Planar Point Location in Sublogarithmic Time

Mihai Pătraşcu (MIT)

Point Location in o(log n) Time, Voronoi Diagrams in o(n log n) Time, and Other Transdichotomous Results in Computational Geometry

Timothy M. Chan (U. Waterloo)

Main Theme

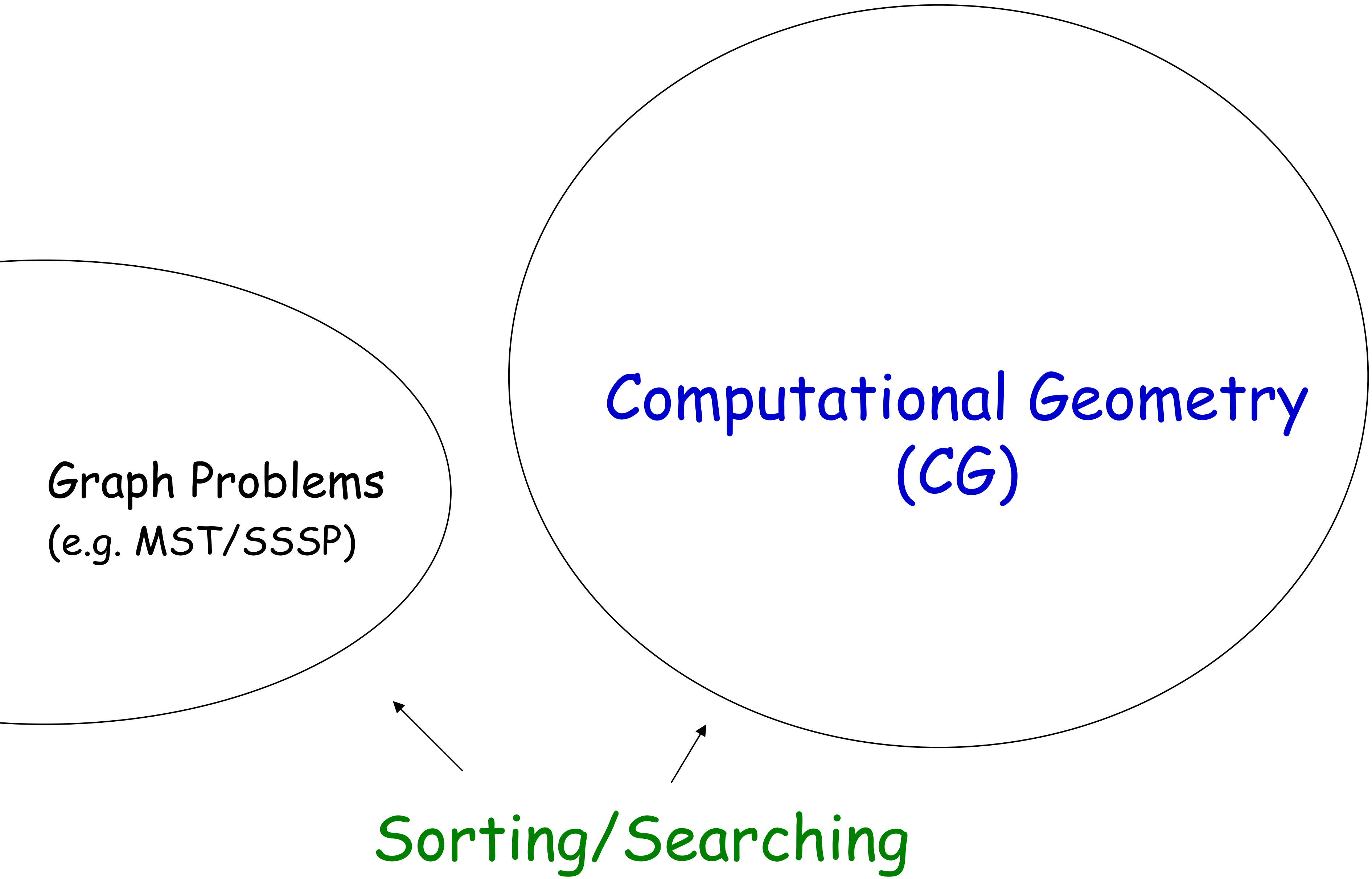
Power of the Word RAM !

