Point Location

Agenda

We will solve various problems from the Point Location chapter in the course book.

Trapezoidal map: warm up

Let *S* be a set of non-intersecting segments, and *T(S)* be its trapezoidal map.

Let *s* be a new segment not crossing any of the segments in *S*.

Prove that a trapezoid *t* in *T(S)* is also a trapezoid in *T(SUs)* iff *s* does not intersect *t*.

Trapezoidal map: warm up

- If *s* does not intersect *t* then *t* belongs to *T(SUs)*.
- The algorithm we have seen to compute a trapezoidal map add the segments one by one.
- If we will add *s* as the last segment then it clearly won't affect *t* (which is already part of the map) QED.

Trapezoidal map: warm up

- If *t* belongs to *T(SUs)* then *s* does not intersect *t*.
- Assume by contradiction that *s* intersects *t,* then clearly *t* is splitted.

Trapezoidal map computation

- Q6.10: Design a deterministic algorithm to compute the trapezoidal map of a set of segments.
- No need to compute the search DAG, just the subdevision (as a DCEL for example).
- Ideas?

Trapezoidal map computation

- Solution: a plane sweep algorithm.
- Order: From left to right.
- Status: The set of segments the sweep line cross.
- Event handle:
  - Start and end: Find the segment above and below, and add vertical lines to them, split the needed segments.
  - Intersection: swap the intersecting lines.

Segment intersection

- Given a set of segments, design an algorithm that computes all the intersections of pairs of segments.
- Complexity O(nlogn+k) on average.
- Ideas?

Segment intersection

- In the trapezoidal map computation we haven’t handled intersecting segments, however it is not problematic.
- Instead of following the segment only to the right, we will follow the segment even if it crosses to top or bottom segments, and update the map accordingly.
- Each intersection will be handled at some point, thus we can easily report all the intersection as a byproduct.

Segment intersection

- What is the average size of a trapezoidal map of n segments with k intersections?
- We can think as any intersection as 4 non-intersecting segments, thus the size is O(n+k), and thus each point location and DAG update will take O(log(n+k)) = O(log(n))
- In addition to point locating, we will handle k intersections, thus, in total the complexity will be O(nlog(n) +k) on average.