Dynamic Data Structures for Computational Geometry

Yakov Nekrich[1]

[1]Michigan Technological University

March 17, 2023

Research Interests

- Computational Geometry
  algorithms and data structures on geometric objects
- Strings
  Sequences of Symbols
- Compressed Data Structures
  Data Compression + Data Structures

(Orthogonal) Range Reporting

The Problem

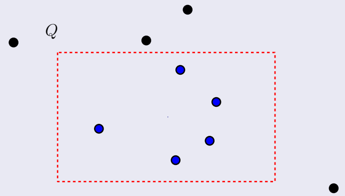
Data Structure: set of d-dimensional points S

Query: axis-parallel rectangle Q

Answer: report all points in $Q \cap S$

$$Q = [x_1^L, x_1^R] \times [x_2^L, x_2^R] \times \cdots \times [x_d^L, x_d^R]$$

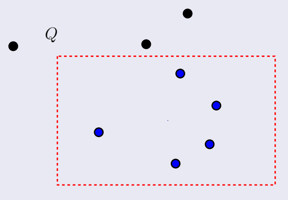
Example (2-D)



Motivation

- Data bases
  report all employees with salary between 80K and 100K and age between 35 and 50
- String algorithms
- Range reporting is a variant of range searching
  other: range counting, color queries, maxima queries, etc.

Example (2-D)



(Orthogonal) Range Emptiness

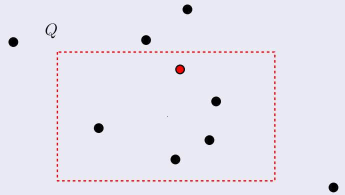
The Problem

Data Structure: set of d-dimensional points S

Query: axis-parallel rectangle Q

Answer: is $Q \cap S = \emptyset$?

Example (2-D)



(Orthogonal) Range Emptiness

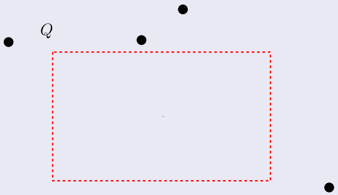
The Problem

Data Structure: set of d-dimensional points S

Query: axis-parallel rectangle Q

Answer: is $Q \cap S = \emptyset$?

Example (2-D)



Multi-Dimensional Range Reporting

$O(n\texttt{polylog}(n))$-space data structure

- 2-D range reporting:
  $O(\log\log n + k)$ query time
  Alstrup,Brodal,Rauhe; FOCS'00
- 3-D range reporting:
  $O(\log\log n + k)$ query time
  Chan TALG'13
- Optimal query time for 2-D and 3-D
  follows from the predecessor lower bound
  Pătrașcu, Thorup' 06

n - number of points in the data structure

k - number of points in the answer

Multi-Dimensional Range Reporting

4-D Reporting

$O(\log n + k)$ query time
Chan TALG'13, SODA'11

Lower Bound

$\Omega(\log n / \log\log n)$ query time for any $O(n\texttt{polylog}(n))$-space data structure
Pătrașcu, 2011

Multi-Dimensional Range Reporting

4-D Reporting: Optimal Time

$O(n\mathtt{polylog}(n))$ space
$O(\log n / \log\log n + k)$ time
Nekrich, SODA'21
Optimal time by Pătrașcu, 2011

Multi-Dimensional Range Reporting

$d \geq 4$ dimensions

$O(n \log^{d-2+\varepsilon} n)$ space
$O((\log n / \log\log n)^{d-3} + k)$ query time
Nekrich, SODA'21

n - number of points in the data structure
k - number of points in the answer
any constant $\varepsilon > 0$

Model of Computation

Pointer Machine Model (PM)
Tarjan '79

- no random acess to memory cells
  each memory cell can be accessed through a series of pointers only
- Informally: model of computation in which the use of arrays is not allowed
  Example: Binary search trees, but no van Emde Boas

Previous Results

PM Model

- 2-D: $\boldsymbol{O(\log n + k)}$ time
  Bentley IPL'79
- 3-D: $\boldsymbol{O(\log n + k)}$ time
  Chazelle, Edelsbrunner Discrete & Comp. Geometry '87

Previous Results

PM Model, 4-D

- $O(\log^2 n + k)$ time
  using range trees *Bentley IPL'79*
- $O((\log n / \log\log n)^2 + k)$ time
  Afshani, Arge, Larsen FOCS'09
- $O(\log^{3/2} n + k)$ time
  Afshani, Arge, Larsen SoCG'12
- Open question: $O(\log n)$ time?

