CMSC 754: Spring 2026 Dave Mount

Homework 3

Due by the start of class on Tuesday, Mar 3. (Submissions will be through Gradescope.) Late homeworks are not accepted (unless an extension has been prearranged) so please turn in whatever you have completed by the due date. Unless otherwise specified, you may assume that all inputs are given in *general position*.

Problem 1. For each of the following applications, present an reduction to linear programming (LP) in a space of constant dimension. For full credit, the dimension should be as low as possible. In each case, explain the following:

- What are the variables in your LP formulation?
- What is the objective function?
- What are the constraints?
- Explain which results are possible from your LP formulation (feasible, infeasible, and unbounded), and explain how the LP result is used to answer the application.
- What is the (expected case) running time of the LP? (Express your answer as a function of the number of constraints, n, and the dimension of the LP formulation, d.)

You do not need to explain how to solve the LP, just give the reduction. (Also, see the notes below on "Guidance for Writing LP Reductions".)

(a) You are given a convex polygon P, represented by a cyclic sequence of its vertices $\langle p_1, \ldots, p_n \rangle$. Compute the largest axis-aligned square contained within P (see Fig. 1(a)). **Hint:** Here is a suggested set of variables for your LP: (c_x, c_y, r), where $c = (c_x, c_y)$ is the center of the square, and r is half the side length of the square.

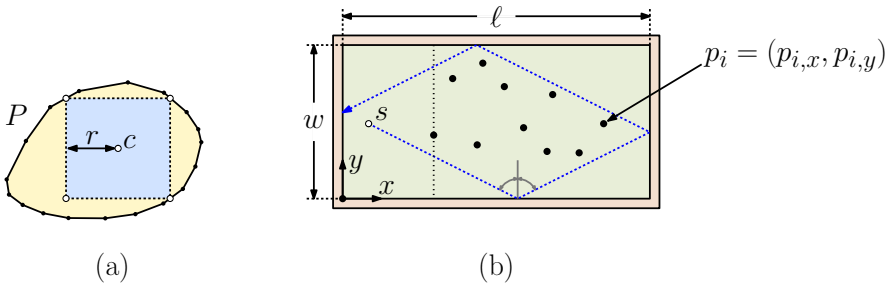

Figure 1: LP reductions.

(b) In your new career as a billiards player, you want to impress your friends with your accurate shot making. The billiards table is ℓ units long and w units wide. Let us assume the origin of the coordinate frame is in the lower-left, with the x-axis directed along the length of the table and the y-axis along the width. The cue ball (which you will shoot)

is located coordinates (x, y). There is a set of n balls on the table $P = \{p_1, \ldots, p_n\}$, where $p_i = (p_{i,x}, p_{i,y})$.

Your shot starts at some point $s = (s_x, s_y)$, then hits the bottom side, then the right side, then the top side, and finally the left side. The shot should travel around all the points of P without hitting any of them (see Fig. 1(b)). That is, each point of P should lie to the left of each directed segment on the shot's path. You are asked to determine whether such a shot is possible (simply, yes or no).

Does there exist a starting point $s = (s_x, s_y)$ on the billiards table and a shot direction, such that if the ball is shot from point s in the given direction, it will hit the bottom ball, then the right wall, then the top wall, and finally return to the left wall, such that shot encircles the points in the sense that each point lies to the left of the directed segments on which the ball travels.

To simplify your life, you should assume that the billiard balls are points (that is, they are balls of radius zero). When the shot hits a wall, it rebounds with the angle of reflection equal to the angle of incidence. Finally, you may assume that the shot starts to the left of all the balls in the sense that $s_x \leq p_{i,x}$ for all i.

Note: In LP, constraints are required to be *closed* halfspaces. This effectively implies that we need to allow the shot to pass through the balls on the table.

(c) Assuming the answer to (b) is "yes", determine how to direct the shot so that its final hit on the left wall is as high as possible point, while still satisfying the conditions of (b).

(d) Assuming the answer to (b) is "yes", determine how to direct the shot that it hits the left wall at a point that is closest to the midpoint of the left wall (either above or below). (**Hint:** If the shot hits the back wall at some location y, the distance to the midpoint is the absolute value $|y - w/2|$. While $y - w/2$ is a linear function of y, the absolute value of this quantity is *not* a linear function.)

Problem 2. The purpose of this problem is to get practice with designing and analyzing randomized incremental algorithms. You are to devise a randomized algorithm for computing the upper hull of a set of points P in the plane. The approach is as follows:

(1) Deterministically, compute the leftmost and rightmost points, denoted v_1 and v_n, respectively. The initial upper hull is the segment $\overline{v_1, v_n}$.

(2) Randomly permute the remaining points, which we denote as $\langle p_2, p_3, \ldots, p_{n-1} \rangle$.

