CMSC 754: Spring 2026 Dave Mount

Solutions to the Final Exam

Solution 1:

(a) Chan's algorithm runs in $O(n \log h)$ time.

(b) The plane-sweep algorithm runs in $O((n+m)\log n)$ time and uses $O(n)$ space.

(c) Every pair of lines intersect, so the number of vertices is $\binom{n}{2}$, or equivalently $n(n-1)/2$.

(d) The size of an s-WSPD for n points in $\mathbb{R}^d$ is $O(s^d n)$.

(e) Axis-aligned squares are pseudodisks. The total number of vertices is $4n$. By the pseudodisk lemma, the number of vertices on the boundary of their union is at most $2(4n) = 8n$.

Solution 2: We will show how to answer water-drop queries efficiently for a collection of segments S.

(a) To build the data structure, we first enclose S within a bounding rectangle that has the x-axis as its bottom side. We then construct the trapezoidal map of the segments and the associated point-location data structure. We label each trapezoid as follows. First, if a trapezoid touches the x-axis, we label this a *ground trapezoid* (shaded in Fig. 1(a)).

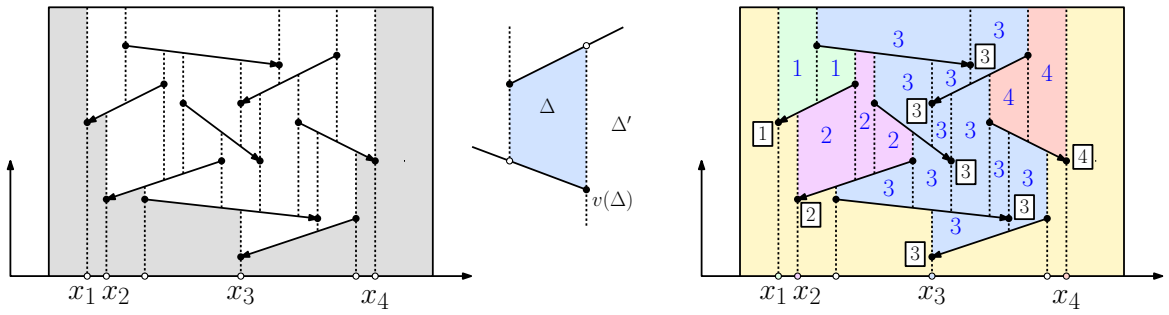

Figure 1: Solution to water-drop queries.

For these, the answer to the query is just q's x-coordinate. For all other trapezoids, the answer to the query is the same for all points that lie within the trapezoid, and is computed as follows. First, sort the trapezoids bottom-up by the lowest y-coordinate in the trapezoid. Visit the trapezoids in this order. (This could also be done by a topological ordering of the map's dual graph, sorted based on the lowest vertex.) For any trapezoid Δ, if Δ is a ground trapezoid, then the drop falls directly onto the x-axis, and the answer to the query is the x-coordinate of the query point. Otherwise, the answer is determined by the lowest vertex of Δ, denoted $v(\Delta)$ (see Fig. 1(b)). This vertex is on the boundary of an adjacent trapezoid, Δ', that shares a vertical wall with Δ. Note that Δ' has an even lower vertex, and hence has been previously processed. Query Δ' for the answer to the query at $v(\Delta)$, and label Δ with this answer (see Fig. 1(c)).

(b) To answer the query, we determine the trapezoid that contains the query point q. If it is a ground trapezoid, we return q_x. Otherwise, we return the label stored in this trapezoid. The query time is dominated by the point-location time, which is $O(\log n)$.

Technically, we placed all the segments in a bounding box. We could either define this box to be large enough to include all possible queries. Alternatively, we should add additional cases for when q lies outside the box. If q lies to the left or right of the box, then the answer is just q_x. Otherwise, it lies above the box. Let q' denote the point on the box just below q. We then apply the query algorithm to q' and return the result as the answer.

Solution 3: We will solve this by reducing it to linear programming (LP). Let's assume that the left edge of the strip is on the y-axis. We will compute the slope d and the y-intercept e of the line of the initial shot.

