Solutions to Sample Final Problems

Solution 1:

(a) The maximum number of scan reflex vertices is $n/2 + O(1)$. A worst-case example consists of a long sequence of zig-zagging edges, where every other vertex is a scan reflex vertex.

(b) P_1^* and P_2^* are two collections of lines, each associated with an upper and lower envelope. We claim that the envelope associated with P_2^* contains the envelope associated with P_1^*. The easiest way to see this is to imagine an arbitrary line translating continuously downwards through P_1 and P_2. In the dual plane, this corresponds to a dual point sweeping vertically upwards (see Fig. 1). Clearly the sweep line ℓ contacts the boundary of P_2 first and last, and (by the incidence properties of duality) the sweeping point ℓ^* contacts the envelope boundaries of P_2^* first and last.

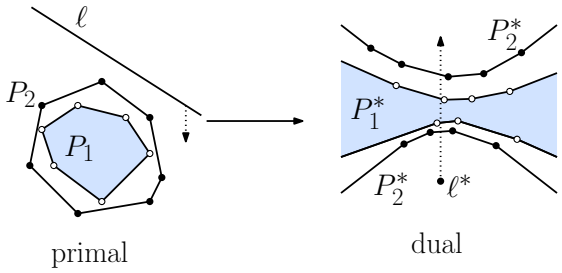


Figure 1: Dual of nested convex polygons.

(c) We follow the analysis given in lecture. Excluding the leftmost trapezoid, charge each trapezoid to its leftmost vertex or intersection point. Each of the n segment left endpoints will be charged by two trapezoids (one above right and one below right), each of the n segment right endpoints will be charged by one trapezoid (to its right), and each of the m intersection points is charged by three trapezoids (above-right, middle-right, and below-right). Thus, the total number of charges is $3n + 3m$. Combining and including the leftmost trapezoid implies that the total number of trapezoids is $3n + 3m + 1$.

(d) In the worst case, every time we add a halfplane in LP the current optimum vertex is infeasible, which triggers a call to the 1-dimensional LP problem, which takes $O(n)$ time. Thus, the final (worst-case) running time is $O(n^2)$.

(e) (i) True: The EMST is a subgraph of the Delaunay triangulation

(ii) True: The Delaunay triangulation maximizes the minimum angle

(iii) False: The Delaunay triangulation does not generally minimize the maximum angle

(iv) False: The Delaunay triangulation is not necessarily the minimum weight triangulation

(v) True: The Delaunay triangulation is a t-spanner, for $t \approx 2.418$.

(f) Let v, e, and f denote the numbers of vertices, edges, and faces in the dodecahedron. Because each face is a pentagon, we have $5f = 2e$ (recall that summing the degree of the faces counts each edge twice), implying that $e = 30$. By Euler's formula, we have $2 = v - e + f = v - 30 + 12$, so $v = 20$.

(g) Since each vertex of a Voronoi diagram (not counting the vertex at infinity) is of degree three, we have $3v \approx 2e$, or $e - v \approx e/3$. By Euler's formula we have $v - e + f = 2$, which means that $f - 2 = e - v \approx e/3$, and therefore $e \approx 3f = 3n$.

(h) When the ith site is added, it is possible that it might be joined to *all* the prior sites. (This happens if the circumcircles of all the triangles contain a common point, and this is the next point to be added.)

(i) The total complexity of a hyperplane arrangement in $\mathbb{R}^d$ is $\Theta(n^d)$, so $O(n^3)$ is certainly an upper bound. To obtain the exact value, observe that each of the n planes contains an arrangement of $n - 1$ lines. Each edge is counted twice (since each edge lies on two planes). Therefore, the exact number of edges is $n(n-1)^2/2$.

(j) (i) Convex-convex: $n + m$—As shown in class; (ii) Convex-Arbitrary: $O(nm)$—By triangulating Q and applying part (i); (iii) Arbitrary-arbitrary: $O(n^2m^2)$—As shown in class.

(k)

(i) Axis-parallel squares: $\Theta(n)$—A system of pseudo-disks

(ii) Axis-parallel rectangles: $\Theta(n^2)$—Can form a cross-hatched pattern with $\Omega(n^2)$ vertices, and there are at most $O(n)$ edges so at most $O(n^2)$ intersections

(iii) General rectangles: $\Theta(n^2)$—Same as (ii)

(iv) 4×1 axis-parallel rectangles: $\Theta(n)$—Each rectangle is the union of four squares, so it is just the union of $4n$ pseudo-disks.

