CMSC 754: Spring 2020 Dave Mount

CMSC 754: Final Exam

Throughout, 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 $O(T(n))$. Total point value is 100 points. **Good luck!**

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

(a) (3 points) You are given four points a, b, c, d in $\mathbb{R}^2$. Using just orientation tests, show how to test whether these points are the vertices of a convex quadrilateral listed in counterclockwise order.

(b) (3 points) In the plane-sweep algorithm for line-segment intersection, we were meticulous in deleting intersection events from the priority queue whenever the two segments generating this event were no longer consecutive on the sweep line. Why did we do this?

(c) (3 points) 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.

(**Hint:** Recall the definition from Lecture 5. No proof is needed. It is okay if your answer is off by a constant additive term. For full credit, the multiplicative factor on n should be tight.)

(d) (3 points) We claimed that the worst-case depth of the point-location data structure based on trapezoidal maps is $\Omega(n)$. Draw an example of a set of line segments and an insertion order for these segments such that the depth of the point-location data structure is $\Omega(n)$. (It should be clear how to generalize your example to work with arbitrarily large values of n.)

(e) (6 points)

(i) Define the *zone* of an arrangement of lines in the plane.

(ii) State the *Zone Theorem*.

(iii) What was the importance of the zone theorem in our algorithm for building line arrangements in the plane?

(f) (12 points) Consider the simplicial complex shown in Fig. 1(a), consisting of four 0-simplices $(0, 1, 2, 3)$, five 1-simplices $(01, 03, 12, 13, 23)$, and two 2-simplices $(013, 123)$. Consider the 1-chains $c_1 = \{01, 12, 23\}$ and $c_2 = \{12, 13, 23\}$ (see Fig. 1(b)).

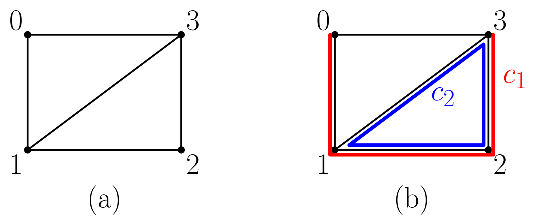

Figure 1: Chains in a simplicial complex. Recall the definitions from Lecture 18.

(i) Express c_1 and c_2 as 0-1 vectors over the basis $\langle 01, 03, 12, 13, 23\rangle$.

(ii) Express $c_1 + c_2$ as a 0-1 vector over this same basis.

(iii) What is the chain c_3 such that $c_1 + c_3 = c_2$? (Express it as a 0-1 vector over this same basis.)

(iv) What are the boundaries of c_1 and c_2? (**Hint:** Each is a 0-chain, which can be expressed as a 0-1 vector over the basis $\langle 0, 1, 2, 3 \rangle$.

Problem 2. (10 points) 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) (5 points) 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. 2(a)). If no such slab exists, you should detect and report this.

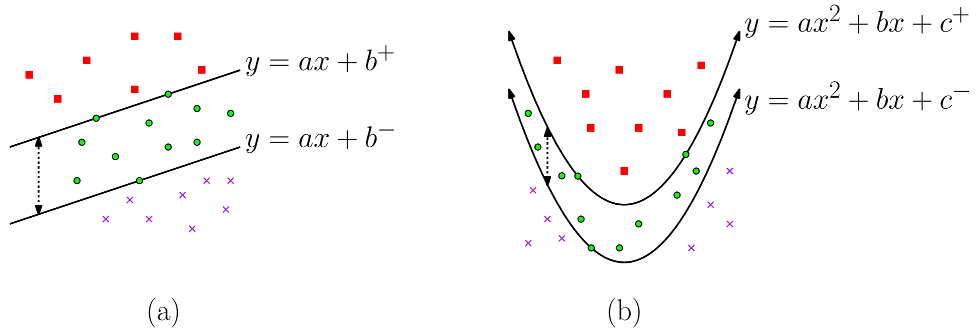

Figure 2: Linear and parabolic slabs.

(b) (5 points) 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. 2(b)). If no such parabolic slab exists, you should detect and report this.

Problem 3. (25 points) You are given two sets of points R and B in $\mathbb{R}^2$ (called red and blue, respectively). Let n denote the total number of points in both sets. Let us assume that the point sets are disjoint, that is, $R \cap B = \emptyset$. Given an arbitrary point $q \in \mathbb{R}^2$ (not in R or B), we *color* it (red or blue) depending on whether its closest point in $R \cup B$ is red or blue. (In Fig. 3(a), q is colored red, since its nearest neighbor is red). If q is equidistant to its closest points in R and B, it may be assigned either color.

