1. Suppose you are given a set of n halfplanes that define a convex polygon P in the plane. Show that each of the following objects inside P can be found in $O(n)$ expected time using linear programming. ("Axis-aligned" means all are edges either horizontal or vertical.)

   (a) The largest axis-aligned square inside P.

   Solution: Suppose the given halfspaces have the form $a_i x + b_i y \le c_i$. We can compute the largest square inside P in $O(n)$ expected time by solving the following linear program using Seidel's algorithm. The LP has three variables: the coordinates (x, y) of the bottom left corner of the square, and the side length w of the square.

maximize		w		
subject to	$a_i x$	$+ b_i y$	$\le c_i$	for all i
	$a_i(x+w)$	$+ b_i y$	$\le c_i$	for all i
	$a_i x$	$+ b_i(y+w)$	$\le c_i$	for all i
	$a_i(x+w)$	$+ b_i(y+w)$	$\le c_i$	for all i

   The four groups of inequalities respectively state that the bottom-left corner, the bottom-right corner, the top-left corner, and the top-right corner lie inside P. ■

   Rubric: 2 points.

   (b) The axis-aligned rectangle inside P with maximum perimeter.

   Solution: Again, suppose the given halfspaces have the form $a_i x + b_i y \le c_i$. We can solve the following linear program in $O(n)$ expected time using Seidel's algorithm. The four variables are the coordinates (x, y) of the bottom left corner, the (horizontal) width w, and the (vertical) height h.

maximize		$2(w+h)$		
subject to	$a_i x$	$+ b_i y$	$\le c_i$	for all i
	$a_i(x+w)$	$+ b_i y$	$\le c_i$	for all i
	$a_i x$	$+ b_i(y+h)$	$\le c_i$	for all i
	$a_i(x+w)$	$+ b_i(y+h)$	$\le c_i$	for all i

   Again, the four groups of inequalities respectively state that the four corners of the rectangle lie inside P. ■

   Rubric: 2 points.

(c) Two interior-disjoint axis-aligned squares inside P with maximum total perimeter.

Solution: Again, suppose the given halfspaces have the form $a_i x + b_i y \le c_i$. Any pair of interior-disjoint squares can either be separated by a horizontal line or by a vertical line. We consider each of these cases separately; for each case, we can find the optimal squares using Seidel's algorithm in $O(n)$ expected time.

The following linear program describes the minimum-perimeter pair of squares in P separated by a vertical line. The LP has six variables: the bottom-left coordinates (x, y) and width w of the left square, and the bottom-left coordinates (x', y') and width w' of the right square.

$$\begin{array}{lllll}
\text{maximize} & & 4(w+w') & & \\
\text{subject to} & & x+w & \le x' & \\
& a_i x & + b_i y & \le c_i & \text{for all } i \\
& a_i(x+w) & + b_i y & \le c_i & \text{for all } i \\
& a_i x & + b_i(y+h) & \le c_i & \text{for all } i \\
& a_i(x+w) & + b_i(y+h) & \le c_i & \text{for all } i \\
& a_i x' & + b_i y' & \le c_i & \text{for all } i \\
& a_i(x'+w') & + b_i y' & \le c_i & \text{for all } i \\
& a_i x' & + b_i(y'+h) & \le c_i & \text{for all } i \\
& a_i(x'+w') & + b_i(y'+h) & \le c_i & \text{for all } i
\end{array}$$

The horizontal case is similar; in fact the only difference is the first constraint, which becomes $y + w \le y'$. ■

Rubric: 3 points.

(d) The largest circle inside P.

