Homework Exam 1 2024-2025

My name and StudentID go here!

Deadline: 22 November 2024, 09:00

This homework exam has 1 question for a total of 9 points. You can earn an additional point by a careful preparation of your hand-in: using a good layout, good spelling, good figures, no sloppy notation, no statements like "The algorithm runs in $n \log n$." (forgetting the $O(..)$ and forgetting to say that it concerns time), etc. Use lemmas, theorems, and figures where appropriate.

Question 1

Let P be a set of n points in $\mathbb{R}^2$, and let $\mathcal{R}$ be a set of m, possibly pairwise intersecting, polygonal regions. You can assume by general position that all points and vertices have unique coordinates, and that no three points and vertices are colinear. The depth $d_{\mathcal{R}}(p)$ of a point p with respect to $\mathcal{R}$ is the number of regions from $\mathcal{R}$ that contain it.

1. *(6 points)* Consider the scenario in which all regions in $\mathcal{R}$ are axis-aligned rectangles. Design an $O((n+m)\log(n+m))$ time algorithm that computes, for each point in P, its depth with respect to $\mathcal{R}$. Prove that your algorithm is correct and achieves the desired running time.
2. *(2 points)* An equilateral triangle is axis aligned when one of its sides is axis aligned. Extend your algorithm from question (a) to the case where all regions in $\mathcal{R}$ are axis aligned equilateral triangles. Briefly argue that your algorithm is correct and analyze its running time.

   (Note that you do not have to repeat the entire description from question (a). Focus on what is different.)
3. *(1 points)* Can your algorithm (from question (a) and/or (b)) still compute the depths in $O((n+m)\log(n+m))$ time in case the regions in $\mathcal{R}$ are arbitrary triangles? Briefly argue why/why not.

a. An algorithm for rectangular regions

Let R be an y-monotone simple polygon: a simple (polygonal) region in which any horizontal line intersects R in a single interval. The union of all left endpoints of these intervals defines the *left boundary* of R, and the union of all right endpoints defines the *right boundary* of R.

Observation 1. *A point $q \in \mathbb{R}^2$ lies inside a y-monotone polygon R if and only if the horizontal leftward ray $\overleftarrow{q}$ starting in q intersects the left boundary of R but not the right boundary of R.*

Let $\mathcal{P}$ be a set of y-monotone regions, and let $\ell_{\mathcal{P}}(q)$ be the number of regions from $\mathcal{P}$ for which $\overleftarrow{q}$ intersects the left boundary. Symmetrically, let $r_{\mathcal{P}}(q)$ be the number of such regions for which $\overleftarrow{q}$ intersects the right boundary.

Lemma 2. *Let q be a point in $\mathbb{R}^2$, and let $\mathcal{P}$ be a set of y-monotone polygons. We have that $d_{\mathcal{P}}(q) = \ell_{\mathcal{P}}(q) - r_{\mathcal{P}}(q)$.*

Proof. Let $R \in \mathcal{P}$ be some region. If $q \in R$ then by Observation 1 $\overleftarrow{q}$ intersects the left boundary of R but not the right boundary. Hence R correctly contributes one to the count of $\ell_{\mathcal{P}}(q)$ and zero to the count of $r_{\mathcal{P}}(q)$. If $q \notin R$ then either: (i) the horizontal line h through q does not intersect R at all, (ii) q lies left of the interval $h \cap R$, or (iii) q lies right of $h \cap R$. In the first two cases R clearly contributes zero to both $\ell_{\mathcal{P}}(q)$ and $r_{\mathcal{P}}(q)$, as desired. In the last case R contributes one to both $\ell_{\mathcal{P}}(q)$ and $r_{\mathcal{P}}(q)$, and thus the contribution to $d_{\mathcal{P}}(q)$ is zero as well. □

Observe that rectangles are y-monotone polygons. So, we can use Lemma 2 to express the depth of a point q in terms of $\ell_{\mathcal{R}}(q)$ and $r_{\mathcal{R}}(q)$.

The main idea is now to use a sweep-line algorithm to compute the $\ell_{\mathcal{R}}(q)$, with $q \in P$, values. We compute the $r_{\mathcal{R}}(q)$ values analogously. We sweep a horizontal line downwards while maintaining the set of left boundary edges of the regions in $\mathcal{R}$ currently intersecting the sweep line. We maintain as invariant that for all points above the sweep line we correctly computed the depth.

Observation 3. *The order in which the sweepline h intersects the left boundaries of the regions in $\mathcal{R}$ changes only when h sweeps over an endpoint of a left-boundary edge (a corner) of region in $\mathcal{R}$.*

The events are the endpoints of the (left) boundary edges in $\mathcal{R}$, and the points in P, sorted on decreasing y-coordinate. Since we know all such events before the sweep we can simply use a sorted list as an event queue.

As a status structure we use a balanced binary search tree storing the left edges intersecting the sweep line in left to right order. We augment the tree with subtree counts such that we can efficiently report the number of elements in the tree smaller than some query value q. This data structure takes linear space and answers queries in $O(\log m)$ time. Updates take $O(\log m)$ time.

At an endpoint of a left boundary edge; either insert into or delete the edge from the status structure.

At a query point $q \in P$: query the status structure to compute $\ell_{\mathcal{R}}(q)$: the number of left boundary segments intersecting the sweep line left of q.

Handling every event takes $O(\log m)$ time. There are $O(n+m)$ events, and sorting them in the initialization phase takes $O((n+m)\log(n+m))$ time. Hence:

Theorem 4. *Let P be a set of n points in $\mathbb{R}^2$, and let $\mathcal{R}$ be a set of m axis aligned rectangles. We can compute the depth $d_{\mathcal{R}}(q)$ of each point $q \in P$ in $O((n+m)\log(n+m))$ time.*

b. Extending to equilateral triangles

There are four types of axis aligned equilateral triangles (two types where the side is horizontal, two where the side is vertical). We compute the depth of each type separately and sum the results. Consider type where the triangles have a horizontal side and pointing downwards (the other cases are symmetric). In this case, we again have that the order in which the left (right) sides intersect the sweep line changes only when we sweep over an endpoint. Therefore, we can apply the algorithm from Theorem 4 to compute the $\ell_{\mathcal{R}}(q)$ and $r_{\mathcal{R}}(q)$ values, and thus the depths (w.r.t the horizontal downward triangles) in $O((n+m)\log(n+m))$ time. By running this algorithm four times we compute the depth with respect to all triangles in $\mathcal{R}$.

c. Extending to arbitrary triangles

No. The sweepline algorithm needs that the order in which the left boundary edges intersect the sweep line remains fixed in between events (since we need a way to easily count the number of edges left of a query point). This order changes at intersections of the left boundary edges, and there may be $\Theta(n^2)$ such intersections. Unfortunately, we also can no longer partition $\mathcal{R}$ into disjoint sets whose left boundaries do not pairwise intersect (as in b). Hence, the algorithm sketched above needs to handle intersections explicitly and there are too many of them to handle in the stated time.

Note that strictly speaking we only need the number of left edges before some query point q, not there ordering. So, we could try to maintain a status structure that does not explicitly maintain this order. Unfortunately, there is no data structure (of subquadratic size) that supports such queries in $O(\log(n+m))$ time. (And there is a lowerbound that states such a data structure cannot exist)