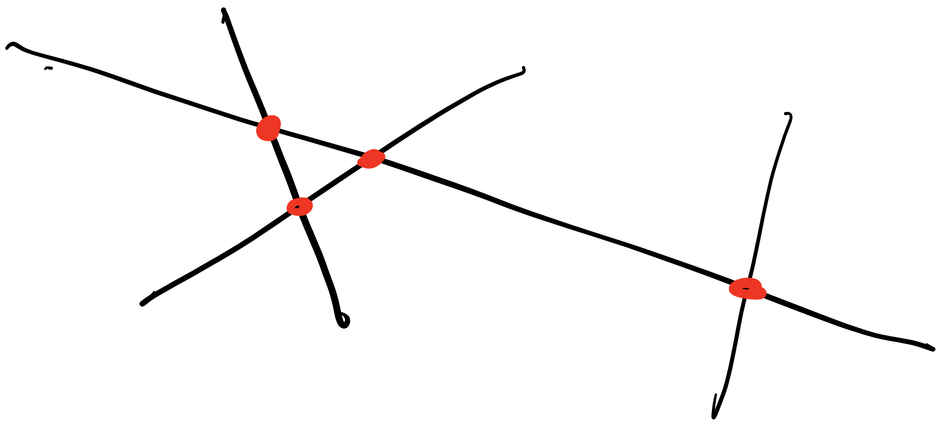
Line segment intersection algorithm

Input: $\{s_1, \dots, s_n\}$ Finite set of line segments.



Output: set of intersection points

How do we find intersection point of $\overrightarrow{ab}$ & $\overrightarrow{cd}$?

Points on $\overrightarrow{ab}$ are $p = \lambda a + (1-\lambda) b$ for $\lambda \in [0,1]$.
$\overrightarrow{cd}$ ... $q = \mu c + (1-\mu) d$ ... $\mu \in [0,1]$.

Solve $\begin{cases} \lambda a_x + (1-\lambda) b_x = \mu c_x + (1-\mu) d_x \\ \lambda a_y + (1-\lambda) b_y = \mu c_y + (1-\mu) d_y \end{cases}$

system of 2 equations & 2 unknowns λ, μ.

Solution = intersection point.

Simple algorithm

- Given n line segments, test each pair for intersection.

Complexity

- No. of pairs is $\binom{n}{2} = \frac{n.n-1}{2}$
- Constant time to test a pair for intersection.
- Complexity : $O\left(\frac{n.(n-1)}{2}\right) = O(n^2)$
- Inefficient! We will describe a more efficient algorithm.

Idea

Often fewer than $\binom{n}{2}$ intersections.

Aim

Test for fewer intersections.

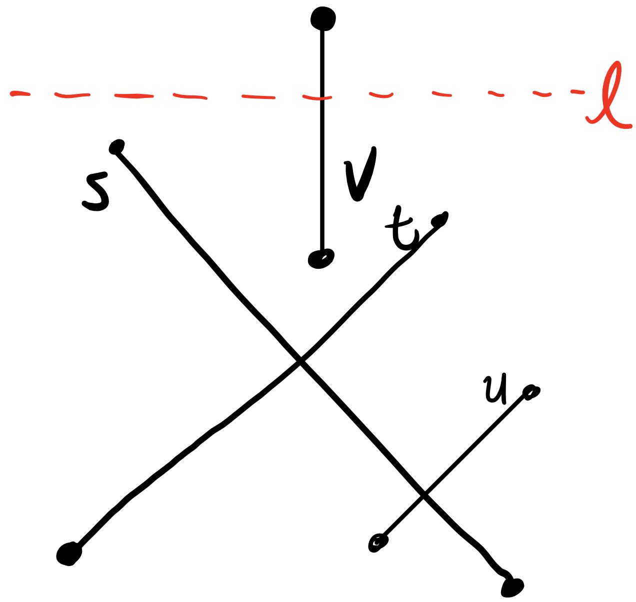
Will describe
output sensitive algorithm
with complexity
$$O((n+k)\log n)$$