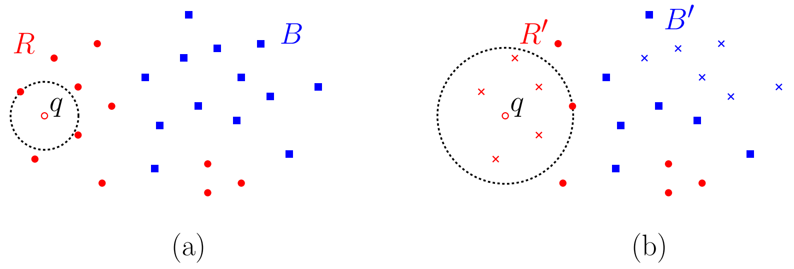

Figure 3: Coloring queries via nearest neighbors and compatible subsets.

(a) (5 points) Present a data structure with $O(n)$ storage which is given R and B as inputs, and answers the following *color queries*: given any $q \in \mathbb{R}^2$, return q's color. It should be possible to build your data structure in $O(n \log n)$ time, and it should answer queries in $O(\log n)$ time. (**Hint:** Don't do this from scratch. Combine structures we have seen in this semester.)

(b) (5 points) In some applications, n may be extremely large, and so it is desirable to reduce the sizes of R and B as much as possible. We say that two subsets $R' \subseteq R$ and $B' \subseteq B$ are *compatible* with R and B if every point $q \in \mathbb{R}^2$ is assigned the same color with respect to R' and B' that it would have been assigned with respect to the original sets R and B. (We have removed all the points marked with "$\times$" in Fig. 3(b). Observe that point q is still correctly colored by its nearest neighbor in the reduced point set.)

Present an efficient algorithm that, given R and B as input, produces the compatible subsets $R' \subseteq R$ and $B' \subseteq B$ having the smallest possible number of points. Just give the algorithm. Correctness and running time will be covered below. (**Hint:** Voronoi diagrams and/or Delaunay triangulations may be helpful.)

(c) (5 points) Prove that your algorithm produces a compatible subset by showing that for every query point $q \in \mathbb{R}^2$, the color of its nearest neighbor in $R' \cup B'$ is the same as its nearest neighbor in $R \cup B$.

(d) (5 points) Assuming that the points of $R \cup B$ are in general position, prove that your algorithm produces a minimum-sized subset by showing that for any smaller subset, there exists a point $q \in \mathbb{R}^2$ that will be incorrectly colored based on its nearest neighbor. (The general-position assumption is critical!)

(e) (5 points) Derive the running time of your algorithm. (**Hint:** $O(n \log n)$ time is possible. Partial credit will be given for a slower algorithm.)

Problem 4. (20 points) 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.

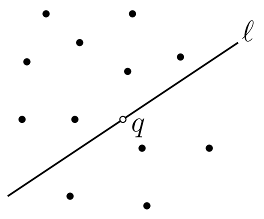

Figure 4: Point q is 4-deep within P.

For example, the point q in Fig. 4 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) (5 points) 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) (5 points) 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) (10 points) 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.

Problem 5. (15 points) A set P of n points in $\mathbb{R}^d$ determines a set of $\binom{n}{2}$ different distances. Define $\Delta(P)$ to be this set of distances $\{\|p_i - p_j\| : 1 \leq i < j \leq n\}$. Given an integer k, where $1 \leq k \leq \binom{n}{2}$, we are interested in computing the kth smallest distance from this set. Normally, this would take $O(n^2)$ time, so let's consider a fast approximation algorithm.

Let $\delta(P,k)$ denote the exact kth smallest distance in $\Delta(P)$. Given $\varepsilon > 0$, a distance value x is an ε-approximation to $\delta(P,k)$ if

$$\frac{\delta(P,k)}{1+\varepsilon} \leq x \leq (1+\varepsilon)\delta(P,k).$$

Present an efficient algorithm to compute such a value x. Justify your algorithm's correctness and derive its running time. (**Hint:** Use well-separated pair decompositions. You may assume that, when the quadtree is computed, each node u of the quadtree is associated with an integer $\mathrm{wt}(u)$, which indicates the number of points of P lying within u's subtree.)

You may assume that d and ε are constants (independent of n). I know of an algorithm that runs in time $O(n \log n + n/\varepsilon^d)$ time, but I will accept for full credit an algorithm that runs in time $O((n \log n)/\varepsilon^d)$.)