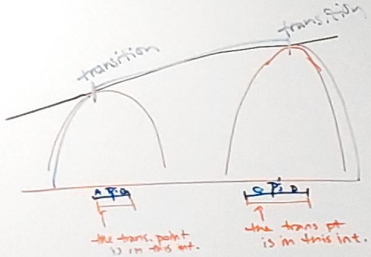


Question:

Given 2 line segments, do they intersect?

$S_1 = (p_1, q_1)$

$S_2 = (p_2, q_2)$

↑ p's are the start of the segment

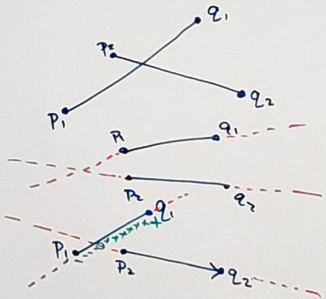
WLOG, assume $p_i.x < q_i.x$

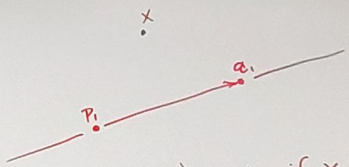
... w/out computing intersections of lines.

... nor y-values for a given x-value ← but this thought is helpful for today

only tool: side-of-line test

Examples:





orient (p_1, q_1, x) = +, if x on Left of line from p_1 to q_1

"side of line" test

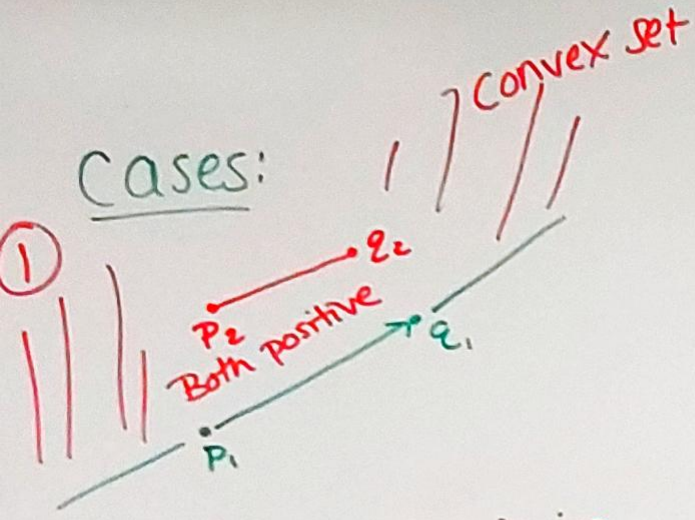
orient (p,q,x) = orient (q,x,p) = orient (x,p,q)

↑ cyclic permutations result in same answer.
more gen: even permutations

= − orient (q,p,x)

GP assumption: pts = $\{q_i\} \cup \{p_i\}$
- no 3 colinear pts ⇝ avoids
- no shared x-coord (or y)

Cases:



$\Rightarrow$ Line seg s_2 is in pos. half plane (open)

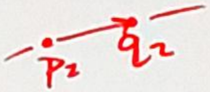
$\therefore$ intersection of 2 lines is not on that line segment.

(2) Both on neg. side.

Symmetric argument.

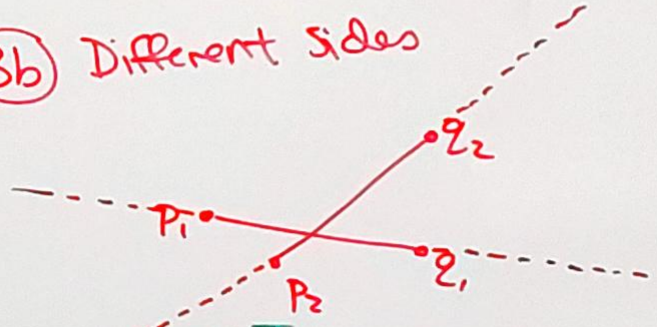
(3) p_2 on pos. side
q_2 on neg side

(3a) Check SOL of q_1, p_1 wrt



Same side: no intersection
(we've seen this before)

(3b) Different sides



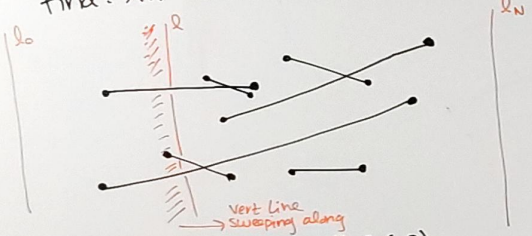
Pf: IVT

thinking about dot. value & not just the sign.

Given: $S = \{s_i = (p_i, q_i)\}_{i=1}^{n}$

set of line segments

Find: All intersection pts of segments



Worst-case, how many intersections? $\Theta(n^2)$

Naive: check if each pair intersects.
$\Theta(n^2)$. Can't get better, right?

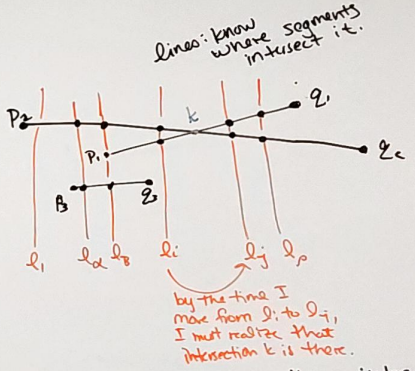
But, we can do better if we think output sensitive. m = # of actual intersections

Sweep line algorithms

Loop invariant "moves along" with l

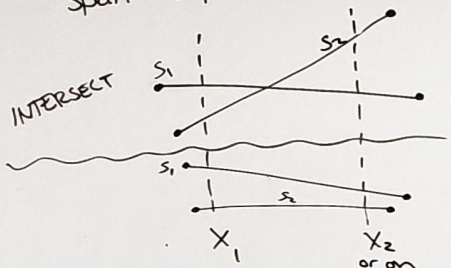
- To the left of l, we know all intersections
- To the right of l, murky, but we know some things
- on l, we keep track of something that helps us make progress / push l along.

Q: What happens as I move l along?



NOTE: I don't care about where things intersect with l, I just care about the order.

Claim: If 2 segments (s_1 and s_2) x-values span X_1 to X_2



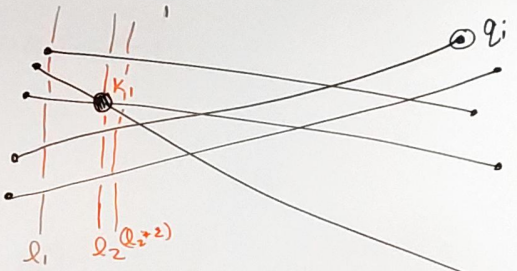
(i.e., left endpt to left of X_1 and right endpoint to right or on X_2)

Then, if X_1 & X_2 see s_1 and s_2 in different orders, then they intersect in that interval!

Events: discrete set of "times" that something happens (or needs to be updated)

Types of Events:

1) l passes a start point, p_i
   $\Rightarrow$ add $s_i \cap l$ to what l is keeping track of
2) l passes an endpoint, q_i:
   $\Rightarrow$ we can drop s_i from the ordered intersections along l
3) INTERSECTIONS



Keep track in a priority queue
INIT: w/ type 1) and 2) events. Add (remove) type 3) as we go along.

The intersection, if the next event, must be between 2 consecutive line segments on l.