Our Result

PM Model, 4-D Range Reporting

Nekrich, Rahul SODA'23

- $O(\log n \log\log n + k)$ query time
- space
  - $O(n \log^4 n)$ space for general 4-D queries
  - $O(n \log n)$ space when the query range is bounded on 4 sides (dominance queries) or 5 sides (5-sided queries)

Dominance queries:

report all p. s.t., $p.x \le a$, $p.y \le b$, $p.z \le c$, $p.z' \le d$

General Queries:

report all p. s.t., $a_1 \le p.x \le a_2$, $b_1 \le p.y \le b_2$, $c_1 \le p.z \le c_2$, $d_1 \le p.z' \le d_2$

Our Result

PM Model, d-dimensional range reporting

- PM Model:
  $O(\log^{d-3} n \log\log n + k)$ time
  $O(n \log^{d} n)$ space
  Nekrich, Rahul SODA'23

General Approach

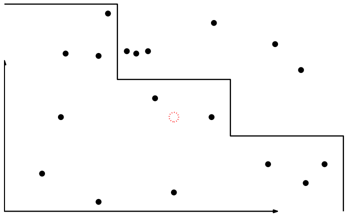
Data structure for queries bounded on 4 sides (dominance queries)

- 3-D Dominance Reporting Queries
  (query range bounded on 3 sides: $\boldsymbol{p.x} \leq \boldsymbol{a}$, $\boldsymbol{p.y} \leq \boldsymbol{b}$, $\boldsymbol{p.z} \leq \boldsymbol{c}$)
- Range Trees
  extends to 4-D

Shallow Cuttings

2-D shallow cuttings

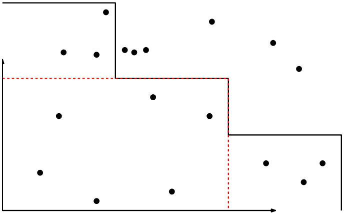
- Each point below the staircase dominates $\leq 2t$ points from S
- Each point that dominates $\leq t$ points from S is below the "staircase"
- A point q dominates q' iff $q.x \geq q'.x$ and $q.y \geq q'.y$



Shallow Cuttings

2-D shallow cuttings

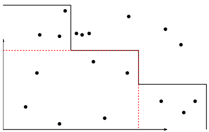
- A t-shallow cutting for a set S consists of $O(\frac{n}{t})$ cells each cell contains $\leq 2t$ points from S
- every planar point that dominates $\leq t$ points is contained in some cell C



Shallow Cuttings

shallow cuttings approach:

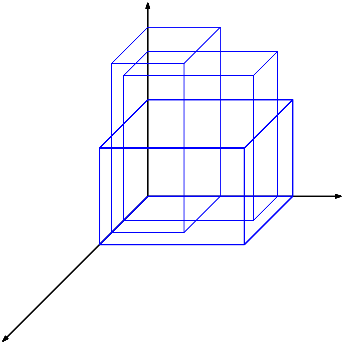
- set $t = \log n$
  keep a separate data structure for each cell
- Queries:
  Find the *cell* C that contains the query point q
  Using the data structure for C, report points dominated by q in $O(\log t + k) = O(\log \log n + k)$ time



3-D Shallow Cuttings

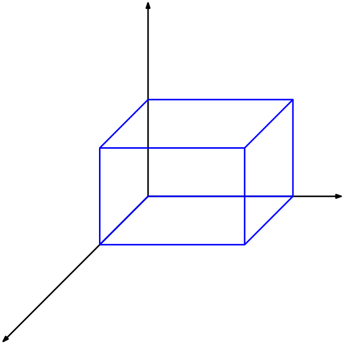
- A t-shallow cutting for a set S consists of $O(\frac{n}{t})$ cells (3-D boxes)
  each cell contains $\leq 2t$ points from S
- every 3-D point that dominates $\leq t$ points is contained in some cell C

