Solutions to Homework 2

Solution 1:

(a) $n + m - 1$: To see that the answer is at most this quantity, draw vertical lines through each of the vertices. Since there are n and m edges in the polygonal chains, there are $(n+1)+(m+1) = n+m+2$ total vertices and hence $n+m+2$ vertical lines. This creates $(n+m+2)-1 = n+m+1$ parallel strips between each pair of lines (see Fig. 1(a)). Except for the first and last, each strip may contain a segment from each chain, and hence each of $(n+m+1)-2 = n+m-1$ strips may contribute one intersection point. Thus, there are at most $n+m-1$ intersections possible.

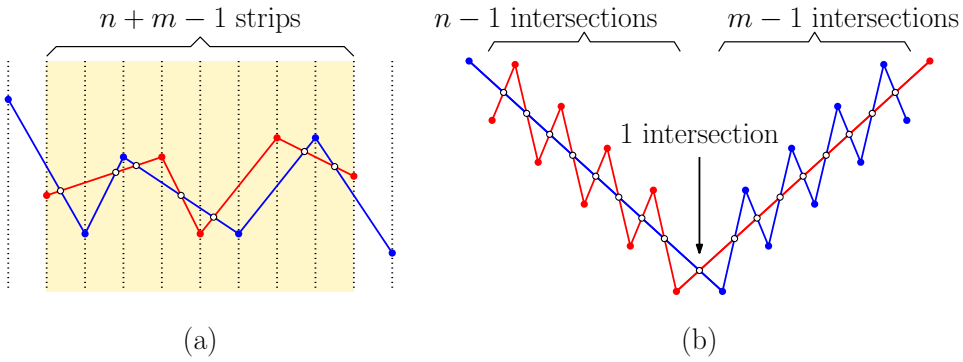


Figure 1: Number of intersections.

(b) To see that this is achievable, consider the example shown in Fig. 1(b). The first consists of a single long edge followed by $n-1$ zig-zagging edges. The second polygon has a single long edge preceded by $m-1$ zig-zagging edges. As seen in the figure, we can engineer the figure so that the two long edges intersect, and each zig-zag edge of one intersects the long edge of the other. This yields at total of $(n-1)+(m-1)+1 = n+m-1$ total intersections, as desired.

Solution 2: Note: The floor/ceiling math in this problem is quite messy. I have spelled out all the details, but I would be happy with any answer of the form $n/3 + c$, where c is some constant.

(a) We will show the stronger result that, given any polygon P and any triangulation of P, there exists a diagonal of this triangulation that subdivides the polygon into two parts, each with at least $\lceil n/3 \rceil + 1$ vertices.

It was shown in class that any triangulation of a simple polygon in the plane with n vertices has $n-2$ triangles. Let T denote the dual graph of the triangulation. Recall from class that T is a free tree (that is, a connected, acyclic undirected graph) with $n-2$ vertices and maximum degree three. Observe that each edge of the dual graph corresponds to an diagonal of the triangulation. The construction rests on the following claim.

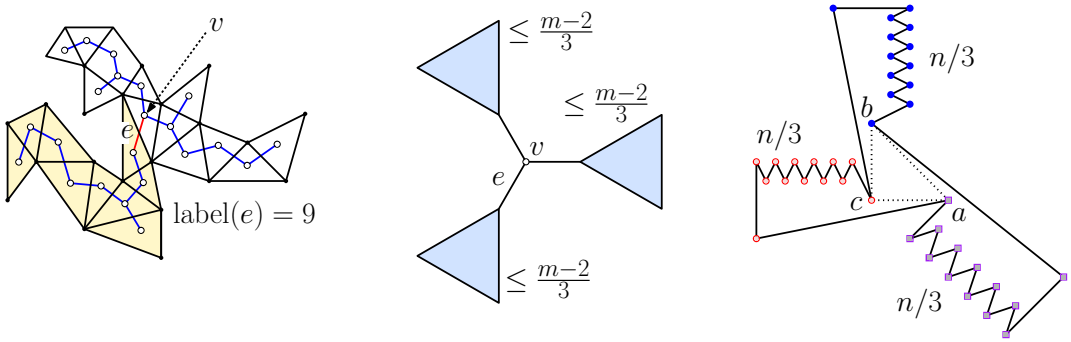
Lemma: In any free tree of degree at most three with $m \geq 2$ nodes, there exists an edge whose removal splits the tree into two subtrees, each with at least $\lceil \frac{m-1}{3} \rceil$ nodes.