There are two approaches to address the fact that the beam's trajectory has four segments. The first is to determine the equations of the four separate lines, and apply them to the points. If we model the initial line by $L_0 : y = dx + e$, the other three lines are L_1 (reflecting L_0 about the lower wall), L_2 (reflecting L_1 about the upper wall), and L_3 (reflecting L_2 about the lower wall). Formally:

$$L_1 : y = -dx - e, \quad L_2 : y = dx + (2w + e), \quad \text{and} \quad L_3 : y = -dx - (2w + e)$$

(see Fig. 2(a)). From these, we obtain $2n$ constraints, stating that the points must lie above/below an appropriate subset of lines. In particular, points of A must be above both L_0 and L_1, points of B must be below both L_1 and L_2, and points of C must be above both L_2 and L_3. That is, each $p \in A \cup C$ must lie on or above the line, and each $p \in B$ must lie below the line. For example, for each $(a_x, a_y) \in A$, we have

$$a_y \ \geq \ da_x + e \quad \text{and} \quad a_y \ \geq \ -da_x - e,$$

and so on for the points of B and C.

In addition, we check that the beam passes through the left edge of the strip by adding the constraint $0 \leq e \leq w$, and that the beam hits the lower wall first by adding the constraint that $d \leq 0$. The objective function is arbitrary. For example, we could just minimize the slope by maximizing $-d$. The LP cannot be unbounded since both d and e are both bounded. If the LP is infeasible, there is no beam that works. Otherwise (it returns a feasible solution), we can select the starting point (s_x, s_y) to be any point on the line that lies on or to the left of the y-axis, e.g., $s = (0, e)$, and the angle is determined by the slope, that is $\theta = \arctan e$.

The alternative is to use a single line equation, $L : y = dx + e$, but transform the points. We need to make two copies of each point set to account for the two sets of constraints. In particular, letting $u = (0, w)$ denote the vertical vector of length w, and given any set S, letting $-S$ denote the reflection of S across the x-axis (by negating the y-coordinates), we have

$$\begin{aligned} A_1 &= A & \qquad A_2 &= -A \\ B_1 &= -B & \qquad B_2 &= B - 2u \\ C_1 &= C - 2u & \qquad C_2 &= -C - 2u. \end{aligned}$$

Then we have the $2n$ constraints that A_1, B_1 and C_1 lie on or above L and A_2, B_2 and C_2 lie on or below L (see Fig. 2(b)). Otherwise, the solution is the same as in the first case.

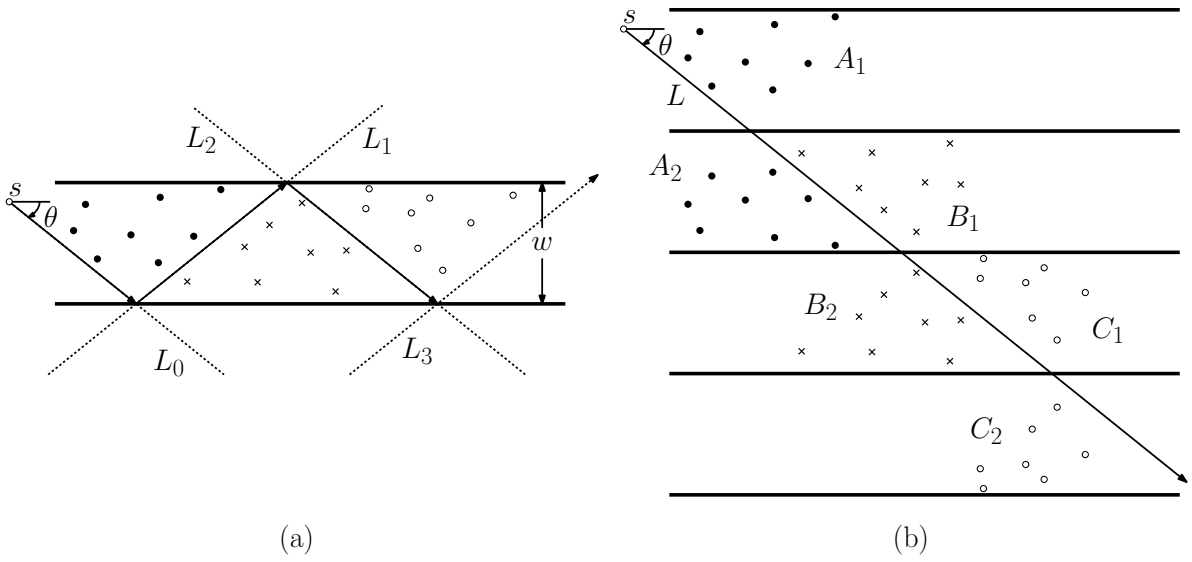

Figure 2: Laser beam.

