CMSC 754: Fall 2021 Dave Mount

Sample Problems for the Midterm Exam

The midterm exam will be this **Thursday, Oct 28 in class**. It will be *closed-book* and *closed-notes*, but you may use *one sheet of notes* (front and back).

Unless otherwise stated, you may assume *general position*. If you are asked to present an $O(f(n))$ time algorithm, you may present a *randomized algorithm* whose expected running time is $O(f(n))$. For each algorithm you give, derive its running time and justify its correctness.

Disclaimer: The following sample problems have been collected from old homeworks and exams. Because the material and order of coverage varies each semester, these problems *do not* necessarily reflect the actual length, coverage, or difficulty of the midterm exam.

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

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

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

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

(c) A convex polygon P_1 is enclosed within another convex polygon P_2 (see Fig. 1(a)). Suppose you dualize the vertices of each of these polygons (using the dual transform given in class, where the point (a, b) is mapped to the dual line $y = ax - b$). What can be said (if anything) about the relationships between the resulting two sets of dual lines.

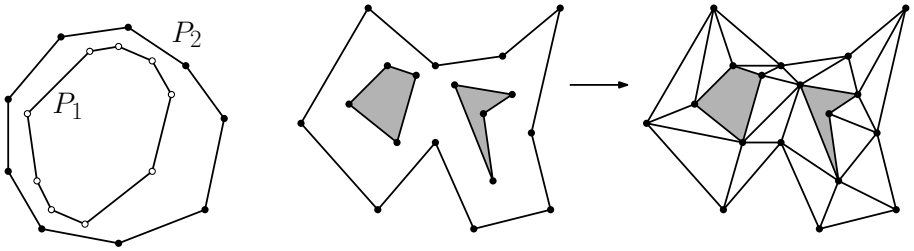


Figure 1: Problems 1(d) and 1(e).

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

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

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

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

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

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

(i) When the ith site is added to the Delaunay triangulation using the randomized incremental algorithm, what is the worst-case number of edges that can be incident on the newly added site? What can you say about the expected-case number of such edges (assuming that points are inserted in random order)?

(j) An arrangement of n lines in the plane has exactly n^2 edges. How many edges are there in an arrangement of n planes in 3-dimensional space? (Give an exact answer for full credit or an asymptotically tight answer for half credit.) Explain briefly.

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

(a) You are given a convex polygon P in the plane having n_P sides and an x-monotone polygonal chain Q having n_Q sides (see Fig. 2(a)). What is the maximum number of intersections that might occur between the edges of these two polygons?

(b) Same as (a), but P and Q are both polygonal chains that are monotone with respect to the x-axis (see Fig. 2(b)).

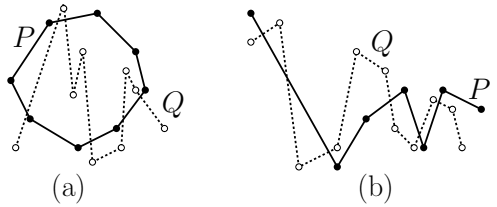


Figure 2: Maximum number of intersections.

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

Problem 3. A simple polygon P is *star-shaped* if there is a point q in the interior of P such that for each point p on the boundary of P, the open line segment $\overline{qp}$ lies entirely within

the interior of P (see Fig. 3). Suppose that P is given as a counterclockwise sequence of its vertices $\langle v_1, v_2, \ldots, v_n \rangle$. Show that it is possible to determine whether P is star-shaped in $O(n)$ time. (Note: You are *not* given the point q.) Prove the correctness of your algorithm.

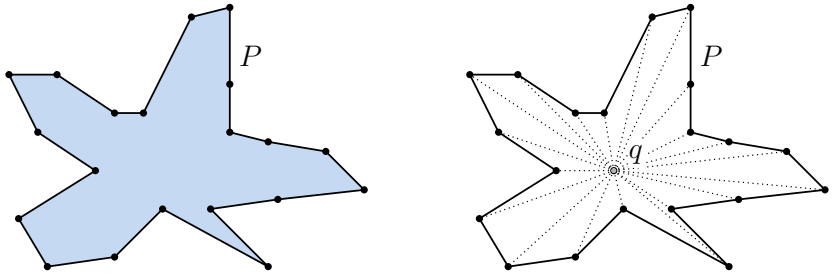


Figure 3: Determining whether a polygon is star-shaped.