Afshani'08



3-D Shallow Cuttings

- A t-shallow cutting for a set S consists of $O(\frac{n}{t})$ cells (3-D boxes)
  each cell contains $\leq 2t$ points from S
- every 3-D point that dominates $\leq t$ points is contained in some cell C

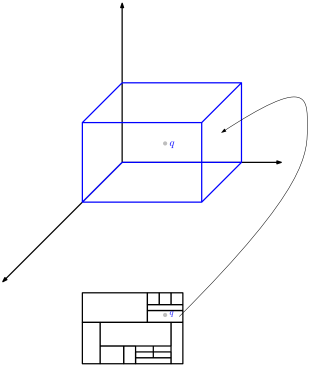


Afshani'08

Shallow Cuttings

shallow cuttings approach:
Bottleneck:

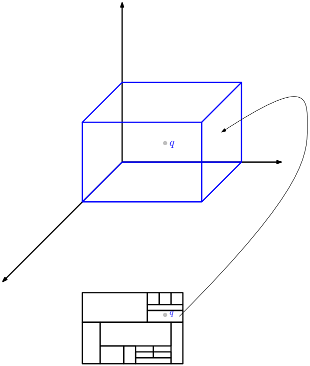
- Find the cell C that contains the query point q
- $O(\log n)$ time in the PM model



Shallow Cuttings

shallow cuttings approach:

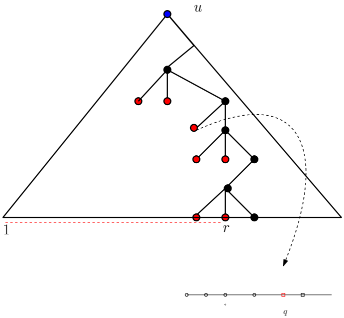
- set $t = \log n$
  keep a separate data structure for each cell
- Queries:
  Find the *cell* C that contains the query point q
  Using the data structure for C, report points dominated by q in $O(\log t + k) = O(\log\log n + k)$ time



Range Tree

Range Tree

- Fractional cascading?
  Chazelle, Guibas '86
  $\boldsymbol{O(1)}$ time per node in **one dimension**

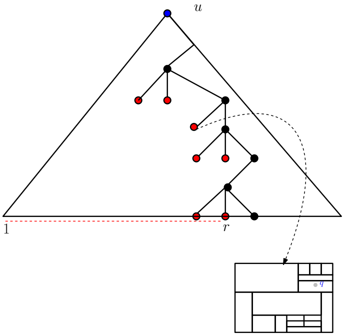


Range Tree

Range Tree

- 2-D fractional cascading: orthogonal point location in each node on a path
  - lower bound for general graphs
    Chazelle, Liu STOC'01
  - can be circumvented in the case of a tree
    Afshani, Cheng FOCS'12

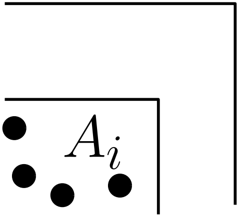
    $O(\log^{3/2} n + k)$ query time
    for this problem
  - this bound is tight!
    Afshani, Cheng FOCS'12



Property of shallow cuttings:

Lemma

Let $\mathcal{A}$ be an t-shallow cutting for a set S and let $\mathcal{B}$ be an $(2t)$-shallow cutting for a set $S' \subseteq S$. Every cell A_i of $\mathcal{A}$ is contained in some cell B_j of $\mathcal{B}$.

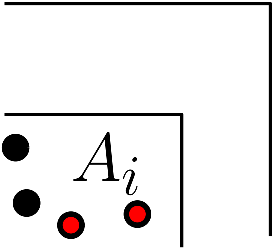


4-D Range Reporting

Property of shallow cuttings:

Lemma

Let $\mathcal{A}$ be an t-shallow cutting for a set S and let $\mathcal{B}$ be an $(2t)$-shallow cutting for a set $S' \subseteq S$. Every cell A_i of $\mathcal{A}$ is contained in some cell B_j of $\mathcal{B}$.

