Assignment 8

CSCI 3110: Design and Analysis of Algorithms

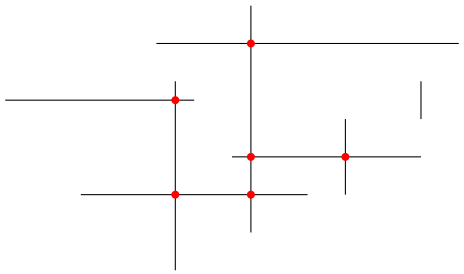
Due July 17, 2018

Banner ID: _____ Name: _____

Banner ID: _____ Name: _____

Banner ID: _____ Name: _____

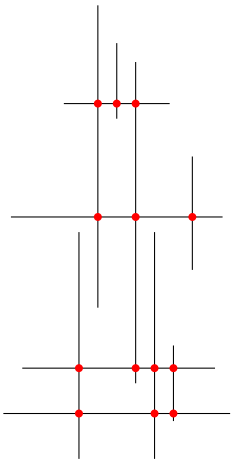
Consider the following problem: You are given a set of n line segments. Each segment is either vertical or horizontal. You may assume that no two segment endpoints have the same x- or y-coordinate unless they are endpoints of the same segment (same y-coordinate for horizontal segments, same x-coordinate for vertical segments). Your goal is to find all intersections between these n segments. For example, in the following example, you want to output the list of all red points:



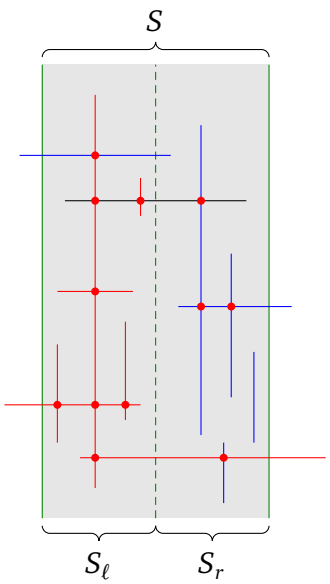
This problem can easily be solved in $O(n^2)$ time by checking for every pair of segments whether they intersect. There can be up to $n^2/4$ intersections, so just outputting all intersections will take $\Omega(n^2)$ time in the worst case. However, if there are no intersections, there is no reason to believe that it really takes $O(n^2)$ time to verify this fact. It *can* be shown that verifying that there are no intersections requires $\Omega(n \lg n)$ time. Thus, a running time of $\Omega(n \lg n + t)$ is the best we can do if there are t intersections to be reported. Can you come up with an algorithm that matches this lower bound, that is, one with running time $O(n \lg n + t)$?

In class, we will discuss how to use the distribution sweeping paradigm and an augmented (a, b)-tree (or in fact any balanced search tree) to obtain an algorithm for this problem with the desired running time. Here, you should use divide and conquer to obtain an algorithm with running time $O(n \lg n + t)$.

(a) Let's start small: Assume the given segments have the property that every horizontal segment h spans the x-coordinate of each vertical segment v, that is, h's left endpoint is to the left of the vertical line through v and h's right endpoint is to the right of the vertical line through v. Under this assumption, the x-coordinates of segment endpoints are irrelevant for deciding whether two segments intersect. Assume further that the segments are given in two lists H and V containing the horizontal and vertical segments, respectively; the segments in H are sorted by their y-coordinates, the segments in V are sorted by the y-coordinates of their bottom endpoints. Under these assumptions, describe a simple algorithm that finds all t intersections between horizontal and vertical segments in $O(n+t)$ time. This algorithm should not use divide and conquer at all.



(b) Now consider a vertical slab $S = [x_\ell, x_r] \times (-\infty, +\infty)$, that is, the set of all points with x-coordinates between x_ℓ and x_r. Assume you are once again given two lists H and V containing horizontal and vertical segments, respectively, and assume they are sorted as in part (a). However, this time the segments in H and V satisfy a weaker property: each segment has at least one endpoint in S. If the segments in H and V have m endpoints in total in S, S can be split into two smaller slabs S_ℓ and S_r containing $m/2$ of the endpoints each and we can define lists H_ℓ and V_ℓ (H_r and V_r) containing all horizontal and vertical segments with at least one endpoint in S_ℓ (S_r). Provide a divide-and-conquer algorithm for finding all t intersections between segments in H and V in $O(n \lg n + t)$ time. To do so, you should split the intersections into two sets: those you can find by recursing on S_ℓ and S_r and those you can't. Can you use your solution to part (a) to find the latter group of intersections?



In this example, the red and black segments are added to list V_ℓ or H_ℓ and the blue and black segments are added to list V_r or H_r.

(c) Fill in the missing details:

- How do you choose the initial vertical slab S so it contains all segment endpoints?
- How do you produce the lists H and V for the first recursive call?
- If you have not done so in part (b) yet, discuss how you produce the input lists H_ℓ, V_ℓ, H_r, and V_r for the recursive calls you make in linear time.
- Argue briefly that your algorithm achieves the desired running time of $O(n \lg n + t)$ and that it finds all intersections.