Proof: Removing any edge from the tree splits the tree into two subtrees. Label each edge with the number of nodes in the smaller of these two subtrees. Let e be the edge with the maximum label (see Fig. 2(a)).

Clearly, it suffices to show that e's label is at least $\lceil \frac{m-1}{3} \rceil$. Suppose towards a contradiction that e's label is strictly less than $\lceil \frac{m-1}{3} \rceil$, which implies that it is at most $\frac{m-2}{3}$. Let v be the node on the "heavier" side of e. Since the tree has degree at most three, v is incident to at most two other edges. Also, since e has the largest label, each of v's other subtrees has at most $\frac{m-2}{3}$ nodes (see Fig. 2(b)). Therefore, the total number of nodes in the entire tree is equal to 1 (for v) plus the sum of the numbers of nodes in the (up to three) subtrees branching off from v. This is at most

$$1 + 3\frac{m-2}{3} \;=\; m - 1,$$

which contradicts the fact that the tree has m nodes.



(a) (b) (c)

Figure 2: Balanced diagonal.

Now, consider any triangulation of an n-vertex polygon. The dual tree has $m = n - 2$ nodes. By the above lemma, there is an edge whose removal splits the dual tree into subtrees each with at least

$$\left\lceil \frac{m-1}{3} \right\rceil \;=\; \left\lceil \frac{n-3}{3} \right\rceil \;=\; \left\lceil \frac{n}{3} \right\rceil - 1$$

nodes. Since each vertex in the dual tree represents a triangle, each of the two sub-triangulations has at least $t = \lceil \frac{n}{3} \rceil - 1$ triangles, or equivalently at least

$$t + 2 \;=\; \left\lceil \frac{n}{3} \right\rceil - 1 + 2 \;=\; \left\lceil \frac{n}{3} \right\rceil + 1$$

vertices, as desired.

(b) Consider the simple polygon sketched in Fig 2(c). The polygon consists three "lobes", joined by three vertices, a, b, and c. All three lobes contain a roughly equal number of vertices. In

particular, all the lobes contain at most $\lceil n/3 \rceil$ vertices. Observe that other than a, b, and c, no vertex in one lobe is visible to any vertex in another lobe.

Clearly, the most even split arises by adding one of the three diagonals $\overline{ab}$, $\overline{bc}$, or $\overline{ca}$. No matter which we add, one of the subpolygons that results will contain at most $\lceil n/3 \rceil + 1$ vertices. Given any parameter $\gamma > 1/3$, we can make n large enough such that $\lceil n/3 \rceil + 1 < \gamma n$. In particular, if we express $\gamma = \varepsilon + 1/3$, for some $\varepsilon > 0$, it is easy to verify that this will hold for all $n > 2/\varepsilon$. Setting n to this value in the above construction yields the desired counterexample.

Solution 3:

(a) Suppose to the contrary that a shortest rectilinear (s, t)-path fails to be x-monotone. This implies that there exists an x-coordinate, x_0, such that the path crosses the vertical line $x = x_0$ at least three times (see Fig. 3(a)). Let y_1 and y_2 be first and last points where path crosses this line, respectively. We can shorten the path by replacing the subpath from y_1 to y_2 with a single vertical segment. This contradicts the hypothesis that the original path is a shortest path.

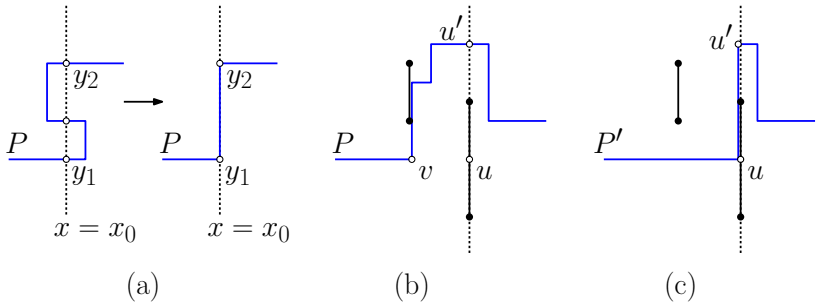


Figure 3: (a) Path monotonicity, (b) non-canonical path, and (c) (more) canonical replacement path P'.