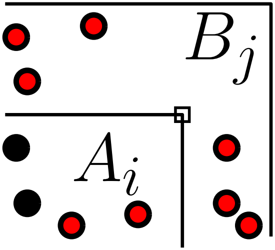


4-D Range Reporting

Property of shallow cuttings:

Lemma

Let $\mathcal{A}$ be an t-shallow cutting for a set S and let $\mathcal{B}$ be an $(2t)$-shallow cutting for a set $S' \subseteq S$. Every cell A_i of $\mathcal{A}$ is contained in some cell B_j of $\mathcal{B}$.

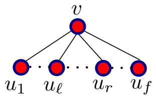


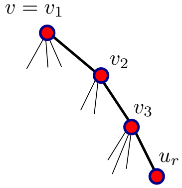
Our Approach

- Hierarchy of Range Trees
- Each internal node in T_i has $n^{(1/3)^i}$ children
- Height (number of levels) of T_i is 3^i: trees T_1, T_2, $\ldots$ have height 3, 9, 27, $\ldots$
- $\log_3 \log n$ trees
  last tree T_h has a constant node degree

Our Approach

- Create a shallow cutting for each group of sibling nodes in T_i
- Iteration i: locate a 3-D point q in 3^i shallow cuttings for $i = 1, 2, 3, \ldots$
- use relationship between shallow cuttings and spend $O(\log n)$ time for each T_i
- Total query time $O(\log n \log\log n + k)$





Example: group of siblings $u_1, \ldots, u_r$ with parent v in T_i is represented by three groups of siblings with parents v_1, v_2, v_3 in T_{i+1}

Summary

Our Result, RAM

$O(\log n / \log\log n + k)$ query time $O(n\text{polylog}(n))$ space
Nekrich, SODA'21

Our Result, PM Model

$O(\log n \log\log n + k)$ query time
$O(n\text{polylog}(n))$ space
Nekrich, Rahul SODA'23

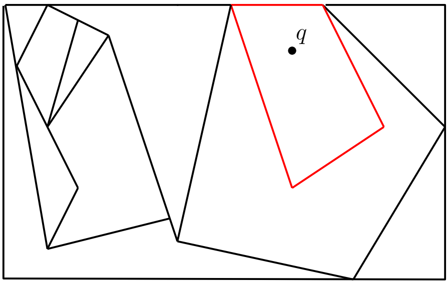
Summary

	RAM	PM
2-D & 3-D	$O(\log\log n + k)$	$O(\log n + k)$
4-D	$O(\log n/\log\log n + k)$	$O(\log n \log\log n + k)$

Open Questions

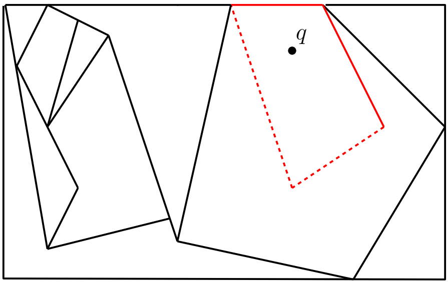
- Closing the gap between 3-D and 4-D in the PM model: $O(\log n + k)$ query time for 4-D range reporting?
- Query times in $d > 4$ dimensions, RAM and PM? Is $\sim O(\log n)$ for each further dimension $d > 4$ necessary?

Geometry: 2-D Dynamic Point Location



Segments can be deleted, new segments can be inserted

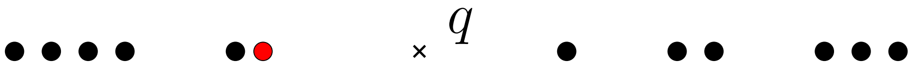
Geometry: 2-D Dynamic Point Location



Segments can be deleted, new segments can be inserted

Geometry: 2-D Dynamic Point Location

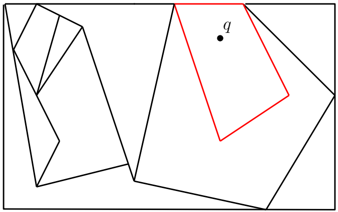
- Natural generalization of one-dimensional predecessor search
- $\boldsymbol{\Omega(\log n)}$ query time
  $\boldsymbol{\Omega(\log n)}$ update time
- Static point location: $\boldsymbol{O(\log n)}$ query time
  Sarnak, Tarjan '86
- Dynamic point location? more challenging