no of line segments

k number of intersections found

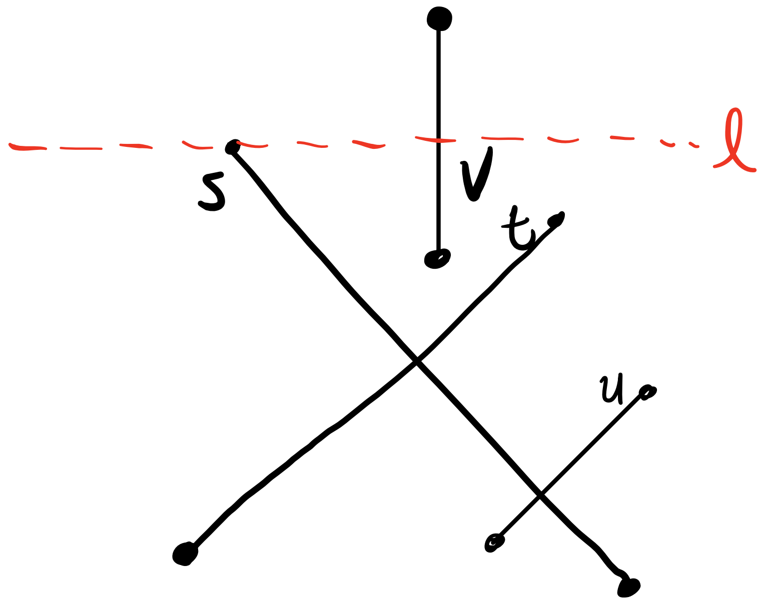
Called a sweep line algorithm.

Intuitive picture



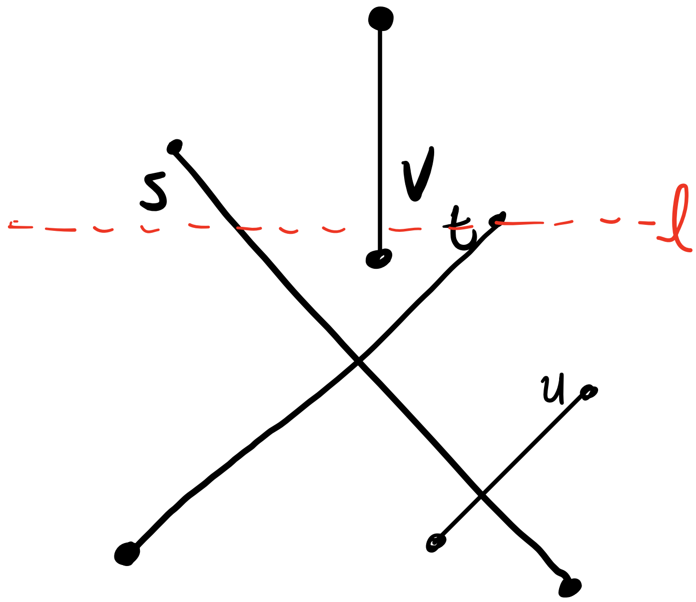
Imagine a line l running down the page from top to bottom.

Intuitive picture



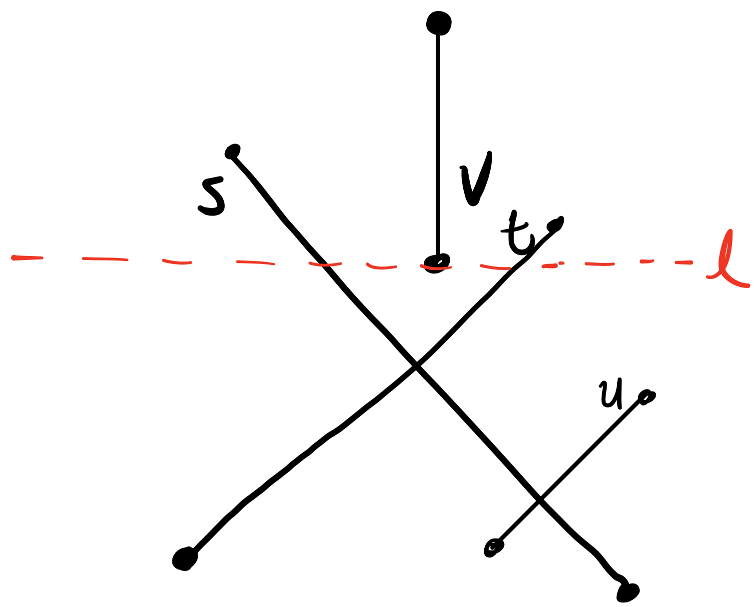
Imagine a line l running down the page from top to bottom.

