

Line Segment Intersection

Input: Set S of line segments.

Output: Report all intersection points of 2 or more segments.

Brute force algorithm:

1. **for each** $s, t \in S$, **where** $s \neq t$, **do**
2. **if** $s \cap t \neq \emptyset$, **then**
3. **report** $(s \cap t)$;

Line Segment Intersection : Plane Sweep 1

Q = queue of endpoints;

T = set of segments intersecting the sweepline;

1. $Q \leftarrow$ queue of endpoints;
2. $T \leftarrow \emptyset$;
3. Sort Q by decreasing y-coordinate;
4. **while** Q is not empty **do**
5. $p \leftarrow Q.$ Dequeue();
6. **for each** segment s with upper endpoint p **do**
7. **for each** $t \in T$ **do**
8. **if** $s \cap t \neq \emptyset$ **then report** $(s \cap t)$;
9. $T \leftarrow T \cup \{s\}$;
10. **for each** segment s with lower endpoint p **do**
11. $T \leftarrow T - \{s\}$;

Line Segment Intersection: Data Structure Operations

$T.$ Left(p) = line segment in T which is left neighbor of p .

$T.$ Right(p) = line segment in T which is right neighbor of p .

(Requires: every segment in T is intersected by horizontal line through p .)

$T.$ Contain(p) = segments in T whose interiors contain p .

$U_S(p)$ = segments in S with upper endpoint p .

$L_S(p)$ = segments in S with lower endpoint p .

Line Segment Intersection: Plane Sweep 2

1. Add all endpoints to Q ;
2. $T \leftarrow \emptyset$;
3. **while** Q is not empty **do**
4. $p \leftarrow Q.$ Remove_First();
5. **if** $|U_S(p) \cup L_S(p) \cup T.$ Contain(p) $| > 1$, **then**
6. Report(p);
7. Delete segments in $L_S(p)$ from T ;
8. Reorder segments in $T.$ Contain(p) in T ;
9. Insert segments in $U_S(p)$ in T ;
10. **if** $U_S(p) \cup T.$ Contain(p) = \emptyset **then**
11. FindNewEvent($T.$ Left(p), $T.$ Right(p), p);
12. **else**
13. $s \leftarrow$ Leftmost segment of $U_S(p) \cup T.$ Contain(p);
14. FindNewEvent($T.$ Left(p), s , p);
15. $s \leftarrow$ Rightmost segment of $U_S(p) \cup T.$ Contain(p);
16. FindNewEvent(s , $T.$ Right(p), p);

FindNewEvent

FindNewEvent(s, t, p)

1. $q \leftarrow s \cap t$;
2. **if** ($q \neq \emptyset$) **then**
3. **if** ($q.y < p.y$ **or** ($q.y = p.y$ and $q.x > p.x$)) **then**
4. **if** $q \notin Q$ **then**
5. $Q.Add(q)$;

Map Overlay

Input: Straight-edge planar embeddings S_1 and S_2 ;

Output: Map overlay of S_1 and S_2 ;

Map_Overlay(S_1, S_2)

1. $D \leftarrow \text{Edge_List}(S_1) \cup \text{Edge_List}(S_2)$;
2. Compute all intersection points of line segments in D ;
3. **at each** intersection point p **do**
4. $D.Update(p, S(p))$;
5. /* $S(p)$ = line segments containing p */