Planar Point Location

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

equiv: find line segment immediately above q

in \( \mathcal{A} \),

\[ \begin{align*}
O(n) & \text{ space} \\
O(\log n) & \text{ time} \\
O(n \log n) & \text{ preprocess time}
\end{align*} \]

> 8 different methods

Method 0: “Slab”

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

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

space \( O(n^2) \)
preprocess time \( O(n^2 \log n) \)

Method 1: Segment Tree

given \( n \) segs intersecting vertical slab \( \sigma \),
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 subslabs.

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

short else

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

each seg is stored \( \leq 2 \log n \) times

⇒ space \( O(n \log n) \)

preproc time \( O(n \log^2 n) \)

("dual" to range tree)

Query algm: \( O(\log n) \) nodes
Query algo:

O(log\(n\)) nodes

binary search in each slab containing \(q\)

\(\Rightarrow O(\log^2 n)\) 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

\(b=3\)

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

\(L^+(u) = \) "non-sorted" 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 \( b \)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 \( O(1) \) time
Query time \( \mathcal{O}(\log n + (\log n) \cdot O(1)) = \mathcal{O}(\log n) \)

List binary search at leaf

Space \( \mathcal{O}\left( \sum_u |L(u)| \left( 1 + \frac{a}{b} + \left( \frac{a}{b} \right)^2 + \left( \frac{a}{b} \right)^3 + \ldots \right) \right) \)
\[ = \mathcal{O}\left( \sum_u |L(u)| \right) \quad \text{for } b > 2 \]
\[ = \mathcal{O}(n \log n) \]

(can reduce space to \( \mathcal{O}(n) \) with extra ideas...)
Method 3: Trapezoid Tree (Prattarata '81)

Use a tree where cells are trapezoids instead of slabs.

Given \( n \) segs intersecting trapezoid \( T \),

If no long segs,

- divide \( T \) by median \( x \)

Else

- divide \( T \) by all long segs

\( \Rightarrow \) \( O(n \log n) \) space as before

Query time \( O(\log n \cdot \log n) = O(\log^2 n) \)

\# levels

Binary search at each multi-degree node

To be cont'd ...