Chapter 2

Sweeping: Segment Intersection

By Sariel Har-Peled, January 26, 2023[1]

> The Party told you to reject the evidence of your eyes and ears. It was their final, most essential command.
> – 1984, George Orwell.

Version: 0.1

Disclaimer: These are sketchy sketch class notes made for me to prepare for lecture. They do not in any way include all details covered in lecture. For full details, see the bibliographical notes in the end.

2.1. Sweeping

The input is a set $S = \{s_1, \ldots, s_n\}$ of n segments in the plane. As a reminder, a segment is a finite connected portion of a line. Formally, given two points p, q, their ***segment*** is $s = pq = \mathcal{CH}(\{p, q\})$. The points p and q are the ***endpoints*** of s.

Given S, we would like to compute all the intersection points between any pair of segments $s_i, s_j \in S$. For the time being, let us assume general position – all intersections and endpoints have unique coordinates. No such point lies in the interior of another segment, and no two segments are colinear (i.e., the two segments are contained by the same line).

Naive, but not horrible algorithm. The most natural algorithm is to compute the intersection point of each pair of segments of S. Since computing the intersection of two segments is a constant size problem, this can readily be done in $O(1)$ time per pair. Overall, the running time is $O(n^2)$.

But what if... the number of intersection points, denoted by k, is relatively small? In particular, of special interest is the question of deciding if the segments of S are disjoint.

To do better, we need to understand better the "structure" of the plane in the presence of this segments.

2.1.1. Towards a better algorithm – vertical and horizontal segments.

Let assume all the segments are horizontal or vertical. For every vertical segment s, we want to know how the horizontal segments intersect the vertical line ℓ supporting s. Consider the configuration in

Figure ?? – computing the intersections is pretty easy – sort the horizontal segments along ℓ (specifically, we store the intersection points with the line ℓ in a balanced binary search tree along the y-axis), and then we an interval query for the vertical segment s.

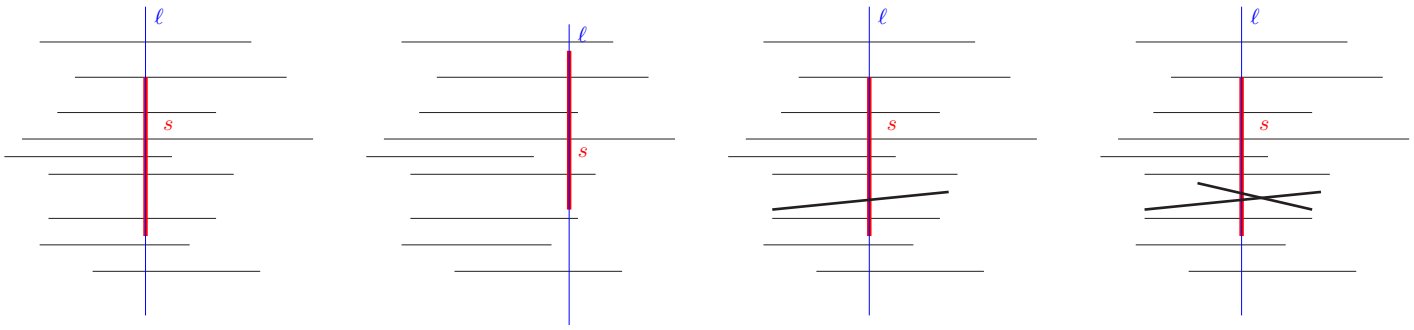

Figure 2.1

To fill in some future:

(A) Extend to two locations.
(B) x-structure $\implies$ queue.
(C) What if segments are not horizontal?
(D) What if segments intersects? Scheduling future events.

2.2. Bibliographical notes

This follows section 2.1 in [BCKO08], except that they sweep by a horizontal line instead of vertical.

References

[BCKO08] M. de Berg, O. Cheong, M. J. van Kreveld, and M. H. Overmars. *Computational Geometry: Algorithms and Applications*. 3rd. Santa Clara, CA, USA: Springer, 2008.