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

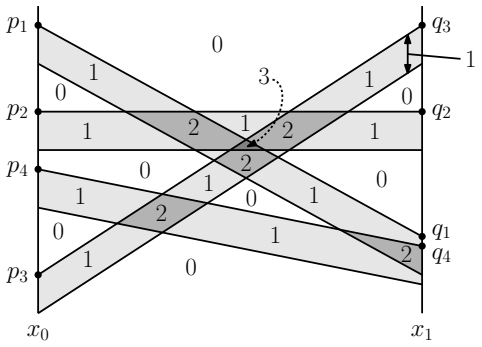


Figure 4: Maximum depth in a set of slabs.

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

Problem 5. Consider the following randomized incremental algorithm, which computes the smallest rectangle (with sides parallel to the axes) bounding a set of points in the plane. This rectangle is represented by its lower-left point, low, and the upper-right point, high.

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

(2) Let $\text{low}[x] = \text{high}[x] = p_1[x]$. Let $\text{low}[y] = \text{high}[y] = p_1[y]$.

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

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

Clearly this algorithm runs in $O(n)$ time. Prove that the total number of times that the "then" clauses of statements 3(a)–(d) (each indicated with a $(*)$) are executed is $O(\log n)$ on average. (We are averaging over all possible random permutations of the points.) To simplify your analysis you may assume that no two points have the same x- or y-coordinates.

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

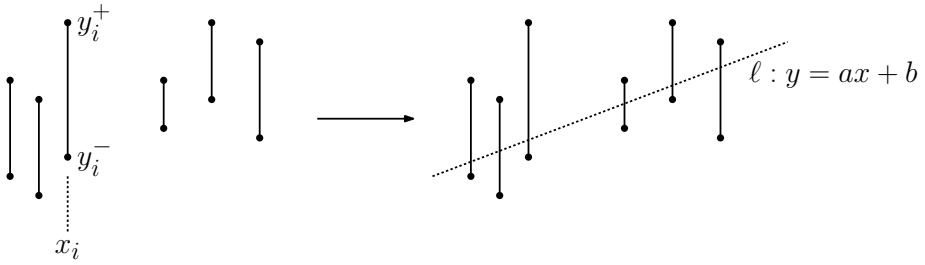


Figure 5: Existence of a transversal.

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

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

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

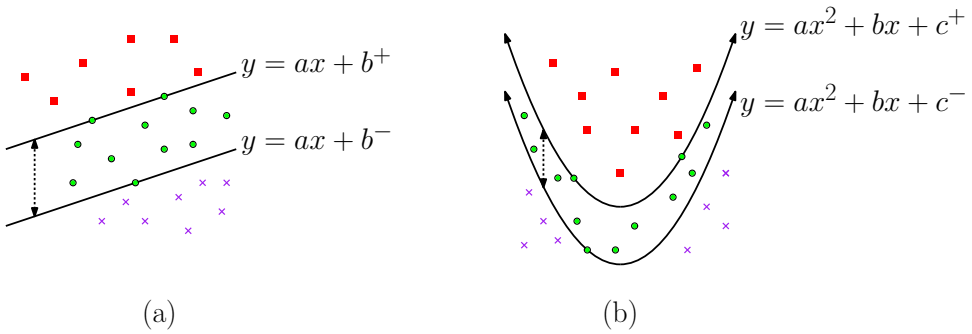


Figure 6: Linear and parabolic slabs.

Problem 8. You are given two sets of points in the plane, the red set R containing n_r points and the blue set B containing n_b points. The total number of points in both sets is $n = n_r + n_b$. Give an $O(n)$ time algorithm to determine whether the convex hull of the red set intersects the convex hull of the blue set. If one hull is nested within the other, then we consider them to intersect. (Hint: It may be easier to consider the question in its inverse form, do the convex hulls *not* intersect.)