Intuitive picture



Imagine a line l running down the page from top to bottom.

Intuitive picture



Imagine a line l running down the page from top to bottom.

- If two segments intersect, they must become neighbours (adjacent) @ some event point

endpoint or earlier intersection point

Idea: test segments for intersection just when they become neighbours.

Structures associated to algorithm

(1) "Event queue" Q ~ a balanced binary tree.

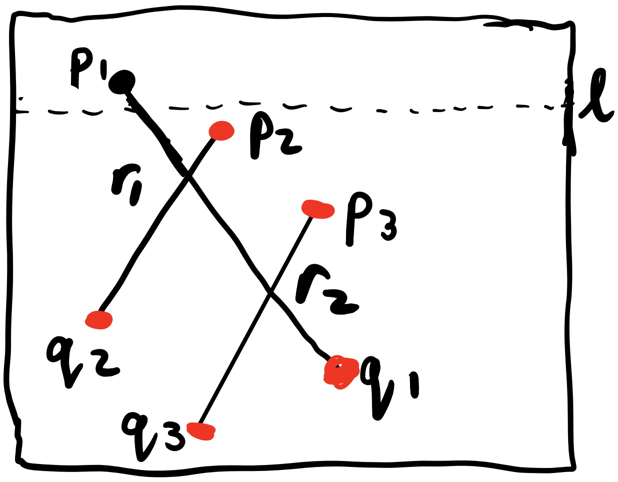
- Leaves of Q store the endpoints & computed intersections.
- updated as algorithm runs.

leaves

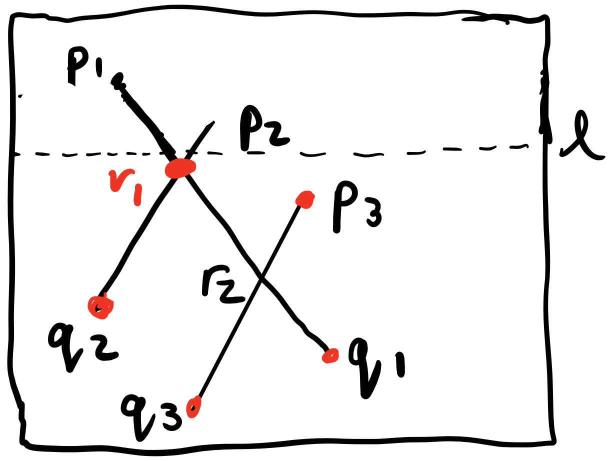
- Order on "event points" in Q is lexicographic: (top to bottom, left to right)

ie. $p < q \iff p_y > q_y$ or $(p_y = q_y \;\&\; p_x < q_x)$

Example (E-Learning 2.2)



$p_2 < p_3 < q_2 < q_1 < q_3$



$r_1 < p_3 < q_2 < q_1 < q_3$

Note: E-Learning: it says we only Q to be a queue, but need a bal. bin. tree.

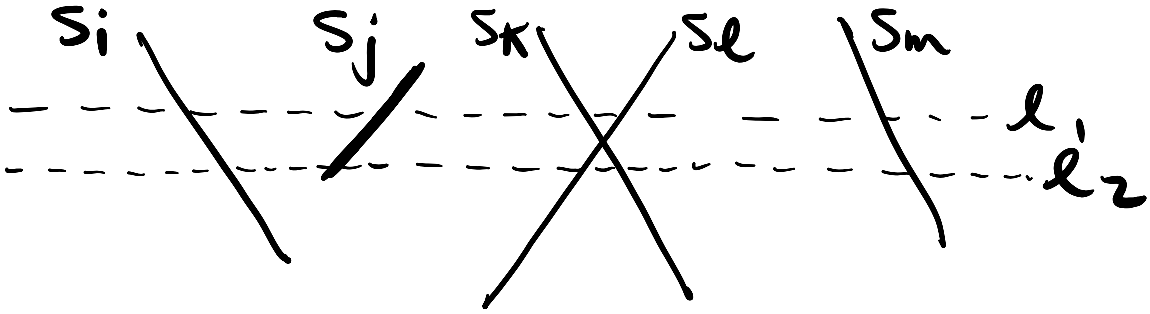
① Event queue continued

As a balanced binary tree,

- Inserting a new point to Q takes time $O(\log n)$.
- Finding next point in Q takes $O(\log n)$