(b) Suppose to the contrary that no shortest rectilinear (s, t)-path is in canonical form. Among all shortest paths, consider the path P where the earliest violation of canonical form is farthest to the right (that is, has the maximum x-coordinate). We will present a contradiction by showing that there is a path P' that is at least as short as P and it is either canonical, or its earliest violation has a strictly larger x-coordinate.

Let v denote the leftmost vertex on P that violates canonical form (see Fig. 3(b)). Since v violates canonical form, there is a point u that is horizontally strictly to the right of v, such that its x-coordinate coincides with the first obstacle segment that is strictly to the right of v or (if there is no such segment) the vertical line through t. Consider the vertical line through u. Path P must cross this line at some point u'. Construct a new path P' by replacing the portion of the path from v to u', with the two segments $\overline{vu}$ and $\overline{uu'}$ (see Fig. 3(c)). Because this is the next obstacle segment, this subpath is obstacle free, and its length is not longer than the length of P. This yields the desired contradiction.

(c) We will design a plane-sweep algorithm that computes all the relevant canonical paths in $O(n \log n)$ time. By monotonicity, we may ignore any obstacle segments to the left of s or to

the right of t. Let's assume that the obstacle segments are sorted by increasing order of their x-coordinates.

The sweep-line status for a given vertical sweep line ℓ contains horizontal components of all possible minimum-length canonical paths starting at s and hitting the sweep line, sorted from bottom to top (see Fig. 4(a)). Whenever a canonical path hits an obstacle segment, it may split into two potential canonical paths, one that zigs up through a_i and the other that zigs down through b_i. (We cannot predict in advance which will lead to the final shortest path, so we will maintain both.) Starting with the original canonical path out of s, each obstacle vertex can generate two canonical paths, one for each of its endpoints. Thus, the size of the sweep-line status is $O(n)$ throughout the algorithm.

By monotonicity, the horizontal lengths traversed by the canonical path up to the current sweep line are all the same. So, it suffices to track their total vertical lengths. For the jth canonical path on the sweep line, let y_j denote its y-coordinate, and let v_j denote the sum of its vertical pieces. There are three events that we process:

Initial: The algorithm starts by sorting all the segments. It starts the sweep line at the vertical line through s. It generates a single path whose y-coordinate is s_y and whose vertical distance is zero (see Fig. 4(b)). This takes $O(n \log n)$ time to sort the segments.

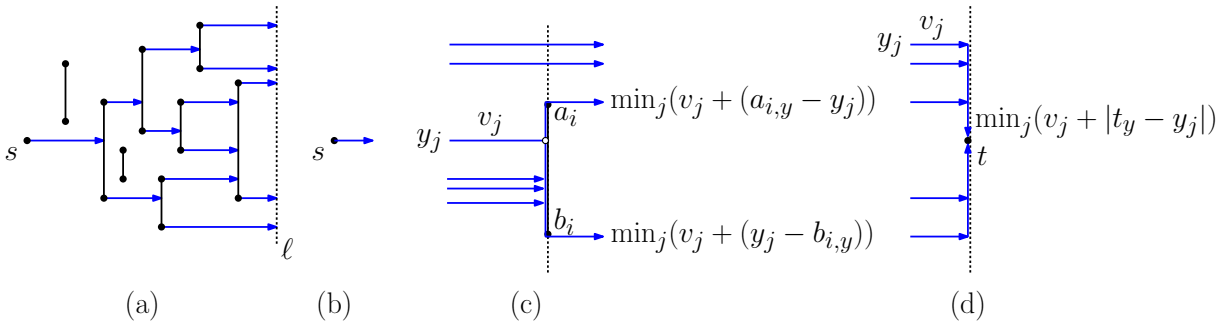


Figure 4: Plane-sweep events.