Solution 4: Given the point set P, let $\langle \ell_1, \ldots, \ell_{n-1} \rangle$ denote the lengths of the edges of the associated Euclidean minimum spanning tree of P, EMST(P), sorted from smallest to largest. We will show below that the ith critical radius r_i equals half the length of the i-th shortest edge in the Euclidean minimum spanning tree (EMST) of P, that is, $r_i = \ell_i/2$. Assuming this for now, it follows that we can compute the sequence $\langle r_1, \ldots, r_{n-1} \rangle$ in $O(n \log n)$ time as follows. We proved in class that EMST(P) is a subgraph of DT(P). We first compute the Delaunay triangulation of P in $O(n \log n)$ time, and then compute the MST of the Delaunay triangulation by Kruskal's algorithm. A nice feature of Kruskal's algorithm is that it produces the edges of the MST in increasing order of length. The overall running time is $O(n \log n)$.

To establish correctness, we first show that $r_i \leq \ell_i/2$. Let T_i denote the forest consisting of the i smallest edges of EMST(P). Clearly, T_i has $n - i$ connected components. Since every edge of T_i has length at most ℓ_i, the balls of radius r_i placed at the endpoints of each edge cover every edge of T_i. It follows that the union of these balls covers each component of T_i, which implies that there are at least i connected components in the union of these balls. This implies that $r_i \leq \ell_i/2$ (see Fig. 3(a)).

Conversely, to show that $r_i \geq \ell_i/2$, suppose we run Kruskal's MST algorithm, but we restrict ourselves to edges of strictly shorter length than ℓ_i. Because Kruskal's algorithm adds edges in increasing order of length, and each new edge reduces the number of connected components by exactly one, it follows that the algorithm will have $n - i$ connected components. By considering the subtrees of the partial spanning tree, it follows that it is possible to partition the point set into disjoint subsets $P_1, \ldots, P_{n-i} \subset P$, such that for any points p_i and p_j in different subsets, $\|p_i - p_j\| > \ell_i$ (see Fig. 3(b)). This implies that if we place disks of radius $\ell_i/2$ at all the points of P, no ball of one group will overlap any ball of another group, and hence the balls are not connected.

Solution 5: Even though we asked for an $O(n^2)$ time solution, this problem can be solved in linear time through linear programming. We will present two solutions: one based on LP, running

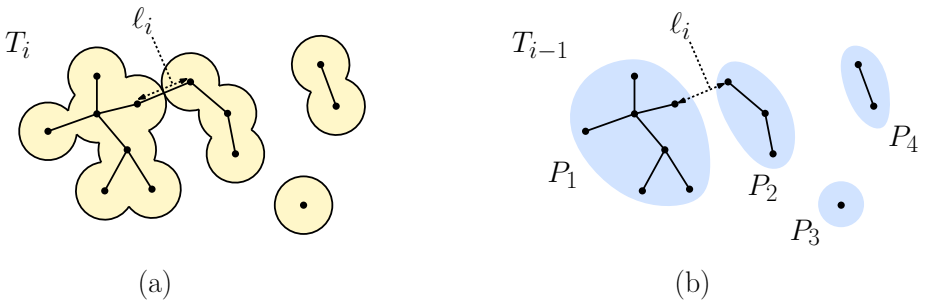

Figure 3: Computing the critical radius values.

in $O(n)$ time, and one based on a topological plane sweep over a line arrangement, running in $O(n^2)$ time and $O(n)$ space.

For the LP-based solution, we seek two parallel lines $\ell_+ : y = cx + d_+$ and $\ell_- : y = cx + d_-$ such that all the points of A lie on or above ℓ_- and all the points of B lie on or below ℓ_+, and the vertical width $d_+ - d_-$ is minimized (see Fig. 4(a)). We model this as an LP in $\mathbb{R}^3$ with the variables (a, d_-, d_+) with the constraints,

$$\begin{aligned} &\text{minimize } d_+ - d_- \\ &\quad \text{subject to:} \quad a_y \geq ca_x + d_- \qquad \text{for all } (a_x, a_y) \in A \\ &\qquad\qquad\quad\; b_y \leq cb_x + d_+ \qquad \text{for all } (b_x, b_y) \in B. \end{aligned}$$

The problem description states that we can assume the two sets are not linearly separable, implying that any solution must also satisfy $d_+ > d_-$. This implies that the LP is not unbounded. It is also evident that there exists a feasible solution for any slope by setting d_- sufficiently small and d_+ sufficiently large. Therefore, a feasible optimal solution always exists. By the algorithm given in class, this leads to an $O(3!n) = O(n)$ time randomized solution.

