n)$ algorithm (either deterministic or randomized) for computing the a list of maximal points for S, sorted by increasing x-coordinates. (You may assume that no two points have the same x- or y-coordinates.) Explain your algorithm and its analysis.

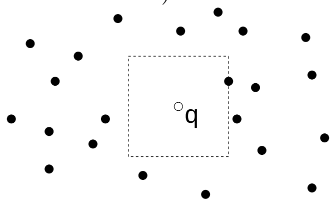
Problem 8. Define a *strip* to be the region bounded by two (nonvertical) parallel lines. The *width* of a strip is the vertical distance between the two lines.

(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.



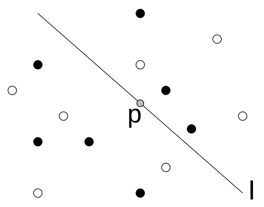
(b) Give an $O(n)$ time algorithm, which given a set of n points in the plane, finds the nonvertical strip of minimum width that encloses all of these points.

Problem 9. Given a set P of n points in the plane, design a structure to answer the following queries efficiently. Given a point q find the largest empty square centered at q. See the figure below. There are a number of different solutions. Your algorithm should run in $O(\log^2 n)$ time and $O(n \log n)$ space. (A faster solution can be given for extra credit.)



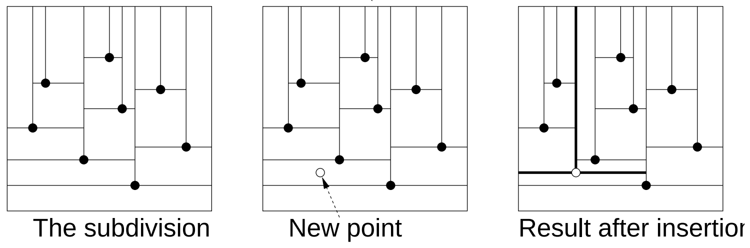
Problem 10. You are given two sets, red points, R, and blue points, B, both having the same number of elements, n. Let p be any point in the plane (not in R or B).

(a) Prove that there exists a line ℓ passing through p, such that the number of points of R that lie above ℓ is equal to the number of points of B that lie above ℓ. (This number need *not* be $n/2$.)



(b) Assuming part (a), give an $O(n \log n)$ algorithm that outputs such a line.

Problem 11. 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:Fall 2002 Dave Mount

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.

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. Short answer questions. Keep explanations short (a few sentences at most).

(a) Given n points in 3-space, if we preprocess these points using a *range 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? (No explanation needed.)

(b) Suppose that you have a set of polyhedral obstacles in 3-space. As in the plane, the visibility graph consists of all pairs of obstacle vertices (including s and t) that are visible to each other. True or false: The shortest path from s to t in 3-space travels along edges of the 3-d visibility graph. Explain briefly.

(c) 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.

(d) Let P and Q be two point sets in the plane, and let $p \in P$ and $q \in Q$ be the two points from these sets that minimize $dist(p, q)$. True or false: $\overline{pq}$ is an edge of the Delaunay triangulation of $P \cup Q$. Explain.

Problem 2. 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

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

(a) 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.)

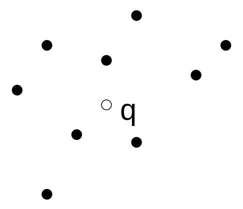
(c) Argue that the Voronoi diagram for n sites using the L_∞ metric has $O(n)$ edges and vertices. (Hint: First show that each cell of the Voronoi diagram is connected. Take care in your application of Euler's formula, since some vertices may have degree two.)

(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 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 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.

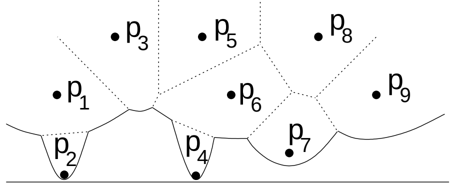
(b) Prove that there exists a k-splitter if and only if in the dual line arrangement, levels $\mathcal{L}_k$ and $\mathcal{L}_{n-k+1}$ can be separated by a line.

(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 4. 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 the figure below the string would be $p_1p_2p_1p_3p_6p_4p_6p_7p_9$.)



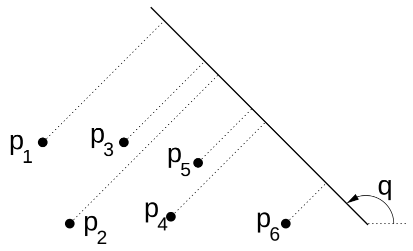
(a) Prove that for any i, j, the following alternating subsequence cannot appear anywhere within such a string:

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

(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 5. 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 6. 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 7. 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.)