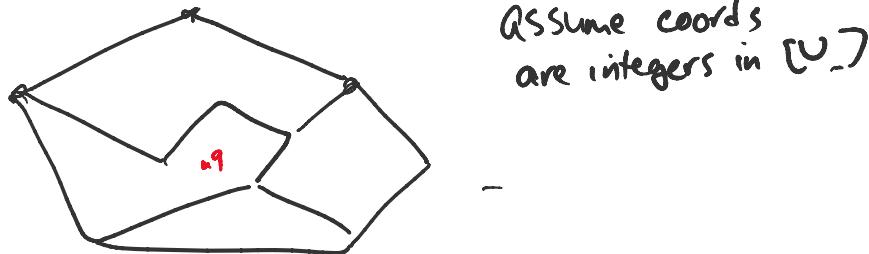


Point Location in Sublog Time

Last time: orthogonal case

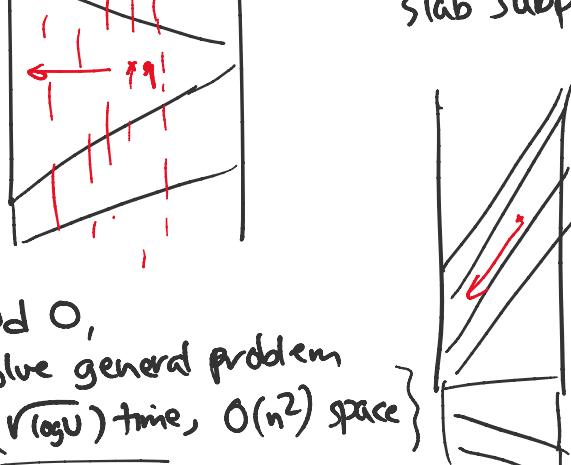
$O(n)$ space, $O(\log \log U)$ time

Today: nonorthogonal case



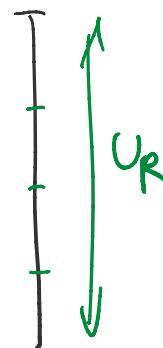
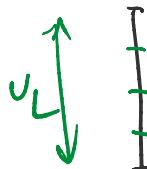
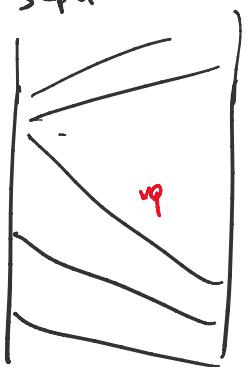
C.-Patrascu '06: $O(n)$ space, $\underline{O(\sqrt{\log U})}$ query time

Suffice to solve slab subproblem }



by Method O,
can solve general problem
in $\underline{O(\sqrt{\log U})}$ time, $O(n^2)$ space}

& then reduce space by
separator or sampling method



idea - divide & conquer
to reduce left/right universe

idea - divide & conquer
to reduce left/right universe

divide left universe into k subintervals of length U_L/k .

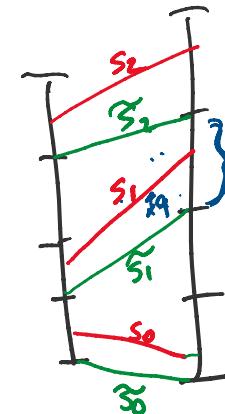
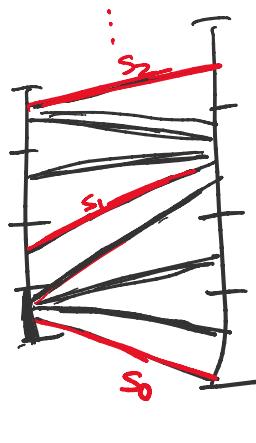
& right " " " "
 U_R/k .