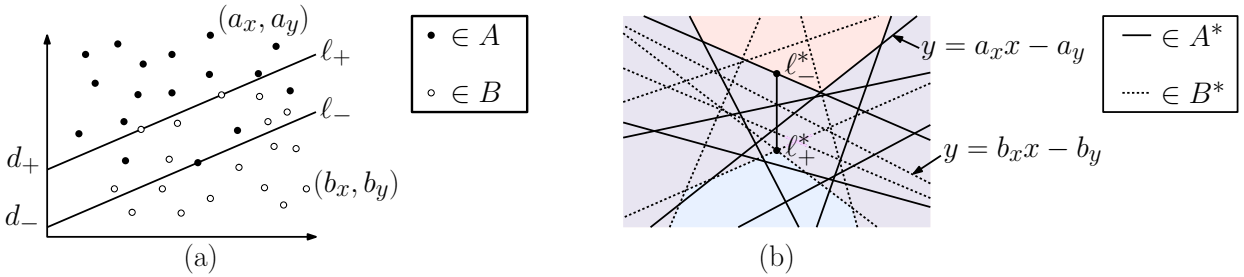

Figure 4: Semi-separating lines.

For the arrangement-based solution, we dualize the points of A and B as lines. For each $(a_x, a_y) \in A$, let $a^* : y = a_x x - a_y$ and for each $(b_x, b_y) \in B$, let ${}^*b : y = b_x x - b_y$. Let A^* and B^* be the associated sets of lines. Let $\ell_+ : y = cx + d_+$ and $\ell_- : y = cx + d_-$ denote the parallel lines we seek (see Fig. 4(b)). By the order-reversing properties of the duality transformation, our objective is to compute dual points $\ell_+^* = (c, -d_+)$ and $\ell_-^* = (c, -d_-)$ such that all the dual lines of A^* pass

below ℓ_-^* and all the dual lines of B^* pass above ℓ_+^*, that is

$$\begin{aligned} -d_- &\geq ca_x - a_y \qquad \text{for all } (a_x, a_y) \in A \\ -d_+ &\leq cb_x - b_y \qquad \text{for all } (b_x, b_y) \in B, \end{aligned}$$

so as to minimize the vertical distance between these dual points, that is, $-d_- - (-d_+) = d_+ - d_-$. (Not surprisingly, these are equivalent to the conditions of the LP-based solutions, just expressed in the dual setting.) An important observation is that in order to obtain a locally optimal solution, the points ℓ_+^* and ℓ_-^* must both lie on lines of the arrangement, and at least one must coincide with a vertex of the arrangement (since otherwise, the distance could be reduced by shifting the vertical segment to the left or right).

To compute ℓ_+^* and ℓ_-^* we compute the dual line sets A^* and B^*. We apply topological plane sweep. Throughout the sweep, we keep track of the extreme lines, that is, the highest line of A^* and the lowest line of B^* that intersects the sweep line. (Equivalently, we track the highest level of A^* and the lowest level of B^*.) At each intersection event between two lines of A^* or two lines of B^* if either is an extreme line, we update this information by making the other line extreme. Whenever we encounter an intersection point, that is, a vertex in the arrangement, we compute the vertical distance between the two current extreme lines at this x-coordinate. When the sweep completes, we report the smallest vertical distance seen throughout the sweep.

By standard topological plane sweep, this takes $O(n^2)$ time and $O(n)$ space.

Solution 6:

(a) We will prove that given any query point $q \in \mathbb{R}^d$, the distance between q and its farthest point in P is at least $\text{diam}(P)/2$. To see this, let p be this farthest point of P from q. Also, let p_1 and p_2 denote the two points that define the diameter of P. We have $\|p_1 - q\|$ and $\|p_2 - q\|$ are both less than or equal to $\|p - q\|$. Therefore, by the triangle inequality, we have

$$\text{diam}(P) \;=\; \|p_1 - p_2\| \;\leq\; \|p_1 - q\| + \|q - p_2\| \;\leq\; \|p - q\| + \|p - q\| \;=\; 2\|p - q\|,$$

and hence we have $\|p - q\| \geq \text{diam}(P)/2$, as desired.

(b) The simplest approach is to place points of P in a grid of diameter $\varepsilon \cdot \text{diam}(P)/2$, and keep one representative point per grid cell. Because all the points of P can be placed within a hypercube of side length $\text{diam}(P)$, it follows that the number of points of this set is $O(1/\varepsilon^d)$. Since distances are preserved up to an absolute error of $\varepsilon \cdot \text{diam}(P)/2$ and by (a) the distance to the farthest neighbor is at least $\text{diam}(P)/2$, it follows that relative error is at most ε. This establishes the lower bound on the error, and the fact that $R \subseteq P$ implies that the upper bound holds trivially.

