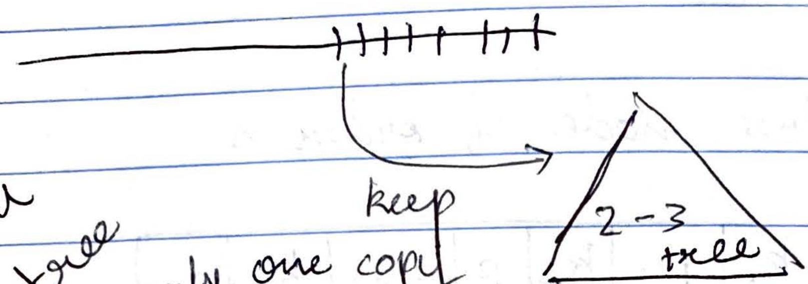
Large data stream



Application of 2-3 tree

keep only one copy ~~only~~ of a data element in 2-3 tree

Segment Intersection (Application of 2-3 trees)

Given a set of line segments, ~~represent~~ report all intersections

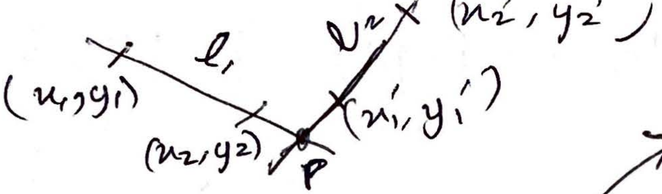
$l_1 ((x_1, y_1), (x_2, y_2))$ $\quad$ $l_2 ((x'_1, y'_1), (x'_2, y'_2))$

For each pair of segments (l_1, l_2) of segments report the intersection if they intersect.

→ Solve ~~of~~ line equations for x, y ~~interce~~ intersection.

→ Check if intersection is on the segment, Put the intersection point back into the equation and see if points (x_1, y_1) & (x_2, y_2) lie on the

Another application of 2-3 trees is to make sure you're keeping only one copy of an intersection



Check if P is one side of both points

Same side of intersection

Time = $O(n^2)$ because n line segments

Geometric sweeping is better than $O(n^2)$

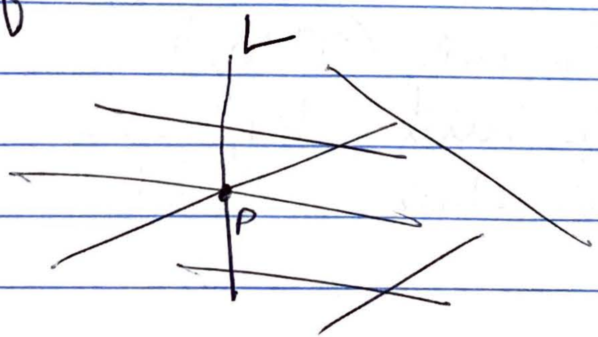
We use a sweeping line L that keeps all segments intersecting L, → configuration of L

L stops at
1) next intersecting point
2) left end of a new segment
3) right end of a segment on L

b/w these points the relative positions of line segments are unchanged.

→ These have to be updated dynamically

* We use 2-3 trees for configurations of L
* We use 2-3 trees ~~for~~ T to keep stopping points of L



input : $l_i = (p_i q_i)$ $i = 1, 2, 3 \ldots n$

1) for ($i=1$; $i<=n$; $i++$)
   Insert (T, pi);
   Insert (T, qi);

T is a 2-3 tree

2) while (T ! empty)
   p = Min(T); $\xrightarrow{\text{Delete}(T,p)}$ will give the ~~leftmost~~ point.

2.1) if (p is leftend of l_t)
   Insert (L, l_t) ← insert line segment into L
   for each neighbour l' of l_t in L)
   if (l' and l_t intersect at q)
   Insert (T, q)

2.2) else if (p is right end of l_i)
   Delete (L, l_t)
   if (the two old neighbours of l_t intersect at q on the right side of L)
   Insert (T, q) O(log

2.3) else // p is intersection of l_s and l_t
   report(p) → interchange the positions of l_s & l_t in L
   → for (~~each~~ the new neighbour l_s' of l_s)
   if (l_s' and l_s intersect at q on the right)
   Insert (T, q)
   if (l_t' and l_t - - - - -)
   Similar - -

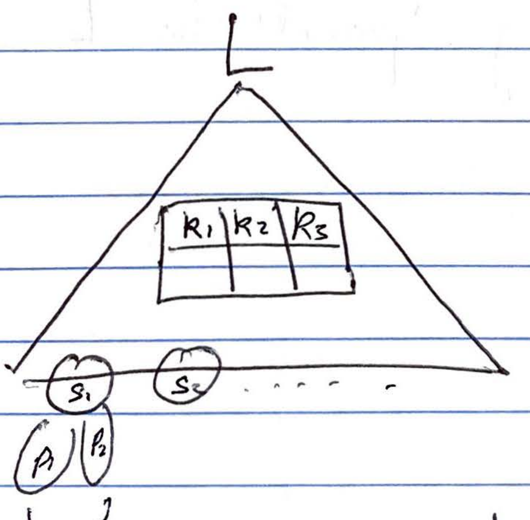
Worst case :- For n line segments there are n^2 intersections. Therefore the complexity for the algo is $O(n^2 \log n)$