Solution: Again, suppose the given halfspaces have the form $a_i x + b_i y \le c_i$. The Euclidean distance from any point (p, q) to the line $ax + by = c$ is exactly

$$\frac{ap + bq - c}{\sqrt{a^2 + b^2}}.$$

For each index i, precompute the following values:

$$\hat{a}_i = \frac{a_i}{\sqrt{a_i^2 + b_i^2}} \qquad \hat{b}_i = \frac{b_i}{\sqrt{a_i^2 + b_i^2}} \qquad \hat{c}_i = \frac{c_i}{\sqrt{a_i^2 + b_i^2}}$$

Then the lines $\hat{a}_i x + \hat{b}_i y = \hat{c}_i$ and $a_i x + b_i y = c_i$ are identical, and for any point (p, q), we have $\hat{a}_i p + \hat{b}_i q \le \hat{c}_i$ if and only if $a_i p + b_i q \le c_i$.

The following linear program has three variables: the coordinates (p, q) of

the center of the circle, and the radius r of the circle.

$$\begin{array}{|l|}\hline \text{maximize} \quad r \\ \text{subject to} \quad \hat{a}_i p + \hat{b}_i q \le \hat{c}_i - r \quad \text{for all } i \\ \hline \end{array}$$

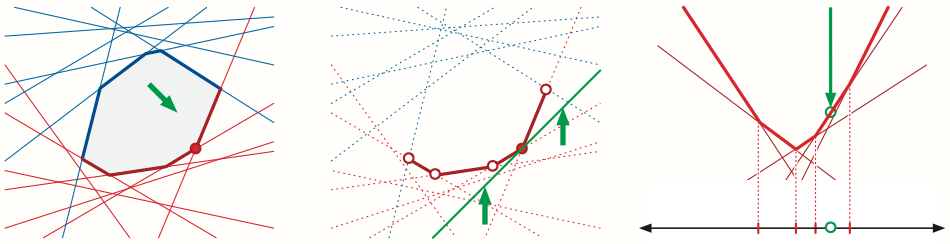
Each inequality specifies that the center point (p, q) not only satisfies the constraint $a_i p + b_i q \le c_i$ but also has Euclidean distance at least r from the line $a_i x + b_i y = c$. Thus, for any feasible solution (p, q, r), the entire circle of radius r centered at (p, q) lies inside each halfplane $a_i p + b_i q \le c_i$, and therefore inside the polygon P. ■

Rubric: 3 points.

2. (a) Describe an algorithm to answer ***two***-dimensional linear-programming queries in $O(\log n)$ time. How much space does your data structure use? How much preprocessing time do you need construct it?

Solution: Let $c = (\alpha, \beta)$ be the given objective vector. Scaling c doesn't change the output of the LP query, so we can assume without loss of generality that $\beta = \pm 1$. (I'm deliberately ignoring the easy degenerate case $\beta = 0$.) I'll describe a data structure to answer queries with $\beta = -1$; a symmetric data structure answers queries with $\beta = +1$. Now the objective vector is specified by a single real number α.

We can interpret any linear programming query geometrically as follows. Let U and L denote the vertices on the upper and lower boundary of the feasible region $\bigcap P$. (The intersection $U \cap L$ contains only the leftmost and rightmost vertices of P.) We can compute the upper and lower boundaries of P in $O(n \log n)$ time using (the dual of) any convex hull algorithm. Given a query vector $(\alpha, -1)$, we need to find the *highest* line ℓ with slope α, such that every point in L is on or above ℓ.



Now consider the projective dual formulation of this problem: We are looking for the *lowest point* ℓ^* with x*-coordinate* α that lies on or *above* every *line* in L^*. In other words, ℓ^* is the intersection point of the vertical line $x = \alpha$ and the upper envelope of L^*.

Finally, let X be the projection of the vertices of the upper envelope of L^* to the x-axis. (These x-coordinates are precisely the slopes of edges in the lower boundary of P.) The points X subdivide the x-axis into intervals, each of which is the projection of an edge of the lower envelope of L^*, and therefore correspond to a point in L. (More directly, X partitions the space of possible query parameters α according to the correct query output.)

