Solutions to Quiz 2

Solution 1:

(a) LP: This recurrence arose in the expected-case analysis of an LP of size n in $\mathbb{R}^d$, where p_n is the probability that the last insertion resulted in a change in the optimal vertex.

(b) $O(d!n)$

(c) $O(i)$: Actually, i is the maximum, which arises when each of the other $i-1$ sites removes an arc of the beach line for a single site.

(d) $O(1)$: More precisely, we proved a bound of $12 + 4/i$.

(e) $O(i)$: In the worst case, a single segment can be responsible for roughly $2i$ new trapezoids, one for each of the segment endpoints.

Solution 2: We will parameterize the square as three numbers: its lower-left corner coordinates (c_x, c_y) and its side length s. This naturally suggests an LP in $\mathbb{R}^3$, with the parameter vector (c_x, c_y, s). The square has corners at (c_x, c_y), $(c_x + s, c_y)$, $(c_x, c_y + s)$, and $(c_x + s, c_y + s)$. Clearly, $s \geq 0$. In order for the square to contain the origin, we have the constraints:

$$c_x \leq 0 \qquad c_y \leq 0 \qquad c_x + s \geq 0 \qquad c_y + s \geq 0.$$

For each line $\ell_i : y = a_i x + b_i$, we require that all four corners of the square lie on the same side of the line as the origin. Thus, if $b_i > 0$, all the corners must lie below the line, and if $b_i < 0$, all the corners must lie above it. This leads to four linear constraints per line. To be parsimonious, we need only one constraint per line, depending on the intercept and slope.

- $b_i > 0$ and $a_i \geq 0$ (Upper-left corner is below) $c_y + s \leq a_i c_x + b_i$
- $b_i > 0$ and $a_i < 0$ (Upper-right corner is below) $c_y + s \leq a_i(c_x + s) + b_i$
- $b_i < 0$ and $a_i < 0$ (Lower-left corner is above) $c_y \geq a_i c_x + b_i$.
- $b_i < 0$ and $a_i \geq 0$ (Lower-right corner is above) $c_y \geq a_i(c_x + s) + b_i$

Combining everything yields a total of $5 + n$ linear constraints in $\mathbb{R}^3$. The objective is to maximize s, which implies that objective vector is $(0, 0, 1)$.

We run the LP algorithm on the system above, which has an expected running time of $O(n)$. The LP cannot be infeasible, since the origin ($c_x = c_y = s = 0$) satisfies all constraints. It can be unbounded (e.g., if none of the lines intersect the first quadrant), in which case we report that the solution is unbounded.

Note that there is some redundancy in our solution. The initial five constraints on c_x, c_y, and s could be eliminated. This is because the corner constraints have been engineered to apply to the side of each line that contains the origin. Thus, the corner points of the square lie within the face of the line arrangement that contains the origin. By maximizing s, we generate a solution with a nonnegative side length.

Solution 3:

(a) $O(n)$: We claim that each exposed vertex is on an edge of the Voronoi diagram (see Fig. 1(a)). This is because the vertex is at unit distance from the associated sites, and no other site can be closer (otherwise the vertex would lie in the interior of some other circle and would not be exposed). Clearly, each edge can intersect a unit circle at most twice, hence the total number of exposed vertices is at most $2n$.

(b) $O(n^2)$: Every pair of circles can intersect at most twice, so the maximum number of intersections is at most $2\binom{n}{2} = n(n-1)$. Since at most a linear number are exposed, the remainder are hidden.

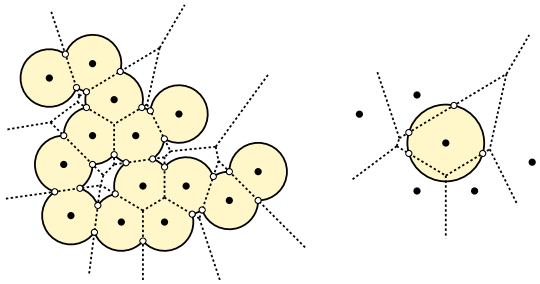


Figure 1: Counting exposed vertices.

(c) As observed in (a), each exposed vertex lies on the edge of the Voronoi diagram. We begin by computing the Voronoi diagram of the point set P in $O(n \log n)$ time (e.g., by Fortune's algorithm). We then inspect each edge to determine whether there are 0, 1, or 2 points on the edge whose distance from its sites equals 1 (see Fig. 1(b)). We then add this quantity to a counter. Since the Voronoi diagram has $O(n)$ edges, this can be done in total $O(n)$ time (assuming any reasonable representation for the Voronoi diagram, e.g., a DCEL).

Solution 4:

(a) We will reduce the ray-shooting problem to point location. Given the shot description (c, u_x, u_y), let $s = u_y/u_x$ denote the slope of the line carrying the ball. The line equation along which the ray flies is $\ell : y = sx + c$. Through point-line duality, we can treat this as a point $\ell^* = (s, -c)$ in the dual plane.

Let $A = \{a_1, \ldots, a_n\}$ and $B = \{b_1, \ldots, b_n\}$ denote the lower and upper endpoints respectively, where

$$a_i = \left(p_{i,x}, p_{i,y} - \frac{h_i}{2}\right) \qquad \text{and} \qquad b_i = \left(p_{i,x}, p_{i,y} + \frac{h_i}{2}\right)$$

(see Fig. 2(a)). The ray misses the ith segment if and only if ℓ passes below a_i or above b_i.

Viewing the problem in the dual plane, we have the dual point $\ell^* = (s, -c)$ and dual lines a_i^* and b_i^*. By the order-reversing property of duality, the ray misses the ith segment if and only if the dual point ℓ^* lies above the dual line a_i^* or below b_i^*. We seek the smallest index i such that either of these conditions occurs.

