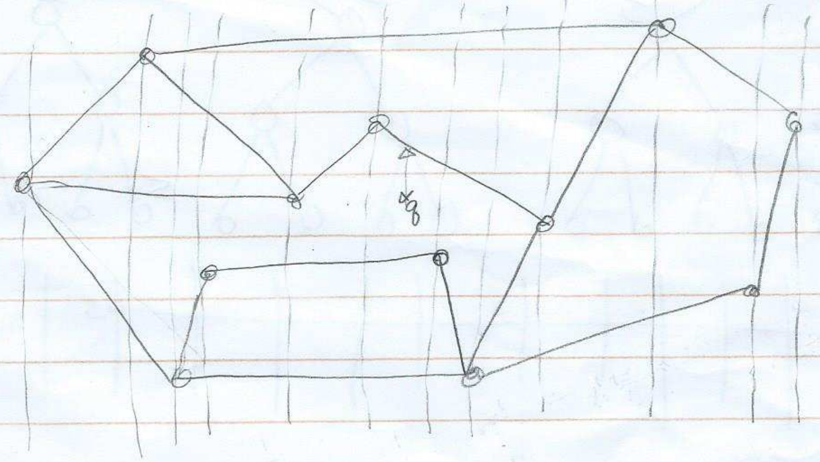
Planar Point Location

by Euler's formula, # edges/faces = $O(n)$

store planar subdivision with n vertices

s.t. given query pt q, find region containing q

equivalently: find line segment immediately above q.



Method 0:

divide into n vertical slabs

store y-sorted list in each slab

$\Rightarrow$ query time $O(\log n)$ (binary search in x + binary search in y)