Our data structure consists of a sorted array of values in X. This array uses at most $\boldsymbol{O(n)}$ ***space*** and can be computed in $\boldsymbol{O(n \log n)}$ ***preprocessing time***. To answer a linear-programming query (with $\beta < 0$), we perform a binary search for the interval between adjacent values in X containing the query parameter α; in $\boldsymbol{O(\log n)}$ ***query time***. ■

Rubric: 4 points. This is more detail than necessary for full credit.

(b) Describe an algorithm ***three***-dimensional linear-programming queries in $O(\log n)$ time. How much space does your data structure use? How much preprocessing time do you need construct it?

Solution: Let $c = (\alpha, \beta, \gamma)$ be the given objective vector. As in part (a), scale-invariance allows us to assume that $\beta = \pm 1$. I'll describe a data structure to answer queries with $\gamma < 0$; a symmetric data structure answers queries with $\gamma > 0$. (The degenerate case $\gamma = 0$ can be solved using the data structure in part (a).) Assume without loss of generality that $\gamma = -1$, so the objective vector is specified by a point (α, β) in the plane.

As in part (a), we partition the parameter plane into regions, so that the answer for all query points (α, β) within each region is the same. We can characterize this partition geometrically by following the derivation in part (a):

- Let L be the vertices on the lower boundary of the feasible polyhedron. A linear programming query with objective $(\alpha, \beta, 1)$ asks for the highest plane π with gradient (α, β), such that every point in L is on or above π.
- The duality $(a, b, c) \leftrightarrows z = ax + by - c$ preserves vertical orderings of points and planes: A point p is above a plane π if and only if the dual point π^* is above the dual plane p^*. Viewed through this duality lens, an LP query asks for the *lowest* point of the form (α, β, z) that lies on or above every dual plane in L^*. In other words, we want the intersection point of the vertical line $(x, y) = (\alpha, \beta)$ with the upper envelope of L^*.
- Finally, let X be the projection of the upper envelope of L^* to the xy-plane. X is a planar-straight-line graph whose faces correspond to the facets of the upper envelope of L^*. The answer to a linear programming query with objective $(\alpha, \beta, -1)$ is determined by the face of X that contains the point (α, β).

To construct X, we start by computing the intersection P of the input halfspaces in $O(n \log n)$ time, using (the dual of) any 3D convex hull algorithm. (In particular, we can build the upper envelope of the upper halfspaces, as well as the lower envelope of the lower halfspaces, in $O(n \log n)$ expected time using the randomized incremental algorithm for weighted Voronoi diagrams. We can also find the "horizon" curve where these two envelopes intersect in $O(n \log n)$ expected time.) Euler's formula implies that P has complexity $O(n)$. The upper envelope of L^* is dual to the lower boundary of P, so we don't need any additional computation to compute the upper envelope of L^* or its projection X.

Finally, we build a trapezoidal decomposition of X in $O(n \log n)$ expected time; the history dag of the trapezoidal decomposition answers point-location queries in $O(\log n)$ expected time, as required. The history dag data structure requires ***$O(n)$ space***, and its construction requires ***$O(n \log n)$ preprocessing time***. ■

Rubric: 6 points. This is more detail than necessary for full credit.

3. (a) Describe a linear program whose solution encodes the line with minimum L_1 error for a given set P of points.

Solution: The following linear program has $n+2$ variables — the coefficients a and b of the solution line $y = ax - b$, and a *residual* variable R_i for each input point — and $2n$ constraints.

$$\begin{array}{lll} \text{minimize} & \displaystyle\sum_{i=1}^{n} R_i & \\ \text{subject to} & ax_i - b + R_i \le y_i & \text{for all } i \\ & ax_i - b - R_i \ge y_i & \text{for all } i \end{array}$$

For each index i, the two constraints involving x_i and y_i are equivalent to the non-linear inequality

$$R_i \ge |y_i - ax_i + b|.$$

If we fix the variables a and b, the objective function $\sum_i R_i$ is minimized by setting $R_i = |y_i - ax_i + b|$ for every i. Thus, in any optimal solution, the objective function is the L_1 error of the line $y = ax - b$, as required. ■