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

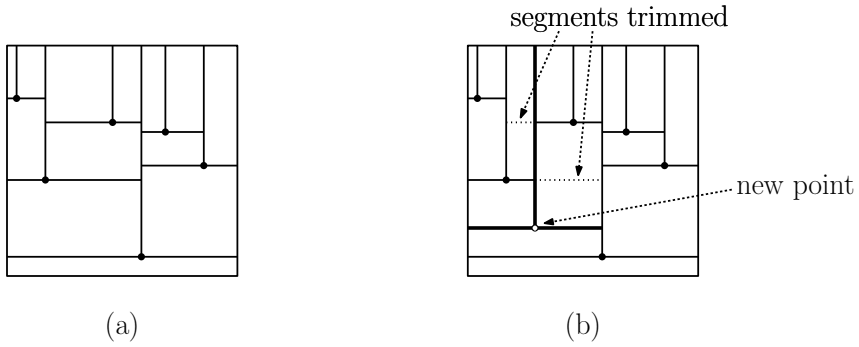


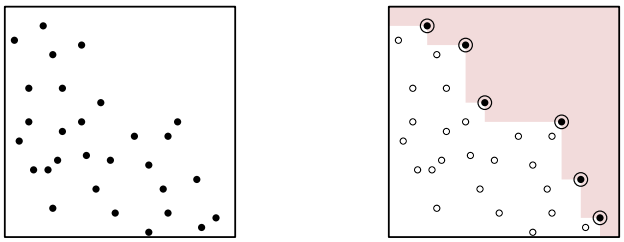
Figure 7: Building a subdivision.

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

(b) Describe an algorithm to add a new point to the subdivision and restore the proper subdivision structure. Note that the new point may have an arbitrary y-coordinate, but the subdivision must be updated as if the points had been inserted in increasing order of y-coordinate (see Fig. 7(b)).

(c) Prove that if the points are added in random order, then the expected number of structural changes to the subdivision with each insertion is $O(1)$.

Problem 10. Given two points $p_1 = (x_1, y_1)$ and $p_2 = (x_2, y_2)$ in the plane, we say that p_2 *dominates* p_1 if $x_1 \leq x_2$ and $y_1 \leq y_2$. Given a set of points $P = \{p_1, p_2, \ldots, p_n\}$, a point p_i is said to be *Pareto maximal* if it is not dominated by any other point of P (shown as black points in Fig. 8(b)).



(a) (b)

Figure 8: Paresto maxima.

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

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

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

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

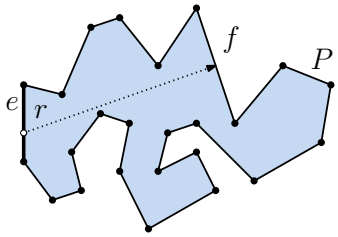


Figure 9: Ray-shooting queries.

Problem 12. You are given a set P of n points in $\mathbb{R}^2$. Present data structures for answering the following two queries. In each case, the data structure should use $O(n^2)$ space, it should answer queries in $O(\log n)$ time. (You do not need to explain how to build the data structure.)

(a) The input to the query is a nonvertical line ℓ. Such a line partitions P into two (possibly empty) subsets: $P^+(\ell)$ consists of the points lie on or above ℓ and $P^-(\ell)$ consists of the points of P that lie strictly below ℓ (see Fig. 10(a)). The answer is the maximum vertical distance h between two lines parallel to h that lie between $P^+(\ell)$ and $P^-(\ell)$ (see Fig. 10(b)).

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

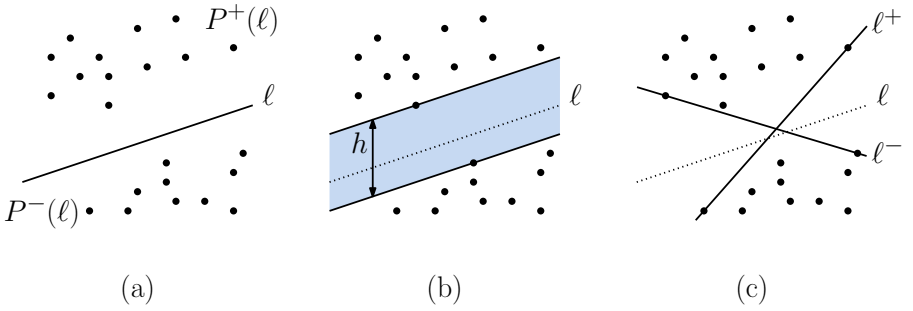


Figure 10: Separation queries.

(b) Again, the input to the query is a nonvertical line ℓ. The answer to the query consists of the two lines ℓ^- and ℓ^+ of minimum and maximum slope, respectively, that separate $P^+(\ell)$ from $P^-(\ell)$ (see Fig. 10(c)). You may assume that $P^+(\ell)$ from $P^-(\ell)$ are *not* separable by a vertical line (implying that these two slopes are finite).

Problem 13. You are given a set P of n points in the plane and a path π that visits each point exactly once. (This path may self-intersect. See Fig. 11.)