Geometry: 2-D Dynamic Point Location

Open Problem

Dynamic Point Location data structure with
$O(\log n)$ query time and $O(\log n)$ update time
using $O(n)$ space?
open question since 90s
asked in e.g., *Chazelle, 1991; Chazelle 1994; Chiang, Tamassia 1992; Snoeyink 2004*



Geometry: 2-D Dynamic Point Location

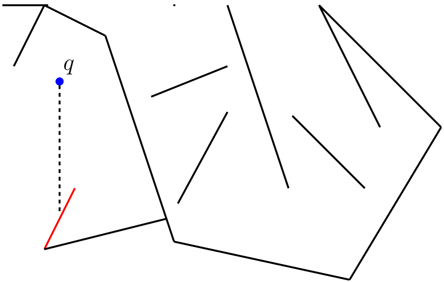
Reference	Space	Query Time	Insertion Time	Deletion Time	
Bentley, 1977	$n\log n$	$\log^2 n$	$\log^2 n$	$\log^2 n$	
Cheng–Janardan, 1992	n	$\log^2 n$	$\log n$	$\log n$	
Baumgarten et al., 1994	n	$\log n \log\log n$	$\log n \log\log n$	$\log^2 n$	†
Arge et al., 2006	n	$\log n$	$\log^{1+\varepsilon} n$	$\log^{2+\varepsilon} n$	†
Arge et al., 2006	n	$\log n$	$\log n(\log\log n)^{1+\varepsilon}$	$\log^2 n/\log\log n$	†‡
Chan and Nekrich, 2015	n	$\log n(\log\log n)^2$	$\log n \log\log n$	$\log n \log\log n$	
Chan and Nekrich, 2015	n	$\log n$	$\log^{1+\varepsilon} n$	$\log^{1+\varepsilon} n$	
Chan and Nekrich, 2015	n	$\log n$	$\log^{1+\varepsilon} n$	$\log n(\log\log n)^{1+\varepsilon}$	
Chan and Nekrich, 2015	n	$\log^{1+\varepsilon} n$	$\log n$	$\log n$	
Chan and Nekrich, 2015	n	$\log n \log\log n$	$\log n \log\log n$	$\log n \log\log n$	‡
Nekrich, 2021	n	$\log n$	$\log n$	$\log n$	

† - amortization, ‡ - randomization (Las-Vegas)

not included: special cases of dynamic point location (*Fries'90, Preparata, Tamassia '89; Chiang, Tamassia '92; Chiang, Preparata, Tamassia '96; Goodrich, Tamassia '98; Gioyra, Kaplan '09; Nekrich'10; Chan, Tsakalidis '18*)

Geometry: 2-D Dynamic Point Location

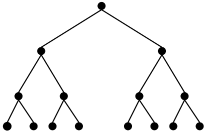
Vertical Ray Shooting



Given: a set of n non-intersecting segments
Query: for any point q find the segment immediately below q
(first segment hit by a downward vertical ray from q)

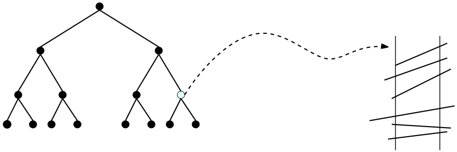
Geometry: 2-D Dynamic Point Location

- Base structure: segment tree
- Associate a vertical slab with every node
  Keep list of segments spanning the vertical slab in each node
- Segments in a node must be ordered



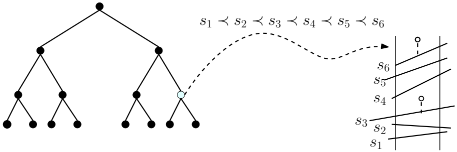
Geometry: 2-D Dynamic Point Location

- Base structure: segment tree
- Associate a vertical slab with every node
  Keep list of segments spanning the vertical slab in each node
- Segments in a node must be ordered



Geometry: 2-D Dynamic Point Location

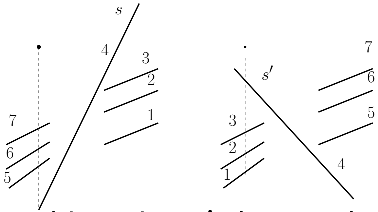
- Base structure: segment tree
- Associate a vertical slab with every node
  Keep list of segments spanning the vertical slab in each node
- Segments in a node must be ordered