Solution 2: Recall that the edges of P are oriented in counterclockwise order around P's boundary. For each edge e_i, let h_i denote the halfplane whose supporting line passes through this edge, and which lies to the left of this directed edge. Let H denote the resulting set of n halfspaces. (Each can be expressed as $\text{orient}(v_i, v_{i+1}, q) > 0$.) We assert that P is star-shaped if and only if the intersection of the halfspaces of H is nonempty, and q may be taken to be any point in the interior of this intersection.[1] Assuming this is true, it follows that we can test this in $O(n)$ time by reduction to LP in $\mathbb{R}^2$. (Since this just a feasibility test, the objective function is arbitrary.)

To prove the assertion, we claim that a point q can "see" every point p on the boundary of P if and only if q lies in the interior of the intersection of the halfspaces of H. Clearly, if q lies outside any halfspace of $h_i \in H$, it cannot see the edge e_i (because then q would lie "behind" e_i). Conversely, if a point q lies within the intersection of halfplanes of H then we claim that for any point p on the boundary of P the open line segment $\overline{qp}$ lies entirely within the interior of P. Suppose not (see Fig. 2). By hypothesis, q must lie within the halfspace associated with the edge on which p lies. Since both p and q lie within P, the segment $\overline{qp}$ must cross the boundary of P at

[1] It might seem at first that the assertion is trivial, but if we changed things slightly so that P is an open polygonal curve, the assertion would be false.

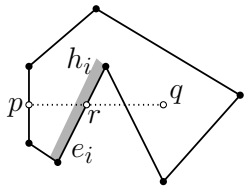


Figure 2: Star-shaped polygon.

some last point r before p. Let e_i be the edge on which r lies. Clearly p can see e_i, and so p lies within h_i. But, this implies that q does not lie within h_i, contradicting the hypothesis that q lies in the intersection of H.

Solution 3: We will show that $\lfloor n/3 \rfloor$ guards always suffice. Begin by computing a triangulation T of the simple polygon P. We assert that we can assign one of three colors to the vertices P, so that every triangle of T has three different colors (see Fig. 3). We will prove this by induction. Recall that any triangulation of a simple polygon defines a dual graph that is a binary tree. Every tree has at least one leaf. Remove this leaf (equivalently remove the associated triangle t from the triangulation), and apply the induction hypothesis to color the remaining triangulation T'. Now, when we add the leaf back (adding triangle t back), two of the color will be assigned to the edge that connects it to the rest of the triangulation, allowing us to use the remaining color for the uncolored vertex of t.

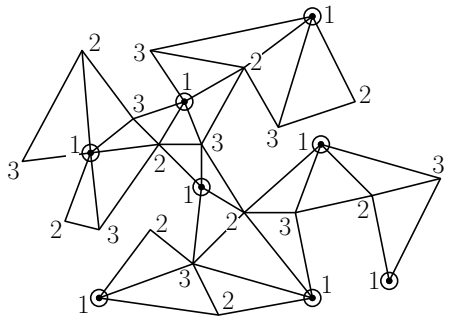


Figure 3: Coloring a triangulation and guarding. (Color 1 is the guarding set.)

Since P has n vertices, one of the color classes contains at most $\lfloor n/3 \rfloor$ vertices. Define G to be the vertices of this color class. Since every triangle of T has all three colors assigned to it, one of its vertices will be in G, implying that G is a guarding set.

Solution 4: We want to determine whether the convex hulls of two point sets R and B intersect. If we had the two convex hulls, we could determine this in $O(n)$ time by plane sweep. (Recall the divide-and-conquer algorithm for computing convex hulls from the lecture notes.)

We can do this in $O(n)$ time even without computing the hulls through linear programming. Let us make the usual general position assumptions. The key observation is that the two convex hulls do not intersect if and only if there is a line $\ell : y = ax - b$ that separates them. Either the red points lie entirely above this line and the blue points beneath, or vice versa. We will show how to do the former, and the latter is symmetrical.

The unknowns are a and b. Each point (x_i, y_i), in the red set must satisfy $y_i \geq ax_i - b$, and each point in the blue set must satisfy $y_i \leq ax_i - b$. This defines n linear inequalities, that is, n halfplanes in the (a, b) coordinate plane. We can make the objective function whatever we like, say to maximize a. We then apply linear programming.

Any feasible point (a, b) gives the coefficients of the separating line. The solution may be unbounded (e.g., if they are separated by a vertical line), but we allow this possibility and again report that they are separable. If the LP is infeasible, we repeat the process, reversing red and blue. Again, if there is such a line, the hulls do not intersect. Otherwise, we infer that it is impossible to separate the two point sets by a line, and this implies that the hulls overlap.

Solution 5: Dualize the ray r by expressing it as the line passing through the ray and dualizing this line to a point. Recall that e is vertical. The set of lines passing through e dualizes to the set of points lying within the strip bounded between two parallel lines (the duals of the upper and lower endpoints of e). To answer the query, we dualize the line passing through r to a dual point. Because we know that the ray must start on e and enter into P, each possible query is mapped to a unique dual point in this way. (This would not work for general ray shooting queries that originate from arbitrary locations in P, but with some added sophistication, this idea can be modified to work even in the case of general rays.)

