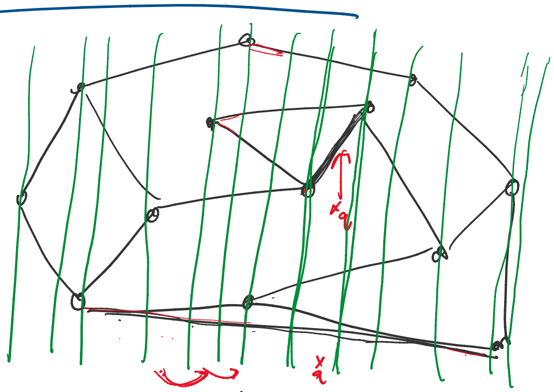
September 7, 2021 10:11 AM

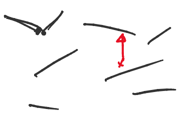
Planar Point Location



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

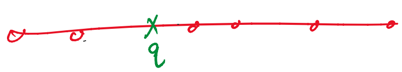
store planar subdivision
s.t. given query pt q, find region containing q.

equiv: find line segment immediately above q



in 1D, $O(n)$ space
$O(\log n)$ time

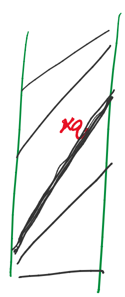
preproc time $O(n \cancel{\log n})$



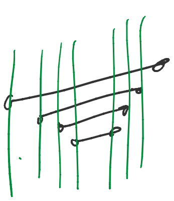
$\geq$ 8 different methods

Method 0: "Slab"

divide into n vertical slabs
store y-sorted list of line segs in each slab



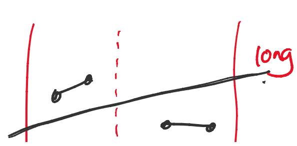
$\Rightarrow$ query time $\boxed{O(\log n)}$
by binary search in x
+ another binary search using y.



space $\boxed{O(n^2)}$ bad!
preproc time $\boxed{O(n^2 \cancel{\log n})}$

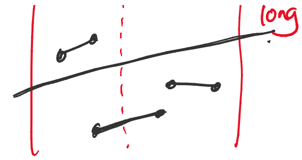
Method 1: Segment Tree

given n segs intersecting vertical slab σ,



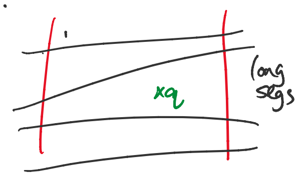
given n segs intersecting vertical slab σ,

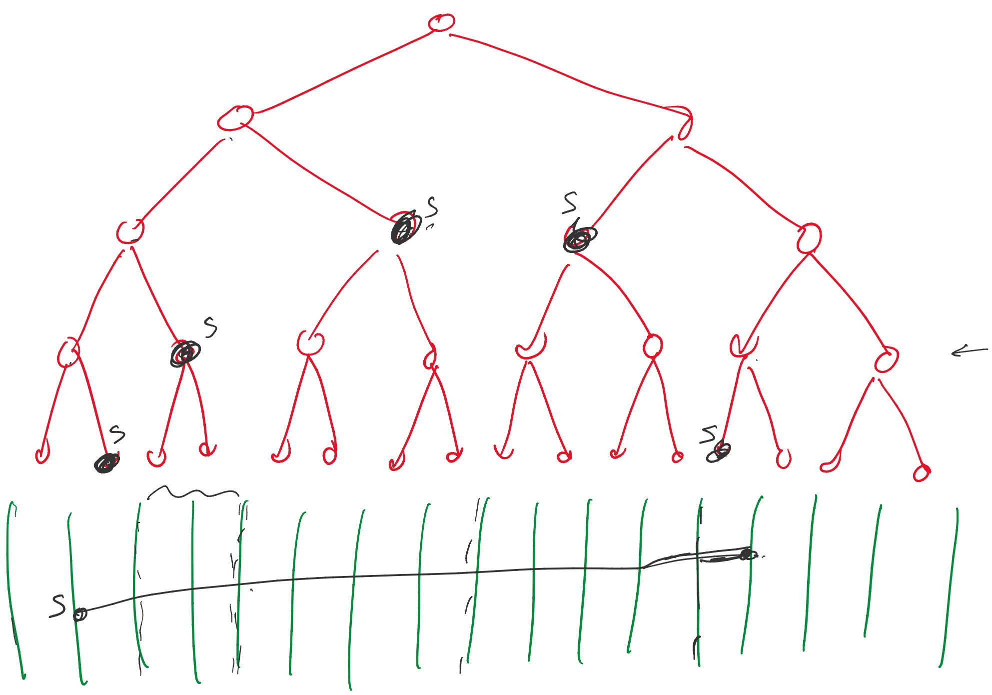
divide by median x
remove all long segs in σ & store them in y-sorted list
recurse in left & right subslab.