space $O(n^2)$

preproc time $O(n^2 \cancel{\log n})$

Method 1: Segment Tree

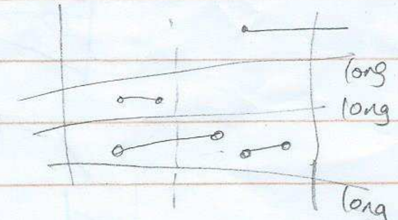
given n segs intersecting slab σ,

divide by median x

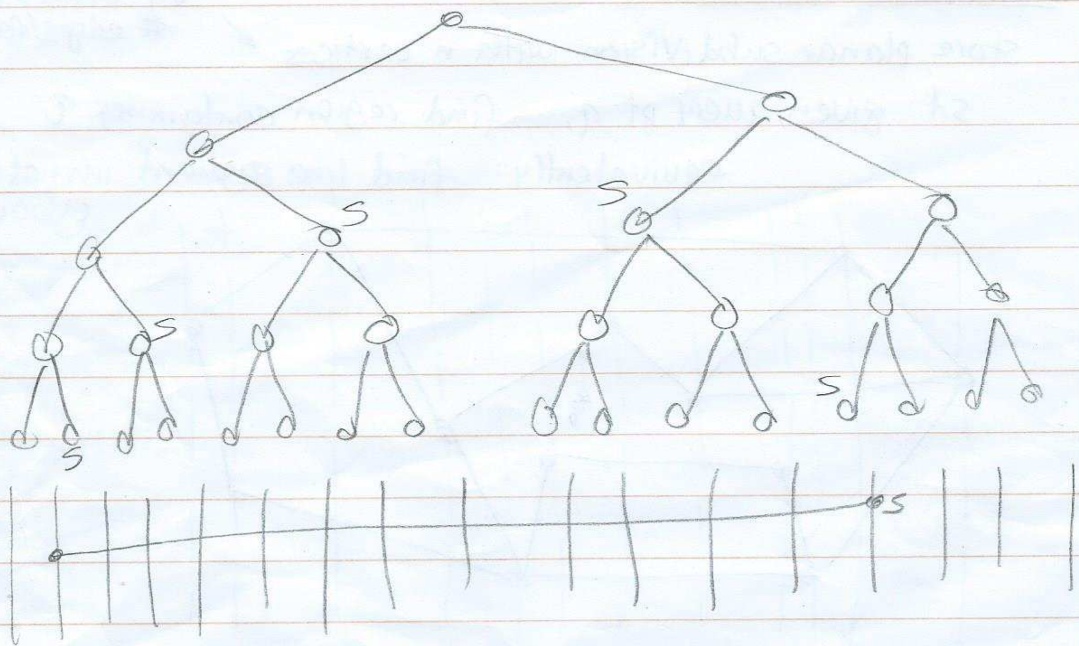
remove all long segs in σ

& store them in y-sorted list

recurse in left & right subslabs



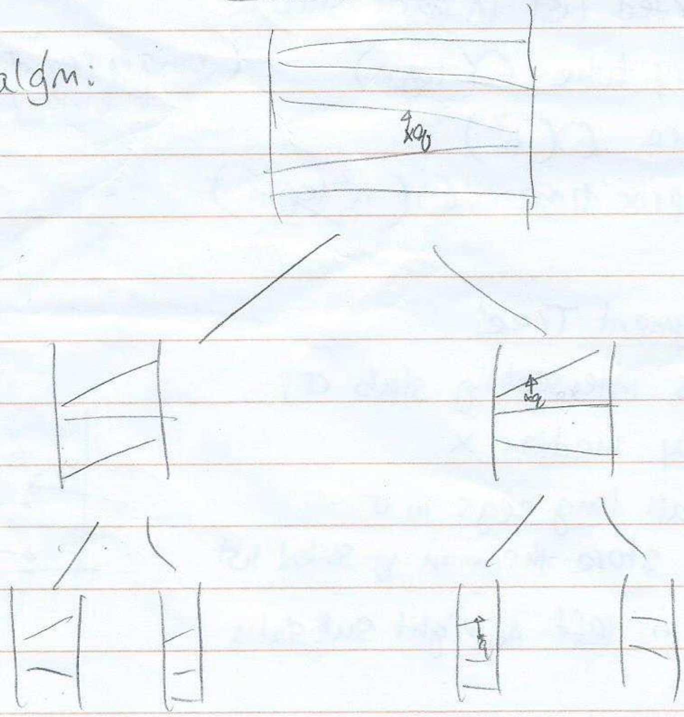
Def s is long in σ if it completely cuts across σ



each seg is stored $O(\log n)$ times

$\Rightarrow$ space $\boxed{O(n\log n)}$ (preproc time $O(n\log n)$)

Query algm:



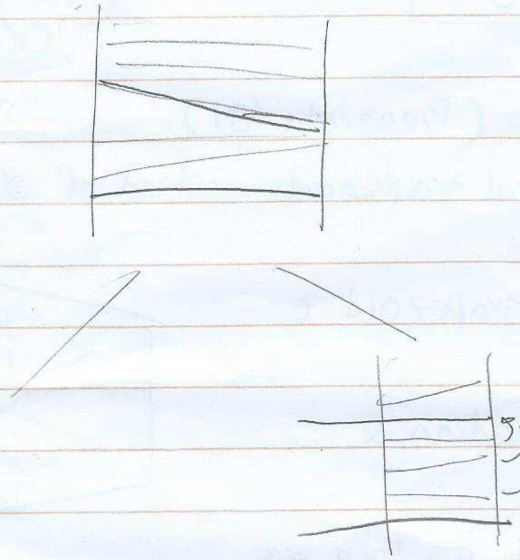
binary search in each slab containing q

$\Rightarrow O(\log n)$ binary searches

$\Rightarrow \boxed{O(\log^2 n)}$ time

Method 2: Segment Tree + "Fractional Cascading" (Chazelle, Guibas '86)

idea - to speed up query,
pass fraction $\frac{1}{b}$ of the list of each node u
to the list of each child v of u



e.g. $b=3$

$L(u)$ = orginal y-sorted list of u

$L^+(u)$ = "augmented" list

sample(L) = take one after every b^{th} element in L

for each child v of u,
define $L^+(v) = L(v) \cup \text{sample}(L^+(u))$.
store ptrs between $L^+(v)$ and $L(v)$, sample($L^+(u)$).

if we know succ of q in $L^+(v)$,
⇒ know succ of q in $L(v)$ & in sample($L^+(u)$) in $O(1)$ time
⇒ find succ in $L^+(u)$ in $O(b)$ time
const

query time $O(\log n + \log n) = \boxed{O(\log n)}$
init binary search at leaf

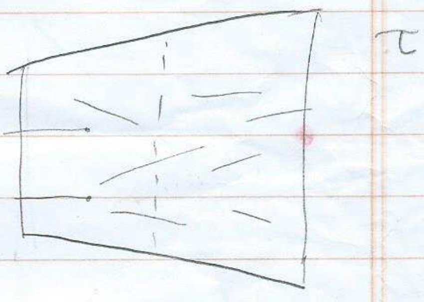
$$\text{space} \; \sum_u |L(u)| \left(1 + \frac{2}{b} + \left(\frac{2}{b}\right)^2 + \left(\frac{2}{b}\right)^3 + \cdots\right)$$
$$= O\left(\sum_u |L(u)|\right) \quad \text{for } b > 2$$
$$= \boxed{O(n \log n)}$$

(can reduce space to $O(n)$ with additional ideas...)