Integer Sorting

• Radix-sort <'54	$O(n \log U/\log n)$
• Van Emde Boas '77	$O(n \log\log U)$
• Kirkpatrick, Reisch'84	$O(n \log(\log U/\log n))$
• Fredman, Willard (FOCS'90)	$O(n \log n/\log\log n)$ (det.)
	$O(n\sqrt{\log n})$ (rand.)
• Andersson et al. (STOC'95)	$O(n \log\log n)$ (rand.)
	$O(n)$ for $w \gg \log^2 n$ (rand.)
• Raman (ESA'96)	$O(n\sqrt{\log n \log\log n})$ (det.)
• Andersson (FOCS'96)	$O(n\sqrt{\log n})$ (det.)
• Thorup (SODA'98)	$O(n (\log\log n)^2)$ (det.)
• Han (SODA'01)	$O(n \log\log n \log\log\log n)$ (det.)
• Han (STOC'02)	$O(n \log\log n)$ (det.)
• Han, Thorup (FOCS'02)	$O(n\sqrt{\log\log n})$ (rand.)

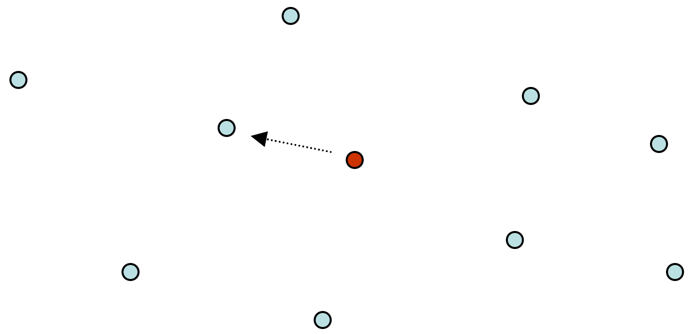
Integer (Predecessor) Searching

- Van Emde Boas '77 ("stratified trees") — $O(\log\log U)$ query time
- Fredman, Willard (FOCS'90) ("fusion trees") — $O(\log n/\log\log n)$, $O(\sqrt{\log n})$
- Beame, Fich (STOC'99) (optimal in cell probe model) — $O(\log\log U/\log\log\log U)$, $O(\sqrt{\log n/\log\log n})$
- Andersson, Thorup (STOC'00) ("exponential search trees") — $O(\sqrt{\log n/\log\log n})$ for query & update
- Pătraşcu, Thorup (STOC'06)
- Etc.