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

short else



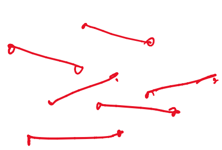


seg is stored at σ iff seg is long at σ but is short at its parent

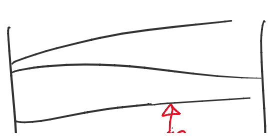
each seg is stored ≤ 2 logn times

⇒ space $\boxed{O(n \log n)}$. ("dual" to range tree)

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

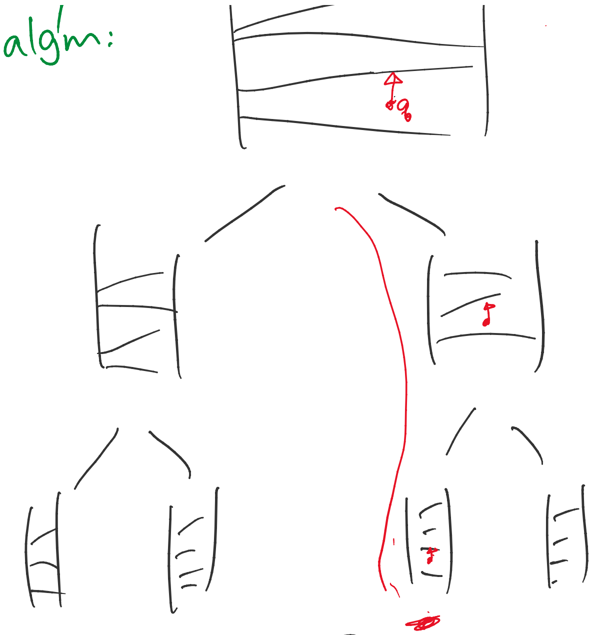


Query alg'm:



O(logn) nodes

Query alg'm:



$O(\log n)$ nodes

binary search in each slab containing q

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

"decomposable search prob"

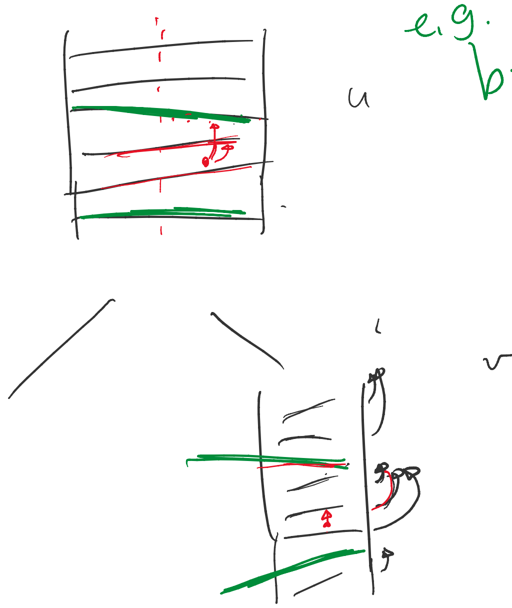
how to speed up query?