(3) For i running from 2 to $n - 1$, add point p_i to the hull as follows (see Fig. 2(a)):

  (a) Determine the edge $\overline{v_j v_{j+1}}$ of the hull lying above or below p_i (see Fig. 2(b)).

  (b) If p_i lies below this edge, then ignore it.

  (c) Otherwise, walk outwards to the left and right of $\overline{v_j v_{j+1}}$ to compute points of tangency with respect to p_i, and delete any vertices that lie beneath these tangent lines. Add p_i to the hull.

How do we determine the edge lying immediately above or below p_i? We maintain a collection of *buckets*, one per edge of the hull, where each contains all the points that lie vertically above or below this edge. Whenever a new point is added to the hull, the points lying within the

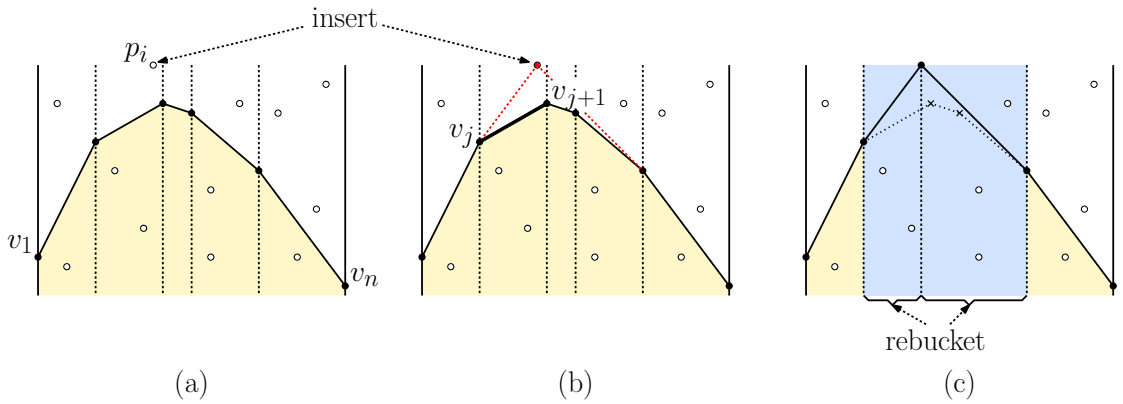

Figure 2: Incremental construction of the upper hull.

bucket of a deleted edge need to be "rebucketed". We assume that the points within each bucket are sorted by their x-coordinates. We merge the points from the deleted buckets (in time proportional to the number of points) and then walk through this sorted list and assign these points to their new buckets.

(a) Show that (ignoring the time for rebucketing) the above algorithm runs in $O(n)$ time.

(b) Show that, assuming that points are inserted in random order, the expected number of times any point is rebucketed is $O(\log n)$. In particular, show that for any point p, the probability that p is rebucketed during the ith insertion is at most c/i, where c is a constant. What is the value of c?

It is interesting to note that, unlike the other convex hull algorithms we have seen so far, this one generalizes to arbitrary dimensions.

Problem 3. (Optional–Ungraded) In this question, we will explore an alternate way in which to define point-line duality in the plane. (This generalizes to any dimension, but the plane is the simplest case to work with.) Any line ℓ in the x, y-plane that does not pass through the origin can be expressed by the equation $\ell : ax + by = 1$. Let us define the *polar dual* of ℓ, denoted ℓ^* to be the point (a, b). Similarly, given a point $p = (a, b)$, define its dual to be the line $p^* : ax + by = 1$. Clearly, this mapping is self-inverse, since $p^{**} = p$ and $\ell^{**} = \ell$.

(a) Given a line $\ell : ax + by = 1$, define its *(open) inner halfplane*, denoted $\ell_<$, to be set of points that lie on the same side of ℓ as the origin, that is $(x, y) \in \ell_<$ if $ax + by < 1$ (see Fig. 3(a)). Define $\ell_>$ analogously for points on the opposite side of ℓ, and define $\ell_\leq$ and $\ell_\geq$ analogously for the closed halfplanes. We say that point p is *inside* (resp., *outside*) ℓ if $p \in \ell_<$ (resp. $\ell_>$).

Prove that polar duality is "order reversing" by showing that point p lies inside/on/outside of ℓ if and only if ℓ^* lies inside/on/outside of p^*, respectively (see Fig. 3(b)).

(b) Prove that polar duality is "incidence preserving" by showing that ℓ_1 and ℓ_2 intersect at point p if and only if the dual line p^* passes through dual points ℓ_1^* and ℓ_2^*.