Graph Problems
(e.g. MST/SSSP)
Computational Geometry
(CG)
Sorting/Searching

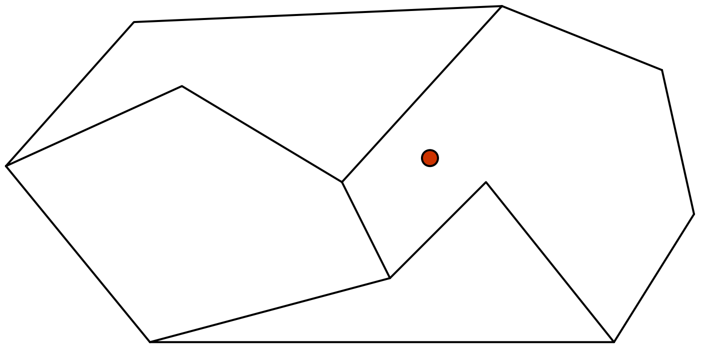
Standard Problems in CG

- 2D nearest neighbor search



O(n log n) preproc.
O(n) space
O(log n) query

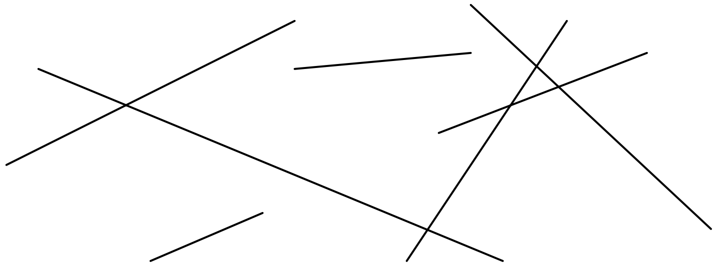
- 2D point location



O(n) preproc.
O(n) space
O(log n) query

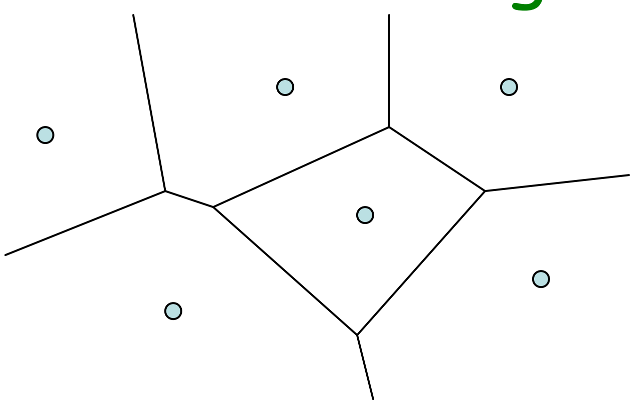
Standard Problems in CG (Cont'd)

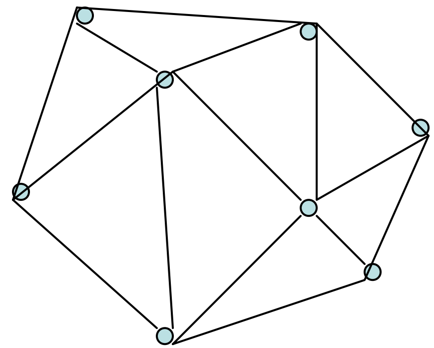
- 2D line segment intersection



O(n log n + K) time

- 2D Voronoi diagrams & 3D convex hulls





O(n log n) time

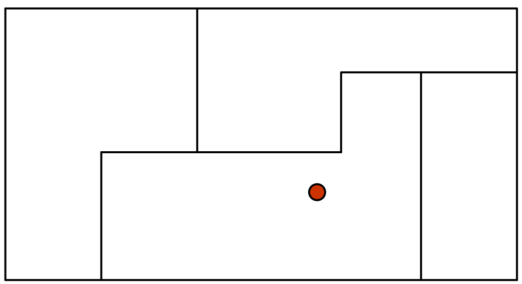
- Etc, etc, etc.

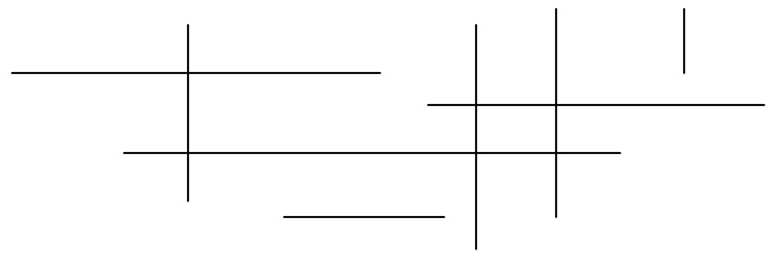
The Model

- Unit-cost RAM with word size w
- Coordinates are integers in {1,…,U}
- $U \le 2^w$, i.e., $w \ge \log U$
- $w \ge \log n$
- Availability of standard ops like <, +, -, *, /, bitwise-&, <<, >>