Segment: To process segment s_i, we locate where the y-coordinates of its endpoints a_i and b_i fall within the current sweep-line status. We insert two new elements into the sweep line at the y-coordinates of a_i and b_i. We repeatedly access successors to find all the sweep-line paths that lie between a_i and b_i. For each, let y_j denote its y-coordinate and let v_j denote its total vertical distance. We remove this path from the sweep-line status and consider two new candidate paths: one that emanates from a_i and has vertical distance $v_j + (a_{i,y} - y_j)$, and the other that emanates from b_i and has vertical distance $v_j + (y_j - b_{i,y})$. Among all the candidate paths emanating from a_i, we keep only the one with the smallest total vertical distance. We do the same for b_i (see Fig. 4(c)).

The time for this operation is equal to $O(\log n)$ to locate a_i and b_i in the sweep-line status. We also spend $O(\log n)$ time for each of the paths that we remove. But since each path emanating from a segment endpoint can be removed at most once, the total time for all these removals is $O(n \log n)$.

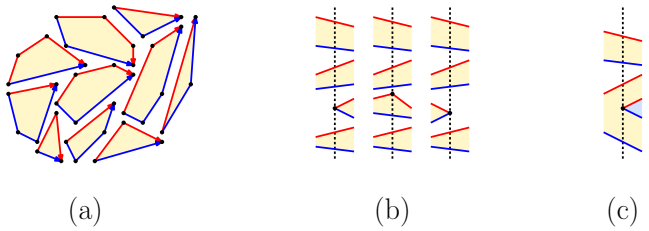
Final: When the sweep-line reaches the vertical line through t, we enumerate all the paths in the sweep-line status. For each path, with y-coordinate y_j and total vertical distance v_j, we compute its final total distance as $v_j + (a_{i,y} - y_j)$. Among all paths, we take the one having the smallest such value as the final path (see Fig. 4(d)). The time for this step is proportion to the number of surviving paths, which is $O(n)$. This only considers the vertical lengths, but the total length is given by adding this to $t_x - s_x$.

The correctness of the algorithm follows from (a) and (b) and the fact that the algorithm maintains all the canonical paths that could be part of the shortest path. The total running time as argued above is $O(n \log n)$.

Solution 4: We are given m convex polygons with a total of n vertices, and we want to determine whether any two intersect.

(a) Let's first consider the easier case of boundary intersections. We can reduce this to line segment intersection, but with a bit more cleverness to keep the running time low in case there are intersections.

- In $O(n)$ time, we can split each convex polygon into two x-monotone chains, and upper chain and a lower chain, each sorted from left to right (see Fig. 5(a)).



(a) (b) (c)

Figure 5: Convex polygon intersection.

- The sweep-line status holds the upper and lower segments of the polygons that intersect the sweep line, sorted from bottom to top. Since there are m polygons, there are at most $2m = O(m)$ segments in the sweep-line status at any time. Therefore, all the sweep-line dictionary operations can be performed in $O(\log m)$ time.
- Rather than sorting all the vertices by their x-coordinates, in $O(m \log m)$ time we sort only the leftmost vertices of each polygon. We maintain a dynamic priority queue, but rather than storing all n vertices, we will just store the right endpoint of each of the $O(m)$ segments in the sweep-line status, along with the leftmost vertices of the convex polygons that lie to the right of the sweep line. Thus, there are at most $O(m)$ entries in the priority queue at any time, implying that priority queue operations can be performed in $O(\log m)$ time.
- There are three types of events: The left endpoint of a convex polygon, a middle segment endpoint of an upper or lower chain, and the right endpoint of a convex polygon (see Fig. 5(b)). All can be processed in $O(\log m)$ time, since there are only $O(m)$ entries in the sweep-line. (Details are omitted.)

- We process segment intersections in exactly the same way as the line-segment intersection algorithm given in class, but we *terminate the algorithm as soon as the first intersection is detected*. Since we stop as soon as we detect an intersection, the number of events processed is $O(n)$, irrespective of the actual number of intersections.

In summary, there are $O(n)$ events, and each can be processed in $O(\log m)$ time, which implies a total running time of $O(n \log m)$.