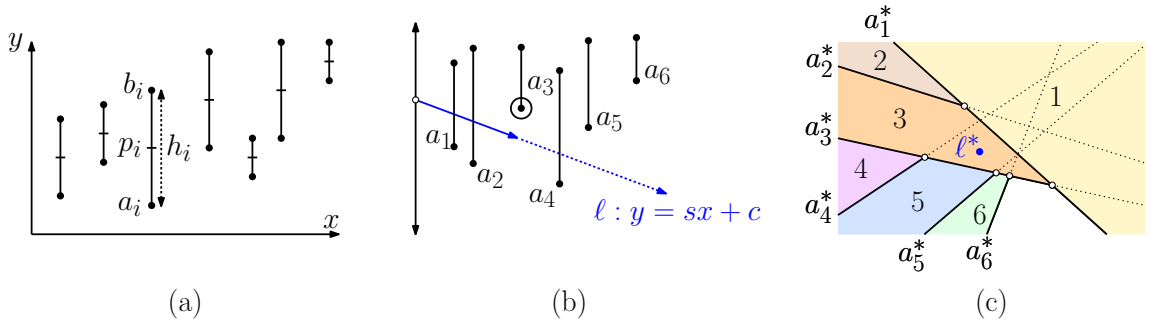


Figure 2: Ray-shooting queries. In (b) and (c) we consider only the lower endpoints a_i.

Let's first consider how to do this for just the points of A (see Fig. 2(b)). Assume these points are sorted left to right. Let's first consider the dual line a_1^*. We label the region above a_1^* with the index 1. Otherwise, consider the portion of line a_2^* below a_1^*, and label the region above a_2^* (but below a_1^*) with index 2. Continuing inductively, we consider the portion of the line a_i^* that lies below the lower envelope of $\{a_1^*, \ldots, a_{i-1}^*\}$, and label the region between the line and lower envelope with i. On termination, every region has been properly labeled with the vertical segment such that the ray passes below this segment (see Fig. 2(c)). The lower envelope corresponds to a ray that passes above all the points of A. Each line is trimmed when it first hits another line, resulting in n non-intersecting segments. We construct the trapezoidal map and associated point-location data structure for the resulting set of segments.

We repeat the process for the points of B, creating the separate point-location data structure. Both structures have space $O(n)$ and query time $O(\log n)$. If either returns an index, we return the smaller of the two indices. If neither does, then we report that the ray intersects all the segments.

(b) It is interesting that these two seemingly similar problems (missing and hitting) have very different complexities. The intuitive reason is that if segment i is the first to be missed, we must have hit all the first $i-1$ segments, which is corresponds to a convex set in dual space. In contrast, if i is the first segment hit, then we must miss all the first $i-1$ segments. Since each segment can be missed in two different ways (from the top or the bottom), the resulting region in dual space is not convex, and indeed it is not even connected. This observation is a first step in the solution, but to establish a lower bound, we need to demonstrate a set of segments that shows that generates super-linear complexity.

By the equivalence of primal and dual space, it suffices to show that there exists a set of n vertical segments in the primal space so that the solution space has $\Omega(n^2)$ connected components in the dual space. The trick is to come up with a set of segments can be "pierced" in $\Omega(n^2)$ essentially different ways.

Rather than determining which segment is first hit, we'll consider the simpler decision problem of whether *any* segment is hit. This is most easily done with small segments, and we'll consider the extreme case where each segment degenerates to a point by setting $h_i = 0$ for all i. (If this bothers you, just make h_i infinitesimally small.) The point configuration consists of placing $n/2$ points on a vertical line at $x = 1$ and $n/2$ on a vertical line at $x = 2$ (see Fig. 3(a)).

(This violates general position, but a small perturbation can be applied without affecting the proof.)

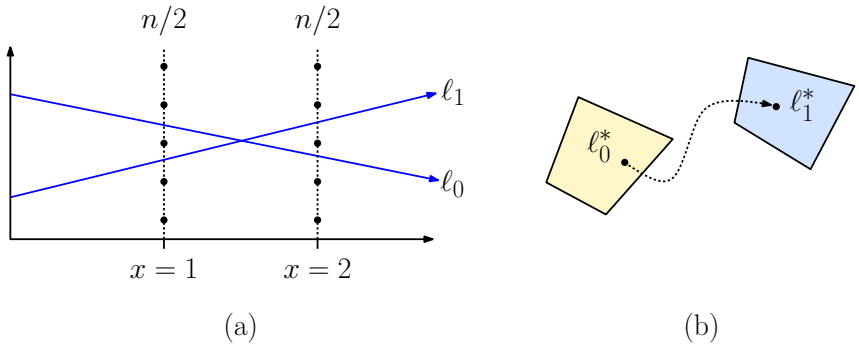


Figure 3: Lower bound on ray-hitting queries.

We assert that the solution space in dual space has at least $(n/2+1)^2 = \Omega(n^2)$ connected components. Observe that these points subdivide the lines $x = 1$ and $x = 2$ into $n/2+1$ *gaps*. Any line that misses all the segments passes through two of these gaps. Consider two lines ℓ_0 and ℓ_1 that pass through a different pair of gaps (see Fig. 3(a)). Observe that any continuous motion that maps one line to the other must pass through at least one of the points of the set. In the dual space, this is equivalent to saying that any path between ℓ_0^* and ℓ_1^* must pass through at least point where the answer changes from "miss" to "hit". Thus, these two dual points lie in different connected components of the solution space (see Fig. 3(b)). Since the solution space has $\Omega(n^2)$ connected components, any subdivision that yields a valid solution must have at least $\Omega(n^2)$ total complexity. (While we have stated from the limited perspective of a point-location problem in dual space, this can be generalized to hold in the algebraic decision tree model.)