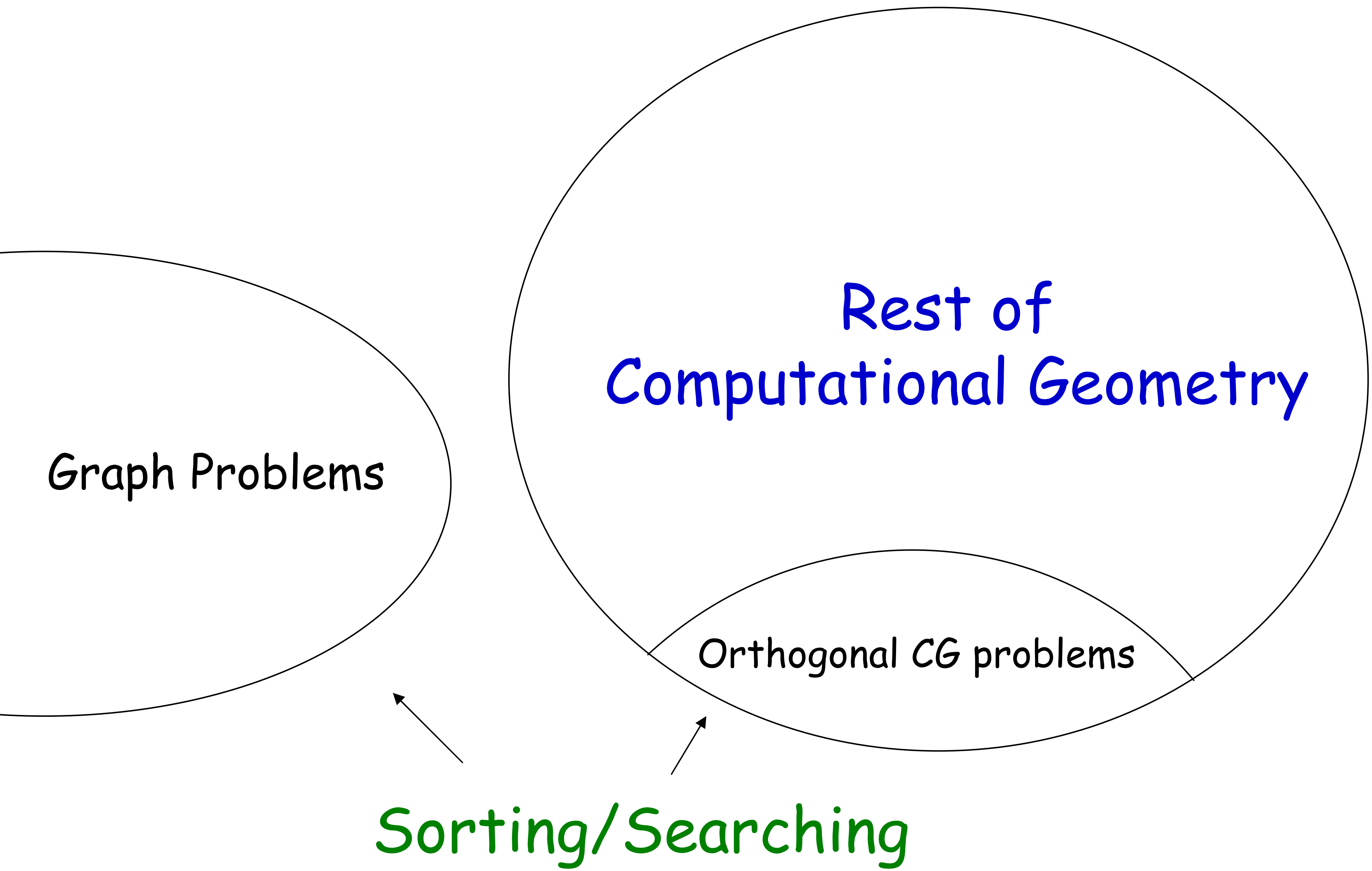
Previous Word RAM Results in CG

- Orthogonal range searching
- Orthogonal cases of 2D point location & segment intersection [e.g. loglog U-type results by Overmars '87]





- L_∞ variants of 2D nearest neighbor/Voronoi diagrams [e.g., loglog U-type results by Karlsson '84]
- NON-orthogonal problems???
  e.g., Willard (SODA'92) asked: standard 2D Voronoi diagrams in o(n log n) time??


