Solution to: Homework Exam 1 2019

Deadline: 22 November 2019, 12:45

This homework exam has 1 questions 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 *(9 points)*

Let S be a set of n disjoint line segments, and let $\mathcal{D} = [x_{\min}, x_{\max}] \times [y_{\min}, y_{\max}]$ be an axis parallel rectangle. You can assume that no two endpoints have the same x-coordinate or the same y-coordinate. Develop an $O(n \log n)$ time algorithm to find a longest vertical line segment $\overline{ab} \subseteq \mathcal{D}$ whose interior intersects the interior of at most one segment in S. Prove that your algorithm is correct and achieves the desired running time.

Solution

We say that a segment is *valid* if it is vertical, contained in $\mathcal{D}$, and intersects at most one segment in S.

Lemma 1. *There is an optimal valid segment $v^* = \overline{ab}$ such that: (i) a and b lie on segments in S or on horizontal edges of $\mathcal{D}$, and (ii) either (ii.a) v^* passes through an endpoint of a segment in S, (ii.b) v^* passes through an intersection point of a segment in S with the horizontal boundary of $\mathcal{D}$, or (ii.c) is a subsegment of the left or right boundary of $\mathcal{D}$.*

Proof. Claim (i) in the lemma statement is trivial. Next, we prove claim (ii) by contradiction. Assume that v^* is optimal, and no optimal vertical segment with properties (i) and (ii) exists.

Let s_a and s_b be the line segments (either those S or those representing the boundary of $\mathcal{D}$) containing endpoints a and b, respectively, and let s_c be the segment intersected by the interior of v^* (if it exists). Furthermore, let $v(x)$, be the segment at x-coordinate x that has its endpoints on a and b.

Consider shifting v^* to the left, while keeping its endpoints on s_a and s_b until we: (1) encounter the left endpoint of s_a, s_b, or s_c, (2) we start to intersect an additional segment s_d, (3) the endpoints a and b exit $\mathcal{D}$, or (4) we are at the left boundary of $\mathcal{D}$. Let $v(x_\ell)$ at x_ℓ be the segment after shifting, and observe that $v(x_\ell)$ is valid (since the only segment it can intersect in its interior is still s_c), and satisfies conditions (i) and (ii). Analogously, we obtain a valid segment v_r at x_r that satisfies (i) and (ii) by shifting to the right.

Let $f(x)$ be the length of $v(x)$. Since f is a linear function, we have that $\max\{f(x_\ell), f(x_r)\} \geq f(a_x)$. That is, either $v(x_\ell)$ or $v(x_r)$ is at least as long as v^*, is valid, and satisfies (i) and (ii). Contradiction. The lemma follows. □

Observation 2. *Let $v^* = \overline{ab}$ be a valid segment that contains point p, then $\overline{pa}$ and $\overline{bb}$ intersect at most one segment in S.*

It follows from Lemma 1 and Observation 2 that the only segments we have to consider are maximal vertical segments passing through an endpoint of a segment in S, or the intersection points of S with the ~~vertical~~ edges of $\mathcal{D}$. Moreover, for every such candidate point p, there are only a constant number of valid maximal segments that contain p. In particular, the upper endpoint of such a segment lies either on the first or the second segment hit by a vertical ray starting in p. The same holds for the lower endpoint.

We will now develop a sweep line algorithm that sweeps a vertical line over $\mathcal{D}$ to compute all these candidate segments and their lengths.

The status structure stores an optimal valid segment left of the sweep line, and a balanced binary search tree (e.g. a red black tree) that stores the segments in S intersected by the sweep line. These segments are ordered from bottom to top.

The status changes when we enter or exit $\mathcal{D}$ and when we sweep over an endpoint of a segment. We can precompute all these events, so the event queue is simply a linked list with all these points, ordered by increasing x-coordinate.

We start the sweep at x-coordinate $-\infty$, and initialize the status structure with an empty BST and a non-existent segment of length $-\infty$.

At each event point p, we query the status structure to find the first two segments above and below p. If p lies in $\mathcal{D}$ this gives us $O(1)$ candidate segments $\overline{ab}$ and their lengths. For each candidate, we test if it is valid, compare it with the maximum length segment found so far, and update the maximum length segment found so far if needed. We then either insert or remove the segment s incident to p (depending on whether p is the left or right endpoint of s).

Since there are $2n$ endpoints, and $2n$ intersection points, constructing the event queue takes $O(n \log n)$ time. We handle each event by $O(1)$ queries, a single update in the status structure, and a constant amount of additional work. Since the operations on the status structure take $O(\log n)$ time each we can handle an event in $O(\log n)$ time. We conclude:

Theorem 3. *Given a set S of n line segments in $\mathcal{D}$, a longest vertical line segment in $\mathcal{D}$ that intersects at most one segment of S in its interior can be computed in $O(n \log n)$ time.*