(b) Containment intersections can be handled through a very minor adjustment to the above algorithm. Whenever we encounter the left endpoint of a convex polygon, we locate its y-coordinate in the sweep-line status. We check the edge lying immediately above and below. If they both belong to the same polygon, we declare that we have discovered a containment intersection and terminate (see Fig. 5(c)).

Solution to the Challenge Problem: Define a *slope-restricted polygon* in $\mathbb{R}^2$ to be one in which the sides of the polygon are restricted to being vertical, horizontal, or having slopes $+1$ or -1. Given a set $P = \{p_1, \ldots, p_n\}$ of points in $\mathbb{R}^2$ an *SR-enclosure* is a slope-restricted polygon that contains all the points of P. In this problem, we will present an algorithm for computing an SR-enclosure of minimum perimeter. Note that there may be many SR-enclosures that have the same minimum perimeter, and among these, we may return any one.

Given any nonzero vector u, a *u-extreme point* of P is any point $p \in P$ that maximizes the dot product $p \cdot u$. Equivalently, if we were to project all the points of P onto an infinite ray in the direction u, the farthest point along this ray is a u-extreme point (see Fig. 6(a)). Given P consider the eight vectors with components drawn from $\{-1, 0, +1\}$, that is, $\{(\pm 1, 0), (0, \pm 1), (\pm 1, \pm 1), (\pm 1, \mp 1)\}$. Let $P' \subseteq P$ be the subset of size at most eight consisting of the points that are extreme along each of these directions. (There may be fewer than eight points, because a single point of P may be extreme for multiple directions.) Let E be the enclosure that results by intersecting the halfplane passing through each of these extreme points that is orthogonal to the directional vector defining this extreme point (see Fig. 6(b)).

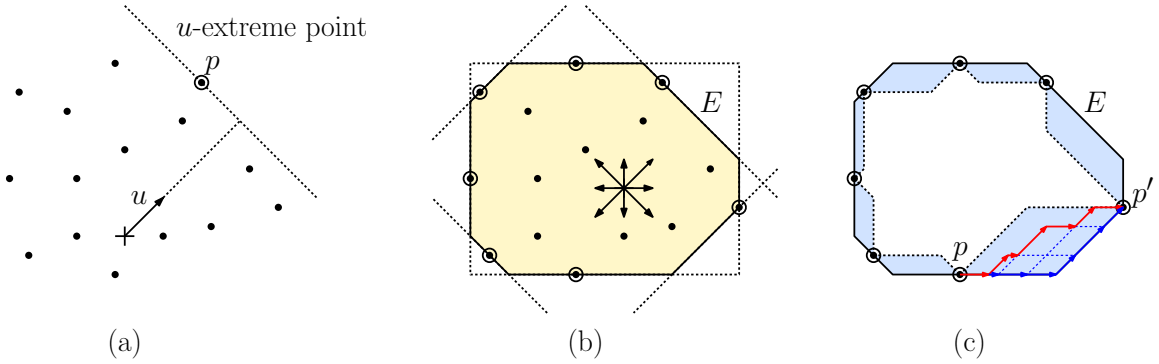


Figure 6: Minimum perimeter enclosure.

Observe that we can compute E in $O(n)$ time. The points of P' can each be computed in $O(n)$ time, and the intersection of halfplanes can be computed in $O(1)$ time as shown in class.

We assert that E is a minimum perimeter SR-enclosure for P. (This is just a sketch.) Consider the points of P' in cyclic order about E's boundary. For each consecutive pair (p, p') consider the

parallelogram with p and p' as vertices whose sides are parallel to the two edges between p and p' (shaded in blue in Fig. 6(c)). We assert that the shortest SR-path from p to p' uses only edges parallel to these two edges. (I don't have a proof of this, at least not a simple one. If you know of one, please let me know.) Consider any path from p and p' consisting of segments parallel to these edges (shown in red in Fig. 6(c)). Observe that by translating each of the edges of this path to the boundary of E, we obtain a new path (shown in blue in Fig. 6(c)) that travels from p to p' along E's boundary. Since any SR-enclosure of P must include the points of P', it follows that no SR-enclosure of P can have a smaller perimeter than E.