Rest of
Computational Geometry
Graph Problems
Orthogonal CG problems
Sorting/Searching

New Results

- 2D Voronoi diagrams
  $O(n \log n / \log \log n)$ time (rand.)
- 3D convex hulls
  $O(n \log n / \log \log n)$ time (rand.)
- 2D line segment intersection
  $O(n \log n / \log \log n + K)$ time (rand.)
- 2D point location & 2D nearest neighbor
  $O(n)$ space, $O(\log n / \log \log n)$ query
- Etc, etc, etc.

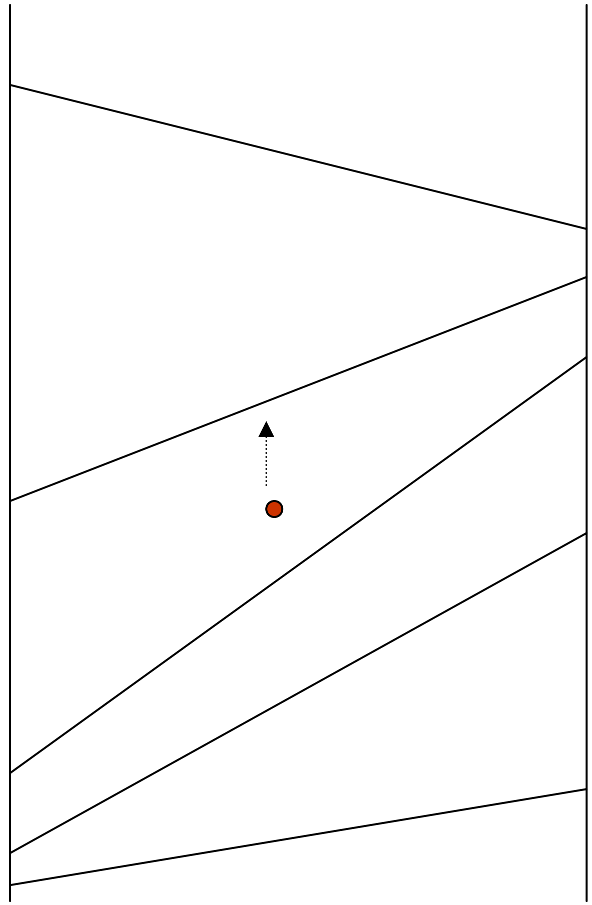
New Results (Cont'd)

- 2D Voronoi diagrams
  $O(n\sqrt{\log U / \log\log U})$ time (rand.)
- 3D convex hulls
  $O(n\sqrt{\log U / \log\log U})$ time (rand.)
- 2D line segment intersection
  $O(n\sqrt{\log U / \log\log U} + K)$ time (rand.)
- 2D point location & 2D nearest neighbor
  $O(n)$ space, $O(\sqrt{\log U / \log\log U})$ query
- Etc, etc, etc.

Key Subproblem: Point Location in a Slab

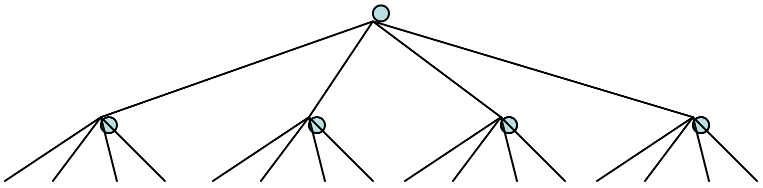
- Given n disjoint line segments spanning vertical slab, how to locate query point in o(log n) time??