To get a better bound on the size, we will show that any ε-coreset for directional-width queries is an (2ε)-coreset for farthest distance queries. From this, it follows that applying the ε'-kernel construction from class, but with ε' set to $\varepsilon/2$, yields the desired result. (Note that dividing ε by 2 merely alters the constant factors in the size of the coreset.)

Unfortunately, proving this is more difficult than the simple solution. The trick is to consider the directional width in the direction of q's farthest point and exploit the fact that orthogonal projection can never decrease distances. To make this formal, consider any query point $q \in \mathbb{R}^d$,

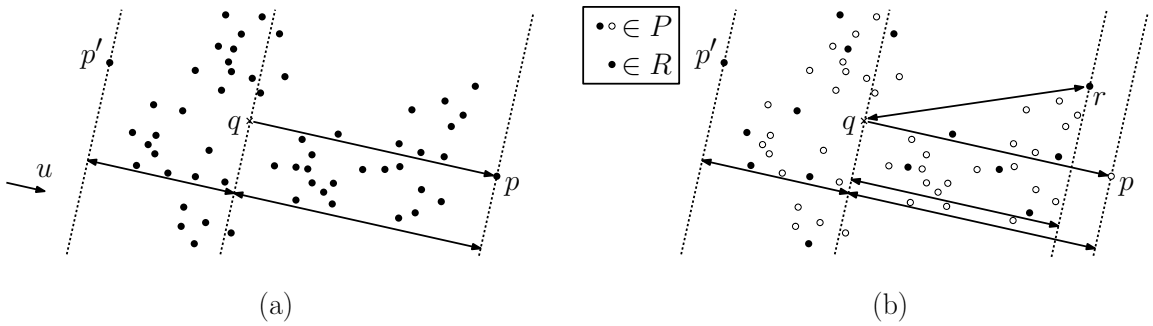

Figure 5: Coreset for farthest distance queries.

let p be its farthest point in P, and let u denote the unit vector in the direction of $p - q$ (see Fig. 5(a)). Observe that $\|p - q\| = (p - q) \cdot u$.

We first assert that $(p - q) \cdot u \geq \text{width}_P(u)/2$. To see why, let p' be the extreme point in direction u on the other side of q, that is, it is the point that maximizes $(p' - q) \cdot (-u)$. By basic linear algebra, we have

$$\text{width}_P(u) \;=\; (p - p') \cdot u \;=\; ((p - q) \cdot u) + ((q - p') \cdot u) \;=\; ((p - q) \cdot u) + ((p' - q) \cdot (-u)).$$

If it were the case that $(p - q) \cdot u < \text{width}_P(u)/2$, then we would have $((p' - q) \cdot (-u)) > \text{width}_P(u)/2$. Since orthogonal projection cannot increase lengths, we have

$$\|p' - q\| \;>\; \text{width}_P(u)/2 \;>\; (p - q) \cdot u \;=\; \|p - q\|,$$

which contradicts the hypothesis that p is the farthest point from q.

Let R be any ε-coreset for directional-width queries, and let $r \in R$ be the point that maximizes $(r - q) \cdot u$ (see Fig. 5(a)). We assert that $(r - q) \cdot u \geq (1 - 2\varepsilon)(p - q) \cdot u$. To see why, suppose to the contrary that $(r - q) \cdot u < (1 - 2\varepsilon)(p - q) \cdot u$. Since p' is extreme for P on the opposite side of q and $R \subseteq P$, the extreme point for R cannot be more extreme than p'. Thus, we have

$$\begin{aligned}
\text{width}_R(u) \;&\leq\; (r - p') \cdot u \;=\; ((r - q) \cdot u) + ((p' - q) \cdot (-u)) \\
&<\; (1 - 2\varepsilon)(p - q) \cdot u + ((p' - q) \cdot (-u)) \\
&=\; (p - q) \cdot u + ((p' - q) \cdot (-u)) - 2\varepsilon(p - q) \cdot u \\
&=\; \text{width}_P(u) - 2\varepsilon(p - q) \cdot u \;\leq\; \text{width}_P(u) - 2\varepsilon\text{width}_P(u)/2 \\
&=\; (1 - \varepsilon)\text{width}_P(u),
\end{aligned}$$

which contradicts the hypothesis that R is an ε-coreset for directional width.

Since orthogonal projection cannot increase lengths, we have $\|r - q\| \geq (r - q) \cdot u$. Since $\|p - q\| = (p - q) \cdot u$, we conclude that

$$\|r - q\| \;\geq\; (r - q) \cdot u \;\geq\; (1 - 2\varepsilon)(p - q) \cdot u \;=\; (1 - 2\varepsilon)\|p - q\|,$$

which implies that R is a (2ε)-coreset for farthest-point queries.