CSCI 534
Homework 03
Due 3-9-2021
David L. Millman

Problem 1 (10 points)

Given an polygonal chain P of n vertices, we define an vertex v as a *local max* if v if all edges adjacent to v are to the left of v. Show that can determine if a polygonal chain with k local maxes is simple in $O(n \log k)$ time.

Problem 2 (10 points)

A friend of yours from the civil engineering department wants to analyze whether a dangerous portion of a river will flood. He presents you with the following (admittedly rather unrealistic) model of the river. The portion of the river of interest is modeled as an x-monotone polygon P that is bounded between two vertical lines at $x = x^-$ and $x = x^+$ (see Figure). The river is bounded on its left and right ends by two vertical line segments of lengths w^- and w^+, respectively. Inside the polygon are some number of disjoint x-monotone polygons that represent islands in the river. Let n denote the total number of vertices, including both the outer banks of the river and the islands.

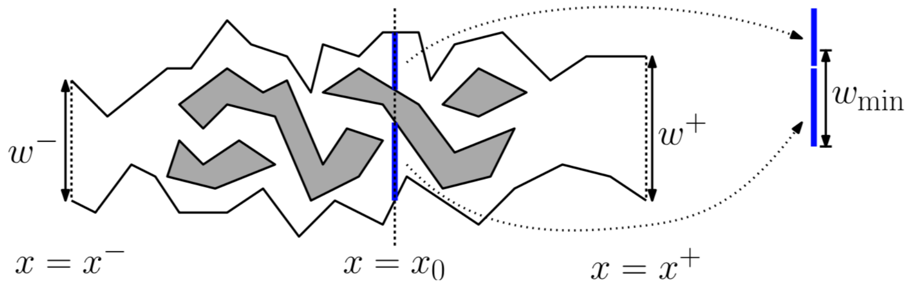

Figure 1: Problem 2: River

Your friend tells you that the in order to avoid a flood, the width of the river (not counting islands) at every vertical cut must be at least some minimum value w_{min}. For example, in the figure, the sum of the two blue vertical segments at $x = x_0$ must be at least w_{min} in order to avoid a flood. Given the polygon P and the value w_{min}, present an $O(n \log n)$ time algorithm that determines whether the river will flood, that is, whether there is a vertical cut whose total width is smaller than w_{min}. If it will flood, your algorithm should output the value x_0 of the bottleneck, that is, the location where the sum of vertical lengths (excluding islands) is the smallest.

1. Hint 1: There is an (uncountably) infinite number of possible vertical cuts to consider. Prove that it suffices to check the width at a discrete set of locations, whose number is $O(n)$.

2. Hint 2: There is a bit of a trick to updating the vertical widths (excluding the islands). For partial credit, explain how to do it under the assumption that the sweep line can only intersect a constant number of islands at any time. (For example, in the figure, the sweep line never hits more than two islands at a time.) For full credit, explain how to do it even if the number of islands hit by the sweep line at any time could be as high as $\Omega(n)$.

Problem 3 (10 points)

1. (7 points) Describe and analyze an algorithm that computes the convex hull of a set of n points in the plane using randomized incremental construction in expected $O(n \log n)$ time. For this problem you are welcome to find an algorithm and its analysis on the web, but please cite where you found it, describe it concisely in your own words, and make the analysis very concise. Where does the log-factor come from?

2. (3 points) Give an example of a set of points in the plane, and a particular input order, that causes the convex hull algorithm to run in $O(n^2)$ when the points are added in this particular order. Make sure it is clear how your example generalizes to arbitrary values of n.

Problem 4 (5 points)

Consider the following instance of the trapezoidal map point location data structure. The left side shows the map, and the right side shows the corresponding DAG. Describe the resulting trapezoidal map and DAG after segment xy has been added.

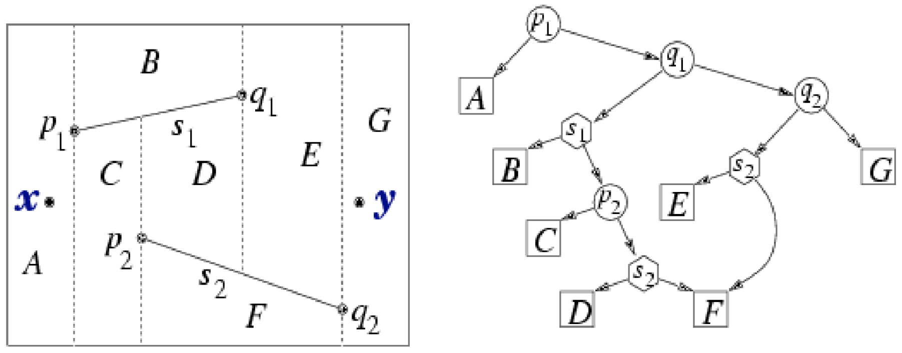

Figure 2: Problem 4: Trapezoid Map

Problem 5 (10 point)

Consider the following algorithm:

```
FindMax(A,n){
    // Finds maximum in set A of n numbers
    if(n==1) return the single number in A
```

```
        else {
            x = extract random element from A // in constant time; x is removed from
            A
            y = FindMax(A,n-1)
            if(x<=y) return y;
            else
            Compare x with all remaining elements in A and return the maximum
        }
    }
```

1. (4 points) Argue that this algorithm is correct, and give its worst-case runtime. (The runtime is proportional to the number of comparisons made.)
2. (6 points) Compute the expected runtime of this algorithm. (Hint: Introduce an indicator random variable for executing the else branch in the i-th step, and use backwards analysis to simplify the analysis.)

Tips and Acknowledgements

Acknowledgements: Homework problems adapted from assignments of David Mount and Carola Wenk.