  [can't project to 1D, can't build 1D structure at every vertical line, …]



Basic Idea

- Replace binary tree with b-ary tree



- But how?
  [Fredman, Willard's original "fusion tree" does not generalize…]

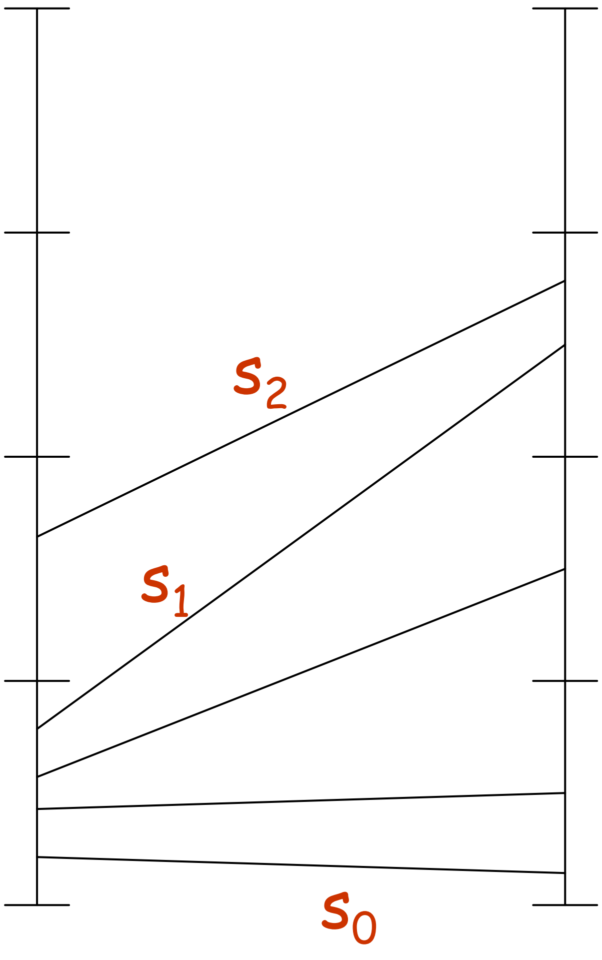
Key Observation

- Fix b, h. Given n segments with left/right endpts in a length-2^ℓ / length-2^m vertical interval, we can find ≤ b segments $s_0, s_1, \ldots$ s.t.
  i. Between s_i & s_{i+1}: there are ≤ n/b segments, OR left endpts are in a length-$2^{\ell-h}$ subinterval, OR right endpts are in a length-2^{m-h} subinterval;
  ii. $\exists$ segments $s_0', s_2', \ldots$, each encodable in O(h) bits, with $s_0 < s_0' < s_2 < s_2' < \ldots$

Proof

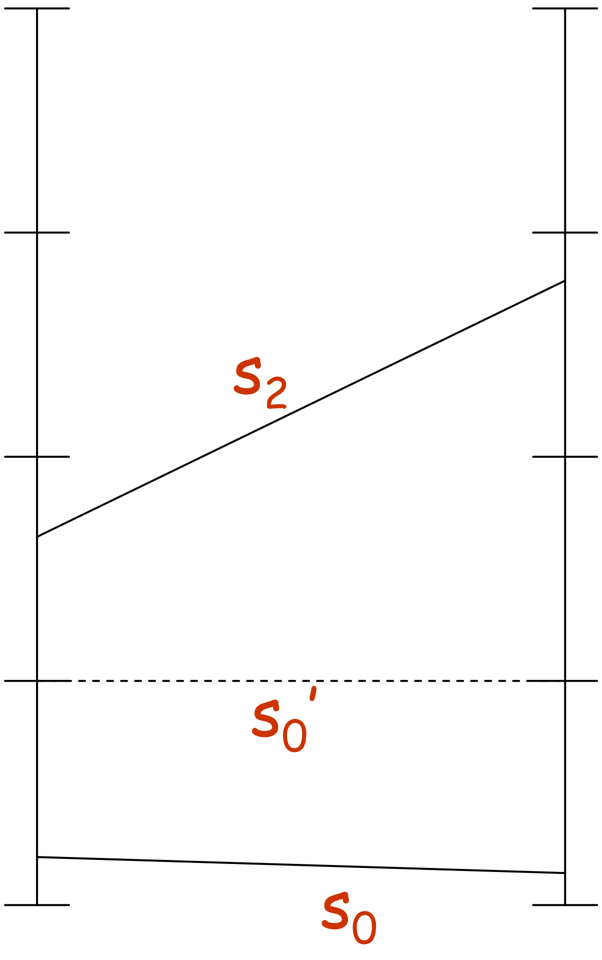
- Divide left/right interval into 2^h grid subintervals
- Draw $(n/b)^{th}$, $(2n/b)^{th}$, $(3n/b)^{th}$,... segment