Our goal will be to produce a subdivision of the strip into $O(n)$ polygonal regions, such that the region containing the dual query point uniquely characterizes the edge that is hit by the query. To do this, first triangulate P. Consider the other vertex p_i of the triangle incident to edge e (p_3 in Fig. 4). Partition the strip by inserting the dual line p_i^*. This subdivides the strip into two convex polygonal regions. The region lying above p_i^* corresponds to rays that pass below p_i and vice versa. We recursively split these two regions by applying the same process. If the edge lying below p_i is an edge e of P (it is e_2 in Fig. 4), then we label this region with e, and stop the recursive process. Otherwise, we consider the vertex p_j that is incident to the neighboring triangle. We split this region using the dual line p_j^*, into two subregions. The subregion lying above p_j^* corresponds to rays passing below p_j and vice versa.

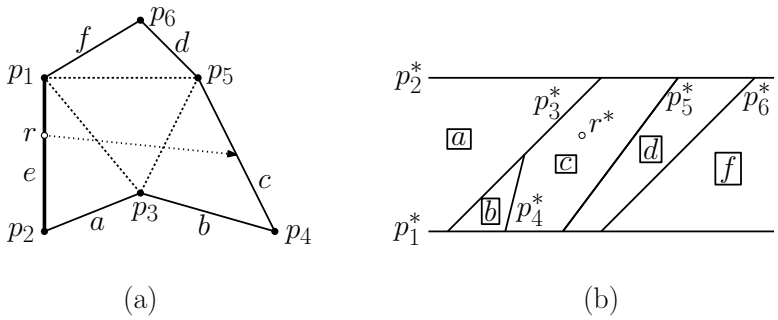
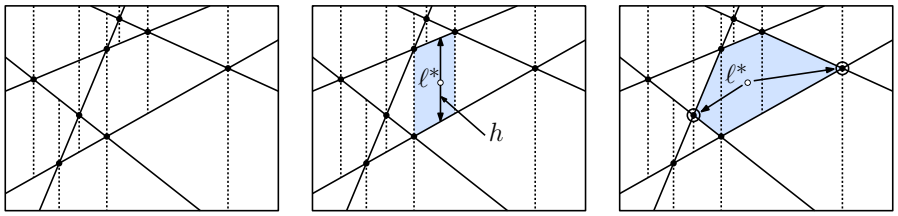


Figure 4: Ray-shooting queries.

The final result is a subdivision of the strip into at most n convex polygonal regions. (At most n, because it is easy to see that each edge of P can contribute at most one region to the subdivision.) Since there are n faces, and there are no vertices of degree two, we may use Euler's formula to prove that there are $O(n)$ vertices in the subdivision. Thus, we may build a point location data structure of size $O(n)$ to answer ray shooting queries in $O(\log n)$ time.

Solution 6: In each case, the preprocessing consists of dualizing the points of P into n lines and constructing the line arrangement. Considering the $O(n^2)$ edges of the arrangement to be line segments, we then construct a trapezoidal map for these line segments and build a point-location data structure for the resulting trapezoidal map in $O(n^2 \log n)$ time (see Fig. 5(a)). Each trapezoid of the map will store a some additional information for the purposes of answering the query. We will see that this information is of constant size, and so the total space of the data structure is dominated by the space for the trapezoidal map. Since there are $O(n^2)$ line segments, the space is is $O(n^2)$.

In order to answer a query, we will dualize the query line ℓ into a point ℓ^*, and locate the trapezoid of the map that contains this point in $O(\log(n^2)) = O(\log n)$ time. The rest of the processing will be described in each of the parts below.



(a) (b) (c)

Figure 5: Solution to Problem 2.

(a) Let ℓ^+ and ℓ^- denote the lines parallel to ℓ that define the top and bottom, respectively, of the largest empty slab that is parallel to and contains ℓ. By standard results from duality, the distance h between these lines is the difference in the y-intercepts of these two lines. In the dual, the corresponding points are the points lying immediately below and immediately above ℓ^* in the cell of the arrangement that contains ℓ^* (see Fig. 5(b)). Thus, if Δ is the trapezoid containing ℓ^* these points lie immediately above and immediately below ℓ^* in Δ. Given ℓ^* and Δ, we can compute the distance between these points in $O(1)$ time. Thus, the query time is dominated by the time to answer the point-location query, which is $O(\log n)$.