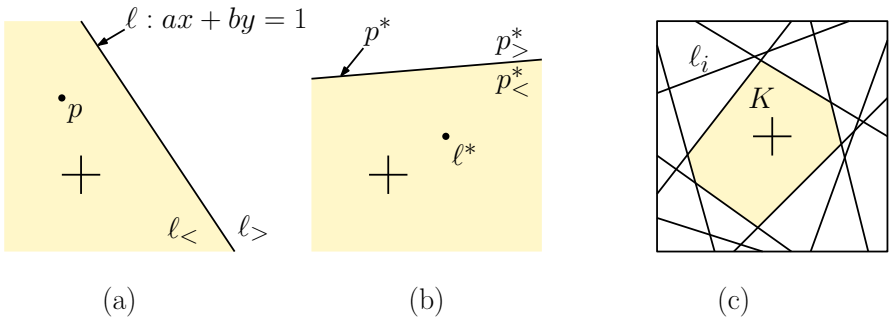

Figure 3: Polar dual.

(c) Let $L = \{\ell_1, \ldots, \ell_n\}$ be a set of lines in $\mathbb{R}^2$ and let K be the intersection of the associated (closed) inner halfplanes, that is, $K = \bigcap_{i=1}^{n} \ell_{\leq}$ (see Fig. 3(c). Let's assume that K is not empty (which implies that it contains the origin).

Describe the relationship between K and the convex hull of the polar dual set of points $L^* = \{\ell_1^*, \ldots, \ell_n^*\}$? In particular, explain how the CCW order of edges around the boundary of K relates to the order of points around $\text{conv}(L^*)$. Justify your answer.

Problem 4. (Optional–Ungraded) In your new career as a tennis player, you want to determine how best to hit a lob shot, which travels high enough to avoid the players on the other side of the net but low enough to avoid hitting lighting fixtures hanging from the ceiling. Let's just consider the problem in two-dimensional space, where the floor is the x-axis, and the y-axis is vertical (see Fig. 4(a)). The lob needs to clear the net, which is of height h along the y-axis. The lob needs to pass above positions likely occupied by the opposing player, which are given as a set of points $P = \{p_1, \ldots, p_m\}$, and it must pass below a set of locations of light fixtures $Q = \{q_1, \ldots, q_n\}$, where $p_i = (p_{i,x}, p_{i,y})$, and $q_j = (q_{j,x}, q_{j,y})$. (Both sets are unsorted.) Finally, it needs to land within the court, which is of length ℓ.

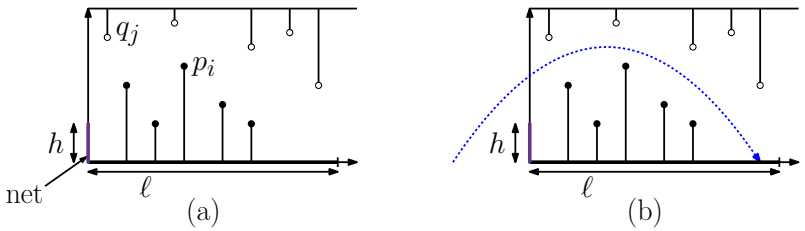

Figure 4: Lob shot.

By standard physics, the shot will travel along a parabolic path, given by the equation $y = -gx^2 + bx + c$ of a parabola, where g is a fixed positive constant determined by the force of gravity (e.g., 32 ft/sec^2), and reals b and c are based on the initial location, direction, and speed with which the ball is hit. Present an algorithm which determines whether it is possible to find real values b and c to satisfy all the following requirements (see Fig. 4(b)). The ball's trajectory must:

(a) pass above the net of height h along the y-axis

(b) pass above all the opposing player points in P

(c) pass below all the lighting fixture points in Q

(d) land on the ground within the court (within distance ℓ of the net)

If such a shot is exists, return the one that travels as high as possible above the net along the y-axis. Your algorithm should run in expected time $O(m+n)$.

Note: Challenge problems are not graded as part of the homework. The grades are recorded separately. After final grades have been computed, I may "bump-up" a grade that is slightly below a cutoff threshold based on these extra points. (But there is no formal rule for this.)

Challenge Problem: (This problem looks complicated, but it has a remarkably simple, cute solution.) The following problem takes place on a square integer grid in the plane. You are given a box of height h and width w. A number of squares in the box contain mirrors that are angled at $\pm 45°$ (see Figure 5(a)). A laser beam enters the box horizontally in the top-left corner, and it is reflected whenever it hits a mirror (see Figure 5(b)). The experiment is considered *successful* if the laser beam eventually exits horizontally through the lower right corner, and otherwise it is a *failure*.