② "Status structure" T – also balanced binary tree

- T stores the order of segments intersecting the sweep-line (left to right)

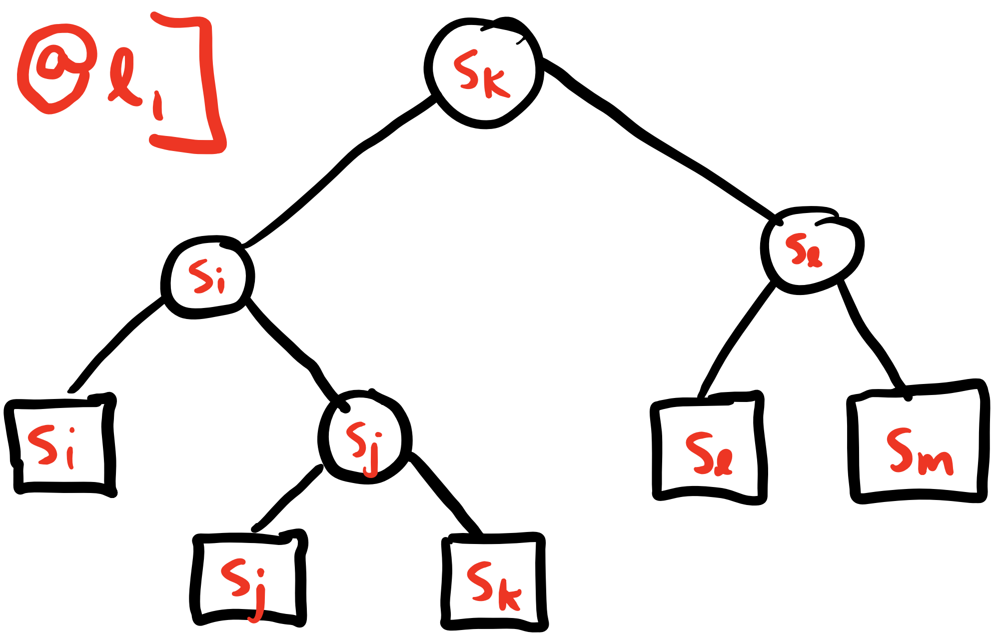


- Order in T at l_1 is $s_i < s_j < s_k < s_l < s_m$

  – – – – – at l_2 $s_i < s_j < s_l < s_k < s_m$

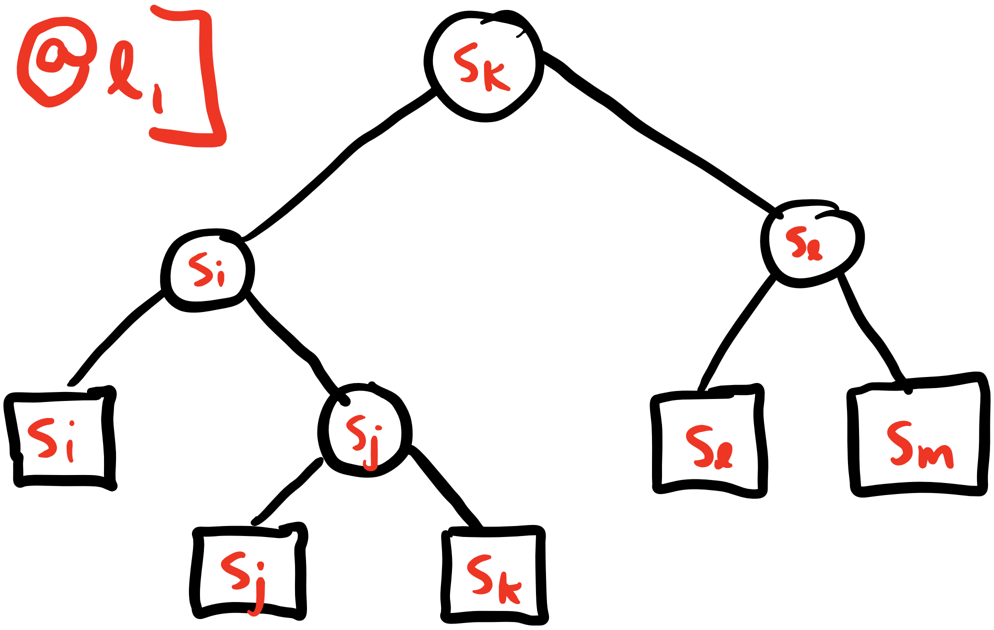
- Ordered segments – leaves of tree ~ see Fig 24 in E-Learning