- - - - - - - - - - - - - - - - - -

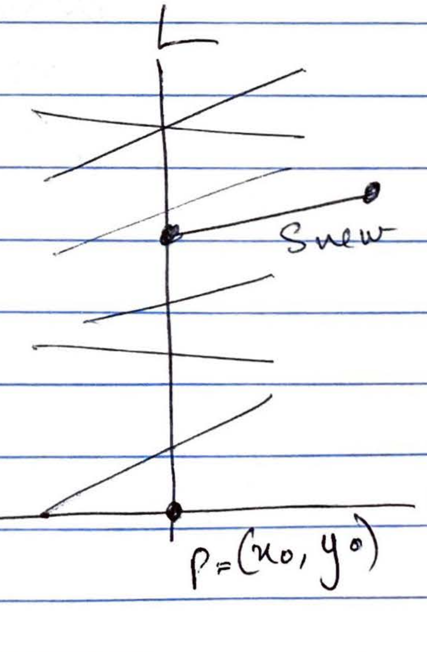
<u>Seg - Intersection</u>

1. intersect all end pts of segments in T
2. For (the next stopping point p in T)
   process
   insert a seg in L
   delete a seg in L
   exchange portions of two segs in L
   insert new stopping pt in T.

$O(\log n)$



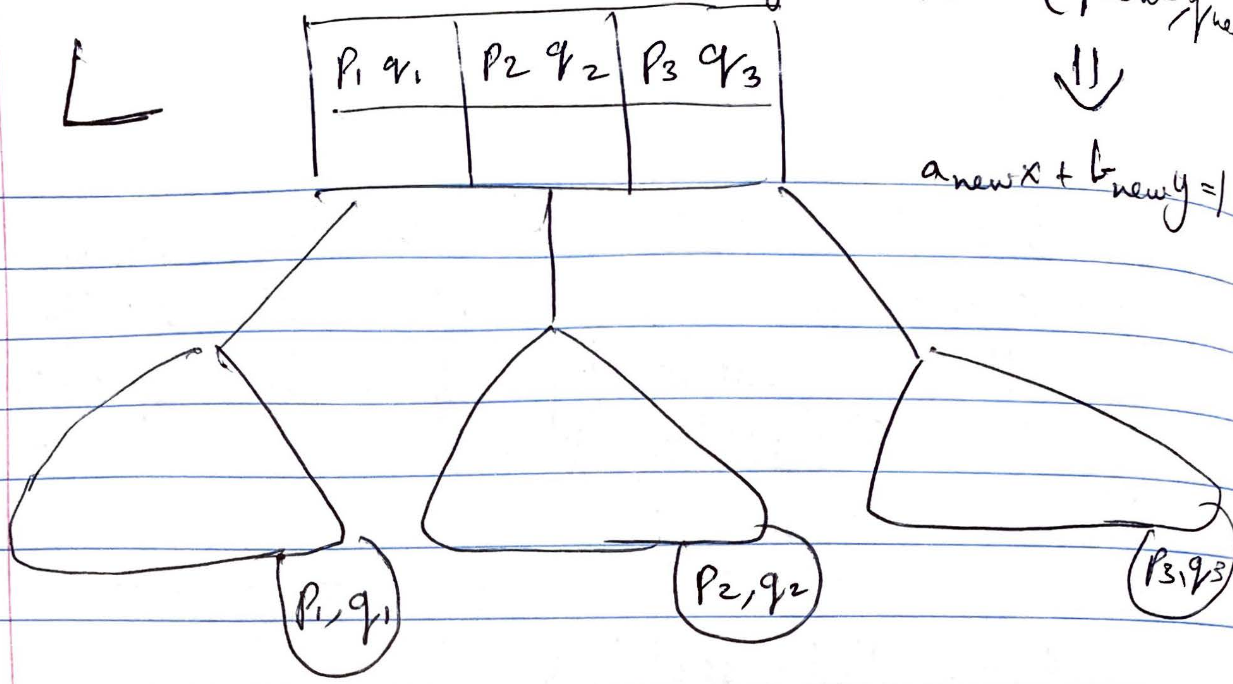
endpoints of segments



New segment $S_{new} = (p_{new}, q_{new})$

$\Downarrow$

$a_{new}x + b_{new}y = 1$



To insert the new segment S_{new} into the tree you have to know where S_{new} lies w.r.t the other segments in L vertically. We can compute the y coordinate of S_{new} using p_{new} and the sweeping line's x-coordinate and compare it with the existing points in L.

Worst case complexity $= O(n^2 \log n)$

$$= O\Big(\big(\underbrace{2n}_{\text{stopping points}} + \underbrace{w}_{\text{\# of intersections}}\big) \log n\Big) \quad \text{or } O(n^2)$$

To find an algo that ~~ba~~ performs better than both of them, ...

input = no. of ~~points~~ intersects $= 10n^2$

??