Preproc algm:

s_0 = lowest seg

for $i=1, 2, \dots$

s_i = lowest seg s.t.
left endpoint is in
a higher subinterval than
 s_{i-1}
& right " " " "
 s_{i-1}



recursively build DS for segs between s_{i-1} & s_i

either left universe $\rightarrow U/k$
or right universe $\rightarrow U_R/k$.

build DS for $\tilde{s}_0, \dots, \tilde{s}_k$ $\tilde{s}_0 \prec s_0 \prec \tilde{s}_1 \prec s_1 \prec \dots$

segs obtained by rounding s_0, \dots, s_k downward

Query algm, given pt q :

→ locate q in $\tilde{s}_0, \dots, \tilde{s}_k$

⇒ can locate q in s_0, \dots, s_k with 1 extra comp.

say $s_{i-1} \prec q \prec s_i$

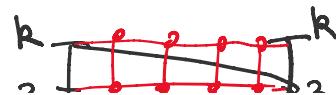
recurse between s_{i-1} & s_i .

Lemma can build pt location DS for $\tilde{s}_0, \dots, \tilde{s}_k$
with $O(1)$ query time
& $O(k^2)$ space.

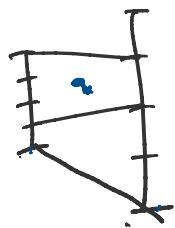
Pf:



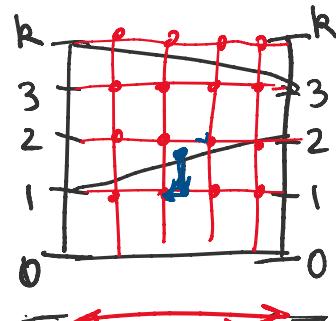
Projective
Localization



Pf:



Projective
transform



store all k^2 answers in a table

given q , apply proj transform
then round to integer pt in $[k]^2$
& do table lookup

(off by $O(1)$)
at most

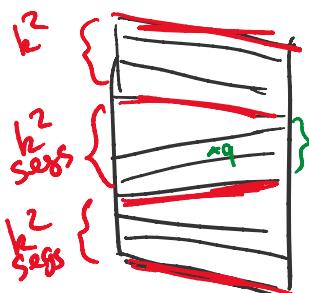
Query time

$$Q(U_L, U_R) \leq \begin{cases} Q\left(\frac{U_L}{k}, U_R\right) + O(1) & \text{or} \\ Q\left(U_L, \frac{U_R}{k}\right) + O(1) \end{cases}$$

$$\Rightarrow O(\log_k U_L + \log_k U_R) = O(\log_k U)$$

Space $O(k^2 n)$

Final Step: reduce space by separator of size $\frac{n}{k^2}$



\Rightarrow space $O(n)$

query time

$$O(\log_k U + \underbrace{\log(k^2)}_{\text{by binary search}})$$

$$= O\left(\frac{\log U}{\log k} + \log k\right)$$

choose k s.t. $\log k = \sqrt{\log U}$

$$= O(\sqrt{\log U})$$

~ Durrer '86 also gave a variant

Rmk - C.-Patrascu '06 also gave a variant
with $\boxed{O\left(\frac{\log n}{\log \log n}\right)}$ query time

(rough idea - instead of table lookup,
use bit packing tricks)

Select b segs St. $b \log k \approx w$

\Rightarrow reduce U_L or U_R by factor k
or reduce n by factor b .

$$\text{query time } O\left(\log_b U + \log_b n\right) \quad (w \nparallel \log U)$$

$$= O\left(\frac{\log^w n}{\log b} + \frac{\log n}{\log b}\right)$$

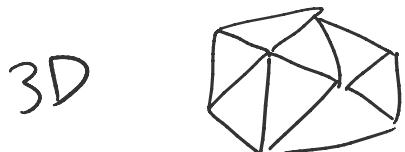
$$b = \log^\varepsilon n \quad = O\left(b + \frac{\log n}{\log b}\right) = O\left(\frac{\log n}{\log \log n}\right)$$

Rmk: can improve $O(\sqrt{\log U})$ to $\boxed{O\left(\sqrt{\frac{\log U}{\log \log U}}\right)}$

Open: better?

[in offline query case,
C.-Patrascu '07: $\boxed{O(c \sqrt{\log \log n})}$ query time
 $\leq o(\log^\varepsilon n)$]

Open Problems:



$\left\{ O(\log^2 n) \text{ query time}$
 $O(n \log n) \text{ space}$

dynamic 2D

Nekrich '21
 $O(\log n)$ query & update