Point Location in Sublog Time

Last time: orthogonal case
O(n) space, O((log log n) time

Today: non-orthogonal case
Assume coords are integers in [0, n]

C. Patrascu '06: O(n) space, O(\sqrt{\log n}) query
Suffice to solve slab subproblem

by method O,
can solve general problem
in O(\sqrt{\log n}) time, O(n^2) space
& then reduce space by
Separator or Sampling method

Idea: divide & conquer
to reduce left/right universe
Idea: divide and 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 $i=1,2,...$
$S_i =$ lowest seg s.t.
left endpt is in a higher subinterval than $S_{i-1}$ & right `...` `...` $S_i$

Recursively build DS for segs between $S_i$ & $S_{i-1}$

Either left universe $\rightarrow U/L/k$ or right universe $\rightarrow U/R/k$.

build DS for $\tilde{S}_0,\ldots,\tilde{S}_e$

segs obtained by rounding $S_0,\ldots,S_e$ downward

Query algm, given pt $q$:

→ locate $q$ in $\tilde{S}_0,\ldots,\tilde{S}_e$

⇒ can locate $q$ in $S_0,\ldots,S_e$ with 1 extra comp.

say $S_i < q < S_{i+1}$

recurse between $S_i$ & $S_{i+1}$.

Lemma: can build pt location DS for $\tilde{S}_0,\ldots,\tilde{S}_e$

with $O(1)$ query time & $O(k^2)$ space.

Pf: \[ \text{Projective Transformation} \]
Pf: 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 $\mathcal{O}(1)$ at most)

Query time

$$\mathcal{O}(u_L, u_K) \leq \left\{ \begin{array}{ll} \mathcal{O}\left(\frac{u_L}{k}, u_K\right) + \mathcal{O}(1) & \text{or} \\ \mathcal{O}\left(u_L, \frac{u_K}{k}\right) + \mathcal{O}(1) & \end{array} \right. $$

$$\Rightarrow \mathcal{O}\left(\log_k u_L + \log_k u_K\right)$$

Space $\mathcal{O}\left(\frac{k^2}{n}\right)$

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

$$\Rightarrow \text{space } \mathcal{O}(n)$$

Query time

$$\mathcal{O}\left(\log_k u + \log(k^2)\right)$$

by binary search

Choose $k$ st. $\log k = \sqrt{\log n} = \mathcal{O}(\sqrt{\log n})$

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

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

Select \( b \) segs \( s \rightarrow b \log k \approx w \)

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

Query time \( O\left(\frac{\log U}{\log n} + \frac{\log n}{\log b}\right) \)

\( = O\left(\frac{\log k}{\log n} + \frac{\log n}{\log b}\right) \)

\( v = \log n \Rightarrow O\left( b + \frac{\log n}{\log b}\right) = O\left(\frac{\log n}{\log \log n}\right) \)

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

Open: better?

[ in offline query case, C.-Patrascu '07: \( O\left(\sqrt{\log \log n}\right) \) query time \( \leq O\left(\log n \right) \) ]

Open Problems:

3D \hspace{1cm} \{ \begin{align*} \text{O}(\log n) \text{ query time} \\
\text{O}(|U| \log n) \text{ space} \end{align*} \}

dynamic 2D \hspace{1cm} \text{Nekrich '21} \\
\text{O}(\log n) \text{ query & update}