(parent list & child list not related ...)

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

idea - pass fraction $\frac{1}{b}$ of the parent list to the child list

e.g. $b=3$



let $L(u)$ = original y-sorted list at node u

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

let $L(u)$ = original y-sorted list at node u

Will generate an "augmented" list $L^+(u)$

let $\text{Sample}(L)$ = sublist of L formed by taking one after every $\underline{b}^{th}$ element

for each child v of u,

let $L^+(v) = L(v) \cup \text{sample}(L^+(u))$

store ptrs between $L^+(v)$ and $L(v)$

& between $L^+(v)$ and $\text{sample}(L^+(u))$

if we know pred/succ of q in $L^+(v)$,

$\Rightarrow$ know pred/succ of q in $L(v)$ ✓

& also also in $\text{sample}(L^+(u))$

$\Rightarrow$ find pred/succ in $L^+(u)$ in $O(\cancel{b})$ time $O(1)$

query time $O(\log n + (\log n) \cdot O(1)) = \boxed{O(\log n)}$

init binary search at leaf

Space

$$O\left(\sum_u |L(u)| \left(1 + \frac{2}{b} + \left(\frac{2}{b}\right)^2 + \left(\frac{2}{b}\right)^3 + \dots\right)\right)$$

for $b > 2$

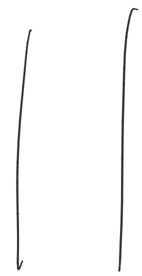
$$= O\left(\sum_u |L(u)|\right)$$

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

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

Method 3: Trapezoid Tree (Preparata '81)

use a tree where cells are vertical trapezoids instead of slabs



Given n segs intersecting trapezoid τ,

if no long segs,

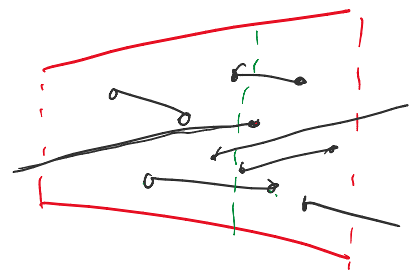
divide τ by median x

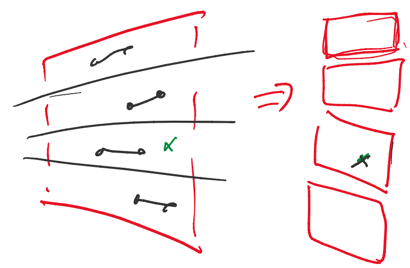
degree-2 node

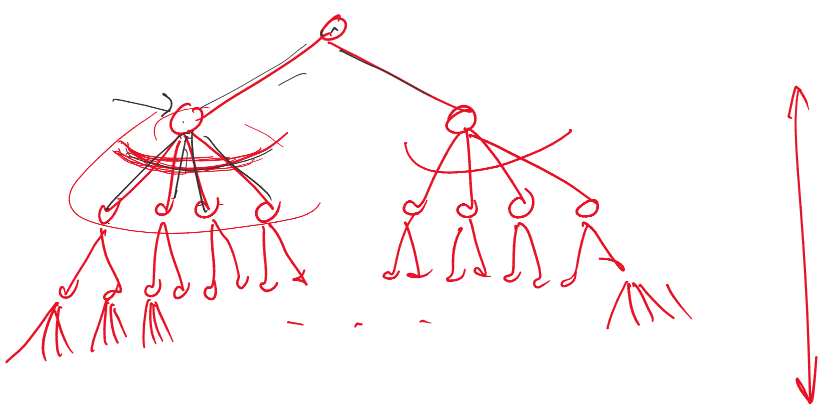
else

divide τ by all long segs

multidegree node







$O(\log n)$ height

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

query time $O(\underbrace{\log n}_{\#\text{ levels}} \cdot \underbrace{\log n}_{\text{binary search at each multi-deg node}}) = \boxed{O(\log^2 n)}$

TO BE CONT'D ...