@ l_1



(Value in node is rightmost leaf of left sub-tree)

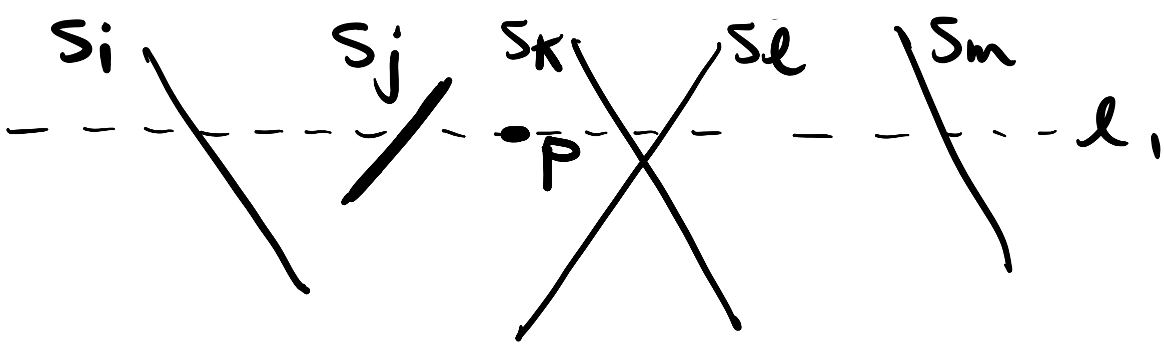
@l_1]



(Value in node is rightmost leaf of left sub-tree)

- Inserting, deleting segments from T takes time O(log n)

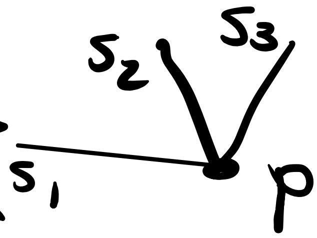
- Finding left, right neighbours of a point p on sweep-line takes Time O(log n)



left neighbour of p is S_j
& right neighbour of p is S_k

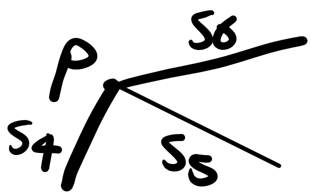
③ Also, we store for each event point p, the sets

$L(p) = \{$ segments with p as lower endpoint $\}$



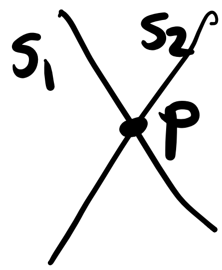
$L(p) = \{s_1, s_2, s_3\}$

$U(p) = \{$ — — — — — — p as upper endpoint $\}$



$U(p) = \{s_4, s_5\}$

$C(p) = \{$ — — — — — — p an interior point $\}$



$C(p) = \{s_1, s_2\}$

Algorithm

Input: $\{s_1, \dots, s_n\}$

Output: intersection points p plus sets $L(p)$, $U(p)$ & $C(p)$ of segments.

1) Add endpoints of segments to Q. Store $L(p)$ & $U(p)$ for each endpoint.

2) Initialise empty tree T.