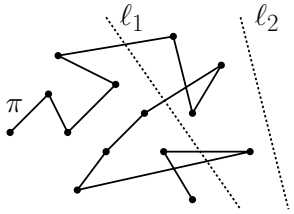


Figure 11: Path crossing queries.

Explain how to build a data structure from P and π of space $O(n)$ so that given any query line ℓ, it is possible to determine in $O(\log n)$ time whether ℓ intersects the path. (For example, in Fig. 11 the answer for ℓ_1 is "yes," and the answer for ℓ_2 is "no.") (Hint: Duality is involved, but the solution requires a bit of "lateral thinking.")

Problem 14. Consider the following two geometric graphs defined on a set P of points in the plane.

(a) *Box Graph:* Given two points $p, q \in P$, define $\text{box}(p, q)$ to be the square centered at the midpoint of $\overline{pq}$ having two sides parallel to the segment $\overline{pq}$ (see Fig. 12(a)). The edge (p, q) is in the box graph if and only if $\text{box}(p, q)$ contains no other point of P (see Fig. 12(b)). Show that the box graph is a subgraph of the Delaunay triangulation of P.

(b) *Diamond Graph:* Given two points $p, q \in P$, define diamond(p, q) to be the square having $\overline{pq}$ as a diagonal (see Fig. 12(c)). The edge (p, q) is in the diamond graph if and only if diamond(p, q) contains no other point of P (see Fig. 12(d)). Show that the diamond graph may not be a subgraph of the Delaunay triangulation of P. (Hint: Give an example that shows that the diamond graph is not even planar.)

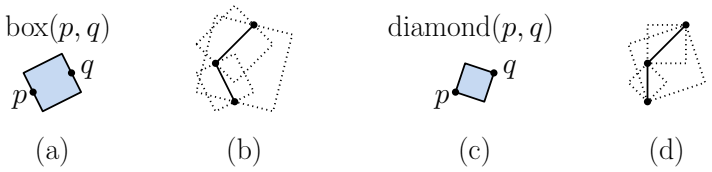


Figure 12: The box and diamond graphs.

Problem 15. You are given a set of n sites P in the plane. Each site of P is the center of a circular disk of radius 1. The points within each disk are said to be *safe*. We say that P is *safely connected* if, given any $p, q \in P$, it is possible to travel from p to q by a path that travels only in the safe region. (For example, the disks of Fig. 13(a) are connected, but the disks of Fig. 13(b) are not.)

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

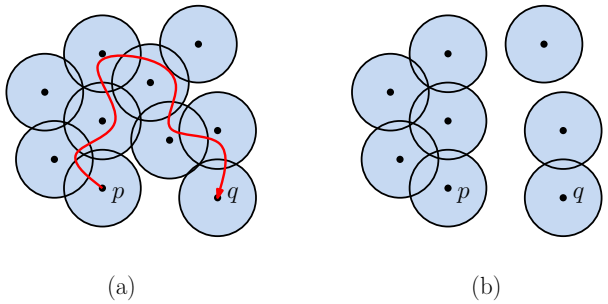


Figure 13: Safe connectivity.

Problem 16. In class we argued that the number of parabolic arcs along the beach line in Fortune's algorithm is at most $2n - 1$. The goal of this problem is to prove this result in a somewhat more general setting.

Consider the beach line at some stage of the computation, and let $\{p_1, \ldots, p_n\}$ denote the sites that have been processed up to this point in time. Label each arc of the beach line with its the associated site. Reading the labels from left to right defines a string. (In Fig. 14 below the string would be "$p_2p_1p_2p_5p_7p_9p_{10}$".)

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

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

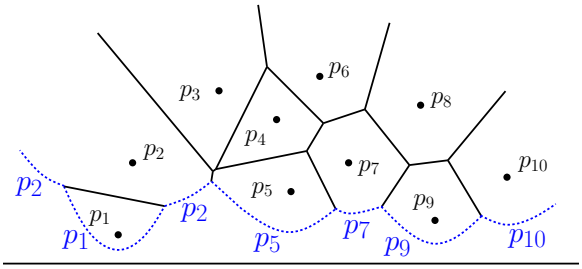


Figure 14: Beach-line complexity.

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

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

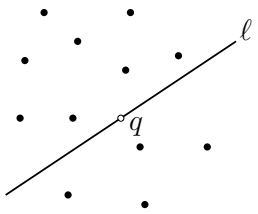


Figure 15: Point q is 4-deep within P.

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

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

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

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

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

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