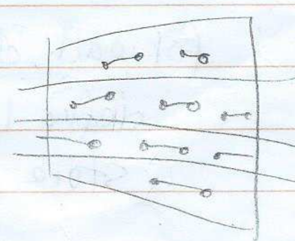
<u>Method 3. Trapezoid Tree</u> (Preparata '81)

idea: tree of vertical trapezoids instead of slabs

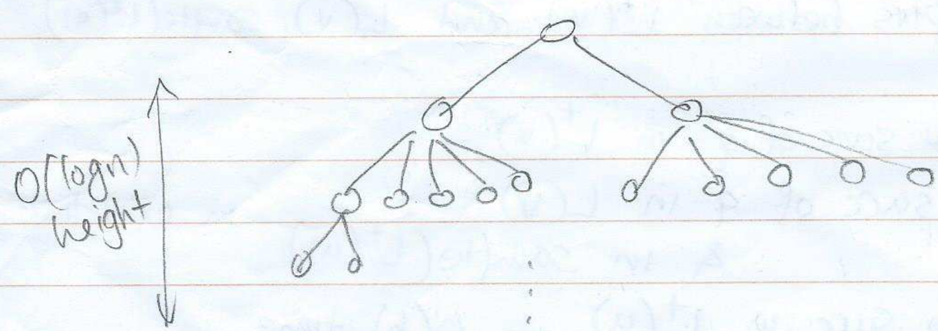
Given n segs intersecting trapezoid τ:



if no long segs,
  divide τ by median x
else
  divide τ by <u>all</u> its long segs



multi-deg node!



$\boxed{O(n \log n)}$ space as before

Query: naively $O(\log n \cdot \log n) = O(\log^2 n)$

binary search at multi-deg node

Fact given m elements $y_1, \ldots, y_m$ in 1D,
with weights $w_1, \ldots, w_m$, $\sum_i w_i = W$,
can find pred/succ y_i of any query pt q
in $O(\log \frac{W}{w_i} + 1) = O(\log W - \log w_i + 1)$ time

Pf: by ~~weighted~~ biased binary search (using wt'd median)

Query alg'm:
at each multi-deg node, do weighted search
with weight(τ) = # segs intersecting τ

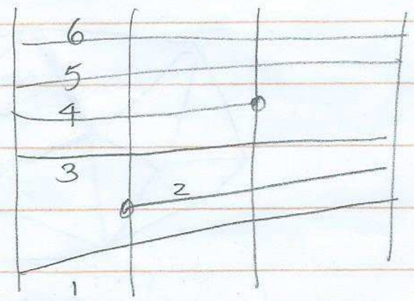
$\Rightarrow$ total query time $O(\log W_0 - \log W_1 + 1 + \log W_1 - \log W_2 + 1 + \ldots)$ telescoping sum with $\log n$ terms

$= O(\log W_0 + \log n)$

$= \boxed{O(\log n)}$

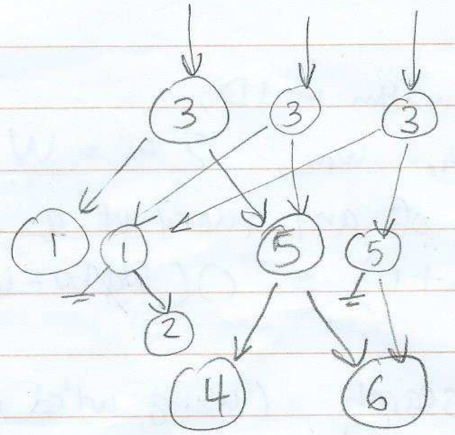
(Rmk: related to BSP tree)
(Rmk: for tree-like structures, $\Omega(n \log n)$ space)

Method 4: Persistent Search Tree (Sarnak-Tarjan '86)
idea - back to Method 0 (slabs)



sweep from left to right
if left endpt,
insert to list ← stored in balanced search tree
if right endpt,
delete

remember history of all changes to search tree
& "persistence" - ability to query in past



insert(2)
delete(4)

by <u>path copying</u>

(can do rotations...)

query time $O(\underbrace{\log n}_{\text{init. binary search}} + \underbrace{\log n}_{\text{time to query in search tree}}) = \boxed{O(\log n)}$

Space $\boxed{O(n \log n)}$

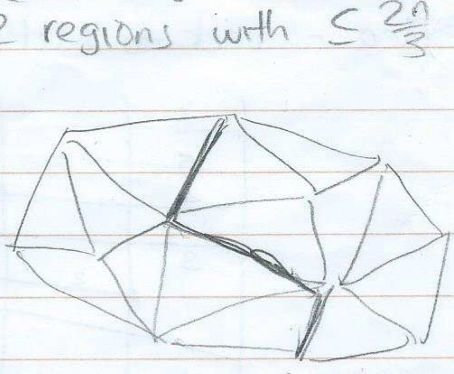
preproc time $\boxed{O(n \log n)}$

Rmk - can improve space to $O(n)$ ... (by fattening nodes to do less copying)

<u>Method 5</u>: <u>Planar Separators</u> (Lipton-Tarjan '77)

<u>Thm</u> Given a triangulated planar graph G with n faces,
can find subset R of $O(\sqrt{n})$ edges
that divide G into 2 regions with $\leq \frac{2n}{3}$ faces

1D



<u>Generalized Thm</u> for any given b, can find subset R of $O(\frac{n}{\sqrt{b}})$ edges
that divide G into regions with $\leq b$ faces.
(Pf: apply Thm recursively ...)