i. Betwn s_0 & s_1, left endpts are in length-$2^{\ell-h}$ subinterval;
Betwn s_1 & s_2, there are n/b segments



Proof

- $\exists$ s_0' betwn s_0 & s_2 using only grid endpts ($O(h)$ bits)



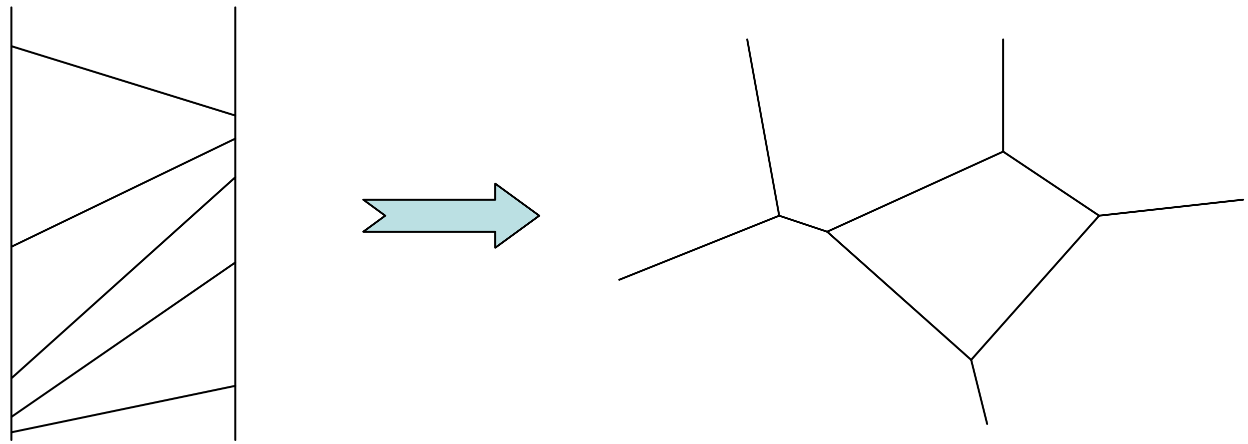
The New "2D Fusion Tree" for the Slab Subproblem

- Just apply Observation recursively!
- Each tree node can be packed in one word if $bh \approx w$ (by ii)
- Query time $=$ tree height (using word ops)

$$\leq \log_b n + 2w/h \quad \text{(by i)}$$
$$= O(\log_b n + b) \quad (\text{set } b = \log^{\varepsilon} n)$$
$$= \boxed{O(\log n / \log\log n)}$$

Switch speakers. Confuse audience

Using the Slab Problem



- talk only about 2D point location
  - a bit more smartness involved for 2D Voronoi, 3D convex hull, line segment intersection…
- before our work:
  - the slab problem is a trivial binary search
  - going to the general case is what’s interesting