(a′) **Extra credit:** Describe a polynomial-time algorithm to compute the line minimizing this error metric. *[Hint: What does your linear program tell you about the structure of the optimal solution?]*

Solution: In the absence of degeneracies, the optimal solution $(a, b, R_1, \ldots, R_n)$ must satisfy exactly $n+2$ of the $2n$ constraints with equality. Thus, there are at least two indices i such that $ax_i - b + R_i = y_i = ax_i - b - R_i$, and therefore $R_i = 0$. In other words, the optimal line $y = ax - b$ must pass through at least two (and assuming general position, *exactly* two) input points.

For each pair of input points p and q, we can compute the L_1 error of the line through p and q in linear time. Thus, we can compute the line with minimum L_1 error by brute force in $O(n^3)$ time. ■

(b) Describe a linear-time algorithm to compute the line with minimum L_∞ error for a given set P of points.

Solution: The following linear program has three variables — the coefficients a and b of the solution line $y = ax - b$, and a *residual* variable R — and $2n$ constraints.

$$\begin{array}{lll} \text{minimize} & R & \\ \text{subject to} & ax_i - b + R \le y_i & \text{for all } i \\ & ax_i - b - R \ge y_i & \text{for all } i \end{array}$$

For each index i, the two constraints involving x_i and y_i are equivalent to the non-linear inequality

$$R \ge |y_i - ax_i + b|.$$

Thus, the constraints are collectively equivalent to the inequality

$$R \geq \max_i |y_i - ax_i + b| .$$

For any fixed values of a and b, the variable R is obviously minimized when $R = \max_i |y_i - ax_i + b|$. Thus, in any optimal solution, the objective function is the L_∞ error of the line $y = ax - b$, as required.

Because this LP has only three variables, we can solve it in $O(n)$ expected time using Seidel's algorithm. ■

4. Let S be a set of n line segments in the plane, each parallel to one of k different directions. Segments in S may or may not intersect. A *stabbing line* for S is a line that intersects every segment in S. Describe an algorithm to determine whether S has a stabbing line. For full credit, your algorithm should run in $O(kn)$ time.

Solution: We solve this problem by reducing it to k instances of the red-blue point separation problem. Without loss of generality, assume that one of the families of input segments is vertical. Sort and index the slopes of the non-vertical segments as $a_1 < a_2 < \cdots < a_{k-1}$, and to simplify notation, let $a_0 = -\infty$ and $a_k = +\infty$. For each index i, let S_i denote the subset of segments with slope a_i.

Suppose there is a stabbing line ℓ with equation $y = ax - b$. For every non-vertical segment with slope *less* than a, the left endpoint is above ℓ and the right endpoint is below ℓ. Similarly, for every non-vertical segment with slope *greater* than a, the left endpoint is *below* ℓ and the right endpoint is above ℓ.

Thus, for any index j, there is a stabbing line with slope in the range $a_{j-1} < a < a_j$ if and only if there is a line ℓ that satisfies the following constraints:

- For every segment in S_0, the upper endpoint is above ℓ and the lower endpoint is below ℓ.
- For every segment in $\bigcup_{i=1}^{j-1} S_i$, the left endpoint is above ℓ and the right endpoint is below ℓ.
- For every segment in $\bigcup_{i=j}^{k-1} S_i$, the right endpoint is above ℓ and the left endpoint is below ℓ.

These $2n$ constraints define a linear program with two variables (slope and y-intercept) that is feasible if and only if there is a stabbing line with slope between a_{j-1} and a_j. We test the feasibility of this linear program in $O(n)$ expected time using Seidel's algorithm.

To find a stabbing line with no slope restriction, we repeat the previous feasibility test for each index j from 1 to k. The resulting algorithm runs in $O(kn)$ expected time. ■

Rubric: 10 points.