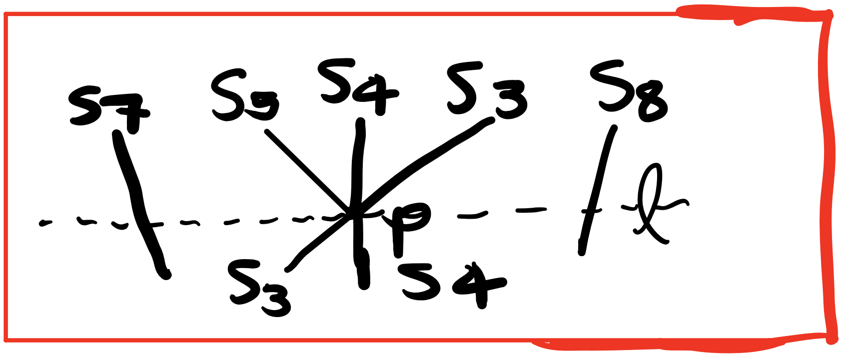
3) At next event point $p \in Q$,

a) Check if p an intersection pt. $(|L(p) \cup C(p) \cup U(p)| > 1)$. If so, report p & $L(p), C(p), U(p)$.

b) Delete p from Q.

c) Update tree T:

Eg @ p



$$s_7 < s_5 < s_4 < s_3 < s_8 \mapsto s_7 < s_4 < s_3 < s_8$$

- Do it by removing segments from $L(p)$, reverse order of those in $C(p)$, add those of $U(p)$.

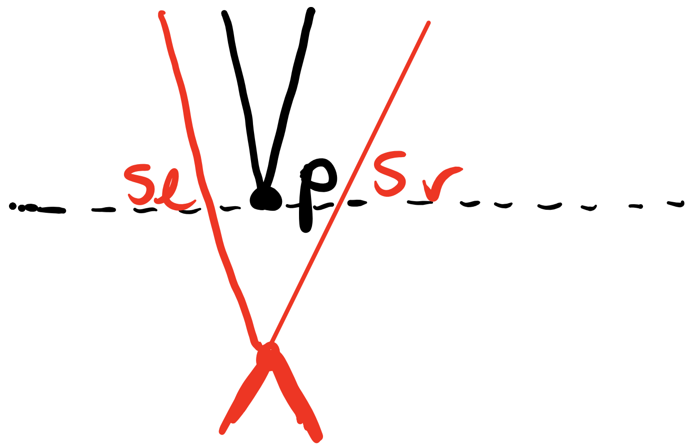
d) Compute intersections & add to Q.

e) When Q is empty, stop.

Details on d - compute intersections

① if $U(p) \cup C(p) = \phi$ (nothing coming out below p)

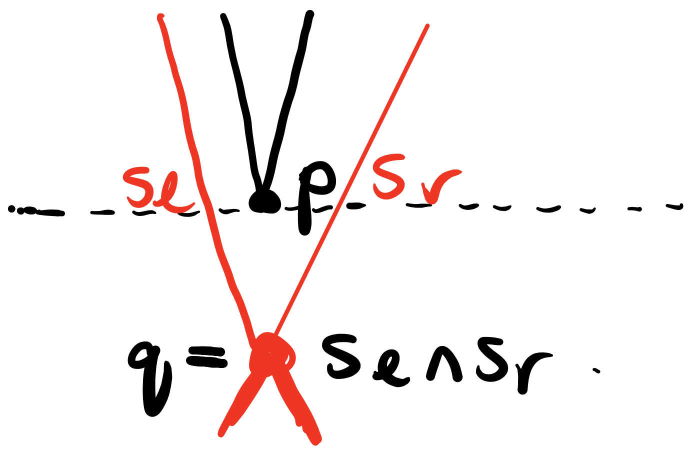
Using T, find left & right neighbours s_ℓ & s_r of p (if they exist)



Details on d - compute intersections

(I) if $U(p) \cup C(p) = \phi$ (nothing coming out below p)

Using T, find left & right neighbours s_ℓ & s_r of p (if they exist)