Conversion Techniques

Lipton+Tarjan — planar separator

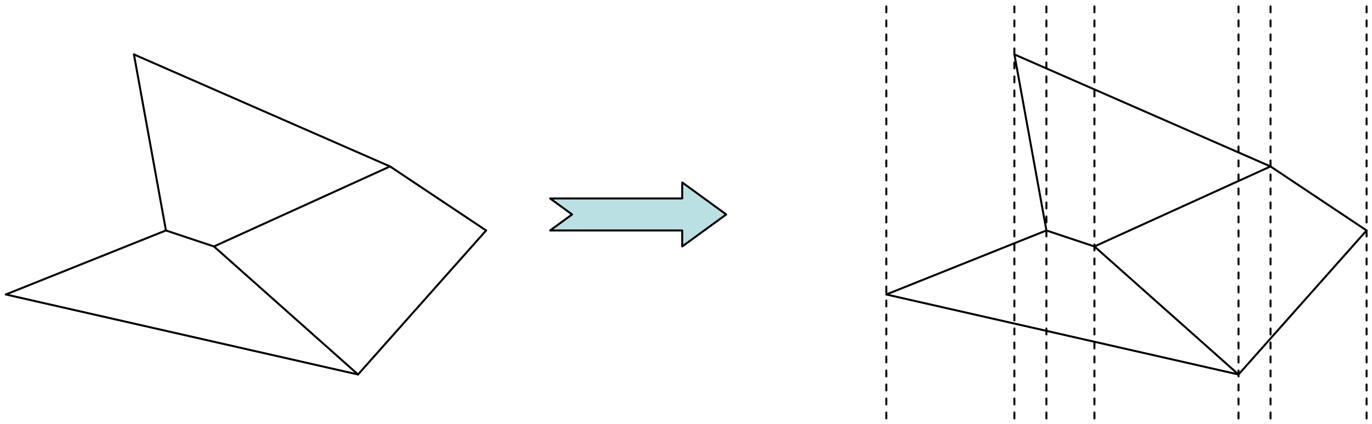
Mulmuley — random sampling

Cole, Sarnak+Tarjan — persistence

- use exponential trees [Andersson] + new ideas

- Kirkpatrick — triangulation refinement
- Edelsbrunner+Guibas+Stolfi — separating chains

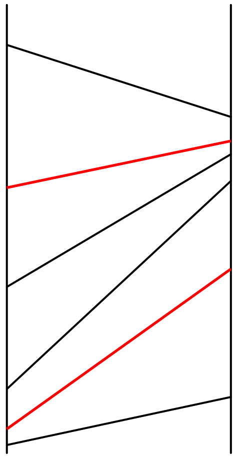
Review: Persistence



- sweep with vertical line
- insert/remove segments into dynamic slab structure

  (next slides)
- keep all past images of the data structure in small space via persistence (can be done)

Review: Exponential Trees



Hope: dynamize slab structure black-box

pick $\sqrt{n}$ separators, $\Theta(\sqrt{n})$ segments apart

construction: static structure on separators; recurse

update: rebuild static structure when needed

separators change infrequently (?)

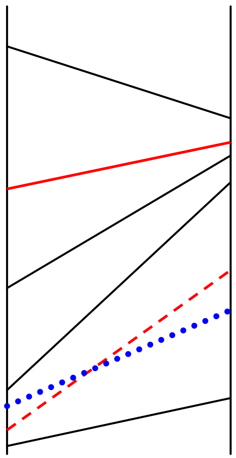
query: $$\frac{\lg \sqrt{n}}{\lg \lg \sqrt{n}} + \frac{\lg \sqrt[4]{n}}{\lg \lg \sqrt[4]{n}} + \ldots \le \frac{\lg n}{\lg \lg n} \cdot (\tfrac{1}{2} + \tfrac{1}{4} + \ldots)$$

2D ≠ 1D

Trouble: segments are not numbers

- remove separator segment
- insert intersecting segment

⇨ separator does not separate



Use offline knowledge:

separators = segments which won't be removed for a long time

Open problems

- <u>count</u> segment intersections in $o(n \log n)$
- derandomize (3D convex hull, 2D Voronoi)
- better algorithms
  - point location problem is offline
- lower bounds for the data structures
  - is exact NN easier than point location?

Thank you!