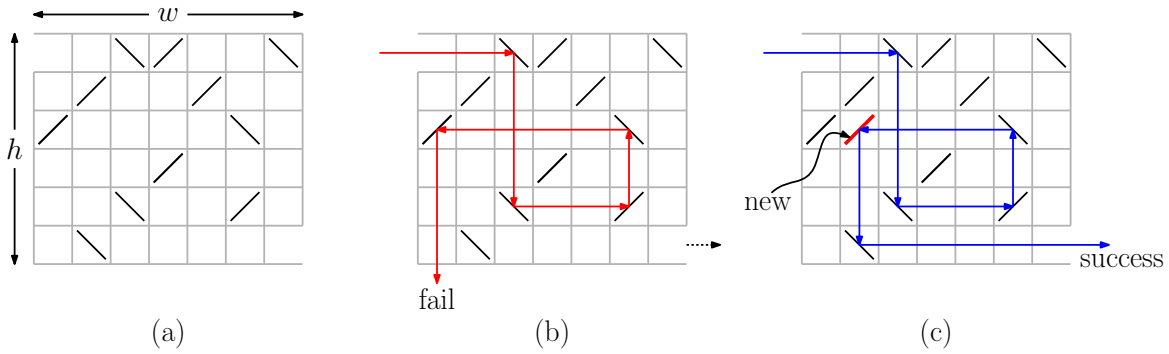

Figure 5: (a) Shooting a laser beam (unsuccessful) and (b) adding a mirror to make it successful.

Given such a box of dimensions $h \times w$, and given the directions are orientations of n mirrors in the box, there are three possibilities:

(i) The laser shot from s successfully arrives at t, following the existing set of mirrors.

(ii) (i) does not hold, but it is possible to insert a *single* new mirror within the box so that the laser beam now hits t (see Figure 5(b)).

(iii) (i) does not hold, but it is *not* possible to reach t by the insertion of a *single* mirror.

Let's assume that case-(i) does not occur. Design an efficient algorithm to determine whether (ii) or (iii) applies. If (ii) applies, determine the square and the orientation of a mirror that leads to success. (There may be many, and your algorithm can return any of them.)

As an aid to your algorithm, you may assume that you have access to a data structure which, given any horizontal or vertical ray, determines the first mirror that the ray strikes in $O(\log n)$ time. Given this data structure, there is a solution that runs in $O(n \log n)$ time.

Guidance for Writing LP Reductions: In an linear programming (LP) reduction, you should explain the following:

- how the solution space is modeled as a vector (and in what dimension),
- what are the constraints,
- what is the objective function (and express it as a vector)
- how to interpret the result (including the cases feasible, unbounded, infeasible)

Here is an example.

Sample Problem: Given two sets of points $R = \{r_1, \ldots, r_n\}$ and $B = \{b_1, \ldots, b_n\}$, both in $\mathbb{R}^3$, present an algorithm that determines whether their exists a plane h in $\mathbb{R}^3$ such that all the points of R lie on or above h and all the points of B lie on or below h.

Sample solution: We reduce the problem to linear programming in $\mathbb{R}^3$. Let's assume that each $r_i \in R$ is given in coordinate form as $(r_{i,x}, r_{i,y}, r_{i,z})$ and similarly for B. Let's model h by the equation $z = ax + dy + e$, for some real parameters a, d, and e. To enforce the condition that each r_i lies on or above h and each b_j lies on or below it, we add the constraints

$$\begin{aligned} r_{i,z} &\geq ar_{i,x} + dr_{i,y} + e, \qquad \text{for } 1 \leq i \leq n \\ b_{j,z} &\leq ab_{j,x} + db_{j,y} + e, \qquad \text{for } 1 \leq j \leq n. \end{aligned}$$

We then invoke LP with $2n$ constraints in $\mathbb{R}^3$ (with the variables (a, d, e)). Since this is a yes-no answer, we don't really care about the objective function. We can set it arbitrarily, for example, "maximize e" (which is equivalent to using the objective vector $c = (0, 0, 1)$).

We interpret the LP's result as follows. If the result is "infeasible", then we know that no such plane exists. If the answer is "feasible" or "unbounded", then we assert that such a plane exists (assuming general position). This is clearly true if the result is "feasible", since we can just take h to be the plane associated with the optimum vertex (a, d, e). If the result is "unbounded", then the plane is vertical, but there exists a perturbation such that R lies above and B lies below.

By the way, if you wish to be even more formal (which I rarely require), you can express the LP in standard form as maximizing c^T form as $Ax \leq b$, where x is the symbolic vector $(a, d, e) \in \mathbb{R}^3$, and A and b can be expressed as

$$\begin{bmatrix} r_{1,x} & r_{1,y} & 1 \\ \vdots & \vdots & \vdots \\ r_{m,x} & r_{m,y} & 1 \\ -b_{1,x} & -b_{1,y} & -1 \\ \vdots & \vdots & \vdots \\ -b_{n,x} & -b_{1,y} & -1 \end{bmatrix} \begin{bmatrix} a \\ d \\ e \end{bmatrix} \leq \begin{bmatrix} r_{1,z} \\ \vdots \\ r_{m,z} \\ -b_{1,z} \\ \vdots \\ -b_{n,z} \end{bmatrix}$$