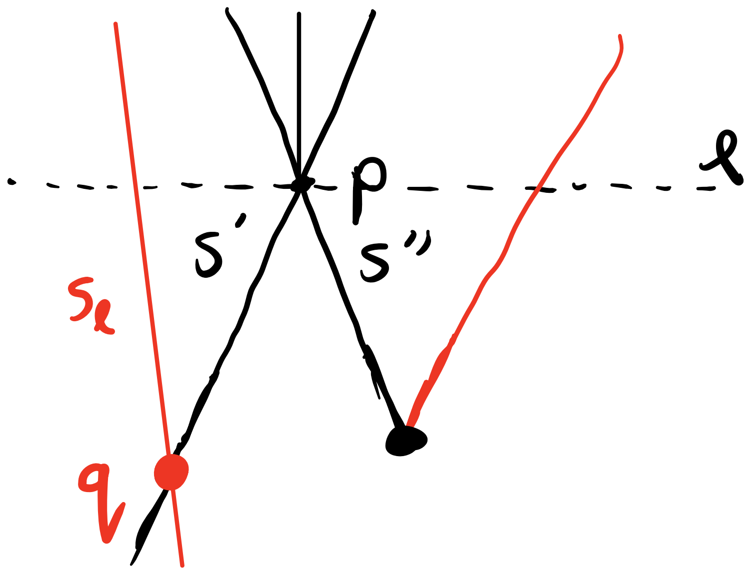


- Calculate $s_\ell \cap s_r$

- Update sets $L(q)$, $U(q)$ & $C(q)$ &

if q is a new intersection point we add it to Q.

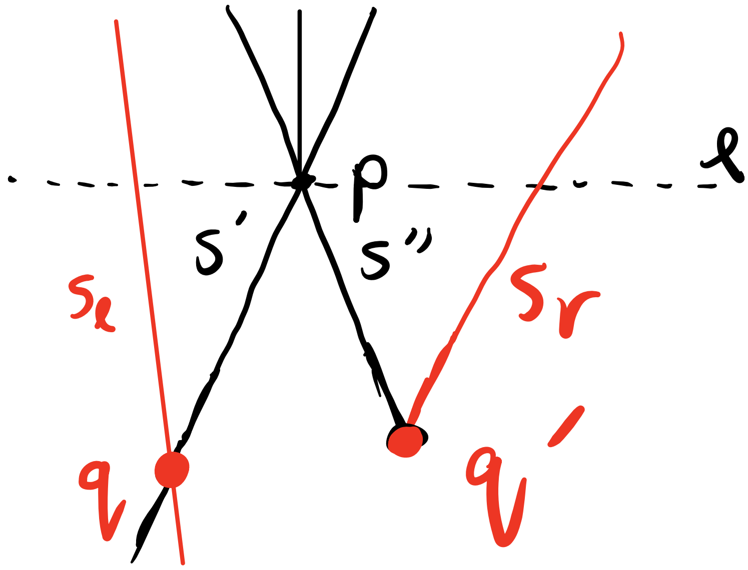
② Else, $U(p) \cup C(p)$ non-empty
(segment coming out below p)



- Let s & s' be leftmost & rightmost segments in $U(p) \cup C(p) \subseteq T$.
- Calc. left neighbour s_ℓ of s' & calc. $q = s_\ell \cap s'$.

  Add q to Q if it is new intersection point.

2 Else, $U(p) \cup C(p)$ non-empty
(segment coming out below p)



- Let s & s' be leftmost & rightmost segments in $U(p) \cup C(p) \subseteq T$.
- Calc. left neighbour s_ℓ of s' & calc. $q = s_\ell \cap s'$. Add q to Q if it is new intersection point.

plus update 3 sets

- Calc right neighbour s_r of s'' & $q' = s'' \cap s_r$. Add to Q if new.

See animation in E-learning.

Running Time

1) At start, order $2n$ endpoints into bin. bal. tree Q - $O(n \log n)$

2) Let $m(p) = L(p) \cup C(p) \cup U(p)$

Actions at event point p:

- add or remove a segment to / from T - $O(\log n)$ } Total $O(m(p) \log n)$

- Find s', s'', s_ℓ, s_r

($O(\log n)$ each so $O(4 \log n)$ Total)