(b) Let ℓ^- and ℓ^+ denote the lines of minimum and maximum slopes, respectively, that separate $P^+(\ell)$ and $P^-(\ell)$. Consider the cell of the arrangement that contains ℓ^*. The duals of these line lie within this cell. Furthermore, since their slopes are extreme, they correspond to the leftmost and rightmost vertices of the cell (see Fig. 5(c)). (In general, the cell might be unbounded, so one or both of these vertices might not exist. However, by our assumption that $P^+(\ell)$ and $P^-(\ell)$ are not vertically separable, both slopes are bounded and hence both vertices exist.) When we build the arrangement, we can associate each cell with its leftmost and rightmost vertex. When we build the trapezoidal map, we can associate this information with each trapezoid that lies within each cell. Thus, if Δ denotes the trapezoid that contains ℓ^*, we access these vertices and output their duals to obtain the desired answer. The query time is dominated by the time to answer the point-location query, which is $O(\log n)$.

Solution 7: It is natural to reduce this to a point location query, but what is the image of a

polygonal curve in dual space? This would seem to be very complicated, but there is a deceptively simple solution.

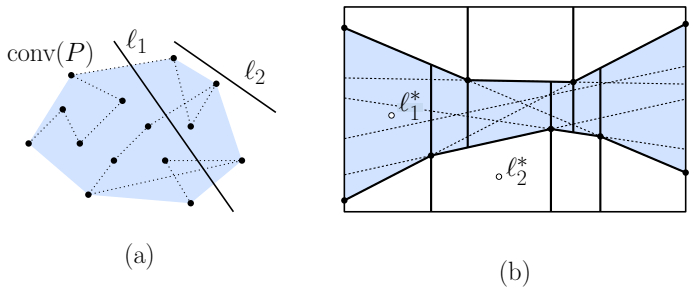


Figure 6: Path-crossing queries.

The key observation is that the path is entirely immaterial. A line intersects the path π if and only if it intersects the convex hull of the points P (see Fig. 6(a)). (To see this observe that the path is entirely contained within the convex hull, and thus any intersection with the path implies an intersection with the convex hull. Conversely, any intersection with the convex hull implies that the line separates at least one point of P from another, and therefore the path must cross the line at least once.)

We can easily solve this in the dual. We dualize the points of P. Let H be the convex hull of P. We know that in dual space the boundary of H is mapped to an upper and lower envelope of n lines in the plane. It follows from basic duality properties that the line ℓ intersects the hull if and only if its dual ℓ^* lies between these envelopes (see Fig. 6(b)). This can be tested in $O(\log n)$ time by building a point location data structure of space $O(n)$. (There are also more direct methods, but this suffices for our purposes.)

Solution 8: Build an s-WSPD for a separation factor $s = 4/\varepsilon$. (We'll see below why this is the correct separation value.) Assume that each well separated pair (P_u, P_v) is associated with a representative pair (p_u, p_v), where $p_u \in P_u$ and $p_v \in P_v$, and the cardinalities of the two subsets $n_u = |P_u|$ and $n_v = |P_v|$. It is an easy matter to augment the WSPD construction so that each node of the compressed quadtree stores these two additional pieces of information (representative and size).

To answer an ε-approximate distance query, for each pair (P_u, P_v) of the WSPD, compute the distance $\|p_u - p_v\|$. If this distance exceeds z, then add the number of pairs $n_u \cdot n_v$ to the count. On termination, output the final count. Clearly, the space and query time are both proportional to the number of well separated pairs, which is $O(ns^d) = O(n/\varepsilon^d)$.

To establish the correctness of this algorithm, consider any pair of points $(x, y) \in P \times P$. We will show that if $\|x-y\| < z/(1+\varepsilon)$ then the well-separated pair covering (x, y) will not be counted, and if $\|x - y\| \geq z(1 + \varepsilon)$, then it will be counted.

Let (P_u, P_v) be the well-separated pair that covers (x, y). By the WSPD Utility Lemma, we have

$$\|x - y\| \;\leq\; \left(1 + \frac{4}{s}\right) \|p_u - p_v\| \;=\; (1 + \varepsilon)\|p_u - p_v\|.$$

By reversing the roles of (x, y) and (p_u, p_v), a symmetrical argument shows that $\|p_u - p_v\| \leq$

$(1+\varepsilon)\|x-y\|$. Therefore we have

$$\frac{\|x-y\|}{1+\varepsilon} \;\leq\; \|p_u - p_v\| \;\leq\; \|x-y\|(1+\varepsilon).$$

Thus, if $\|x-y\| < z/(1+\varepsilon)$, then $\|p_u - p_v\| < z$, and hence none of the pairs of (P_u, P_v) will contribute to the final count. Conversely, if $\|x-y\| \geq z(1+\varepsilon)$, then $\|p_u - p_v\| \geq z$, implying that every pair of (P_u, P_v) (including (x, y)) will contribute to the final count.