- Computing intersection $O(1)$
- Inserting int. point in Q - $O(\log n)$

Total : $O(n \log n) + \sum_{p \text{ event}} m(p) O(\log n)$

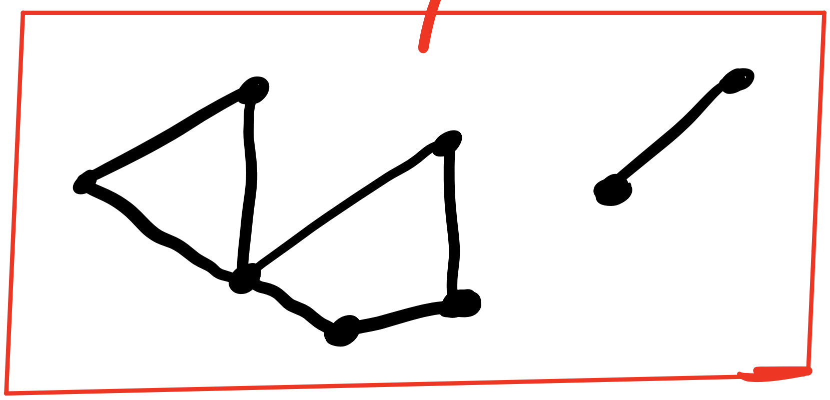
- Can simplify using graph theory.

To simplify, we use Euler's formula for planar graphs

count unbounded face

$V - E + F \geq 2$

$8 - 8 + 3 \geq 2$



- Each edge is adjacent to at most 2 faces.
- each bounded face adjacent to at least 3 edges
- so $3BF \leqslant 2E$ so

  BF = bounded faces

  $F - 1 = BF \leqslant 2E/3$ so

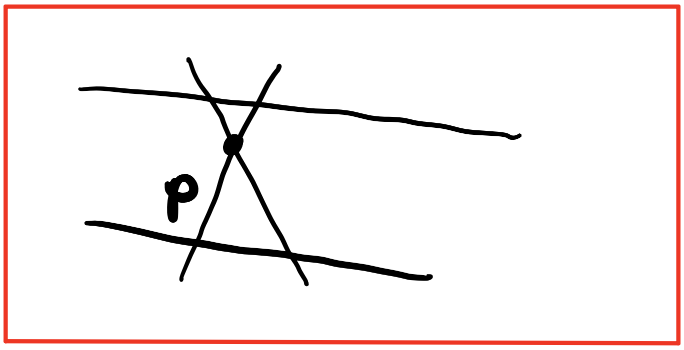
  $F \leqslant 2E/3 + 1$.

- Then $V - E + F \geq 2 \Rightarrow$

  $V - E + 2E/3 + 1 \geq 2 \Rightarrow$

  $V - 1 \geq E/3 \Rightarrow E \leqslant 3(V-1)$

- Planar graph of segments, endpoints intersections



vertices = events
i.e. endpoints & intersections

Degree $s(p)$ of p = no. of edges coming out of p

- In above $s(p) = 4$, $m(p) = 2$.
- In general $m(p) \leq s(p)$
- So $\sum\limits_{p \text{ an event pt.}} m(p) \leq \sum\limits_{p} s(p) = 2E$

as each edge in planar graph has exactly 2 endpoints

$$\leq 6(V-1) \leq 6(\underbrace{2n}_{\text{endpoints}} + \underbrace{k}_{\text{internal pts}} - 1)$$

$$\leq 12(n+k)$$

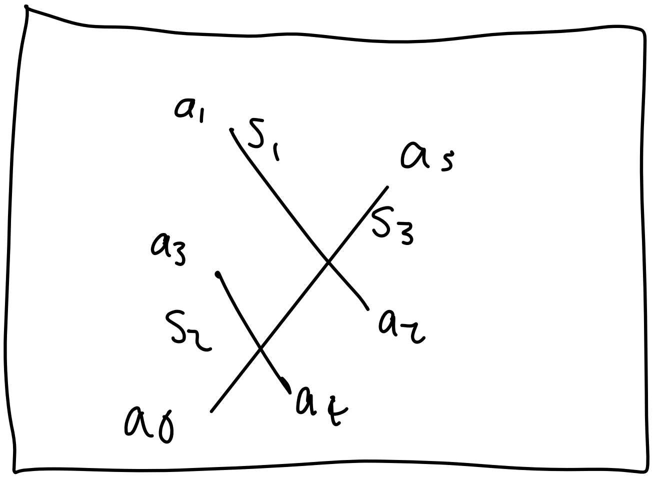
Complexity $O(n \log n) + \sum\limits_{p} m(p) O(\log n)$

$$\leq O(n\log n) + 12(n+k) O(\log n) = O((n+k)\log n)$$

- This is the output sensitive complexity that we claimed at the beginning of the lecture.

- Sweep-line algorithm will also be used next week for "map overlay" and in later weeks.

a)



What happens when the alg. runs?