Geometry: 2-D Dynamic Point Location

Challenge: Order of Segments

No dynamic ordering of segments with respect to ray shooting queries (any vertical ray must hit a larger segment before a smaller segment)

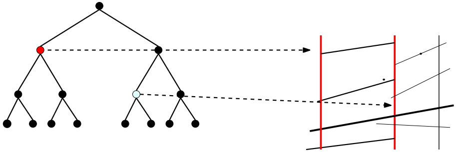


Example: Deleting s and inserting s' changes the order of all segments

Geometry: 2-D Dynamic Point Location

Challenge: no global order of segments!

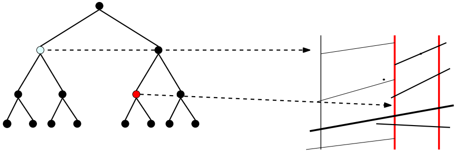
- Each segment is stored in $\sim \textbf{log}\, \boldsymbol{n}$ nodes
- We must visit $\sim O(\textbf{log}\, \boldsymbol{n})$ nodes to answer a vertical ray shooting query
- $O(\textbf{log}^2\, \boldsymbol{n})$ update time $\sim O(\textbf{log}^2\, \boldsymbol{n})$ query time ($\sim \textbf{log}\, \boldsymbol{n}$ independent binary searches)



Geometry: 2-D Dynamic Point Location

Challenge: no global order of segments!

- Each segment is stored in $\sim \log n$ nodes
- We must visit $\sim O(\log n)$ nodes to answer a vertical ray shooting query
- $O(\log^2 n)$ update time $\sim O(\log^2 n)$ query time ($\sim \log n$ independent binary searches)



Geometry: 2-D Point Location

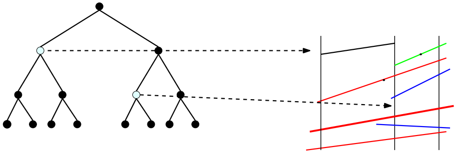
Reference	Space	Query Time	Insertion Time	Deletion Time	
Bentley, 1977	$n \log n$	$\log^2 n$	$\log^2 n$	$\log^2 n$	
Cheng–Janardan, 1992	n	$\log^2 n$	$\log n$	$\log n$	
Baumgarten et al., 1994	n	$\log n \log\log n$	$\log n \log\log n$	$\log^2 n$	†
Arge et al., 2006	n	$\log n$	$\log^{1+\varepsilon} n$	$\log^{2+\varepsilon} n$	†
Arge et al., 2006	n	$\log n$	$\log n(\log\log n)^{1+\varepsilon}$	$\log^2 n/\log\log n$	†‡
Chan and Nekrich, 2015	n	$\log n(\log\log n)^2$	$\log n \log\log n$	$\log n \log\log n$	
Chan and Nekrich, 2015	n	$\log n$	$\log^{1+\varepsilon} n$	$\log^{1+\varepsilon} n$	
Chan and Nekrich, 2015	n	$\log n$	$\log^{1+\varepsilon} n$	$\log n(\log\log n)^{1+\varepsilon}$	
Chan and Nekrich, 2015	n	$\log^{1+\varepsilon} n$	$\log n$	$\log n$	
Chan and Nekrich, 2015	n	$\log n \log\log n$	$\log n \log\log n$	$\log n \log\log n$	‡

Fractional cascading can be used to improve the query time
Baumgarten et al. '94; Arge et al. '06

Geometry: 2-D Dynamic Point Location

Segment Categorization:
Chan, Nekrich FOCS'15

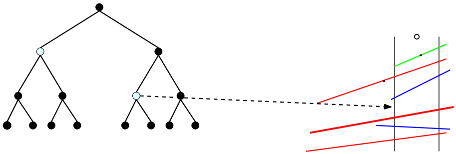
- segments are assigned different colors (categories)
- segments of the same color are ordered, even if stored in different nodes
- segments in the same node can be assigned different colors:



Geometry: 2-D Dynamic Point Location

Segment Categorization:
Chan, Nekrich FOCS'15

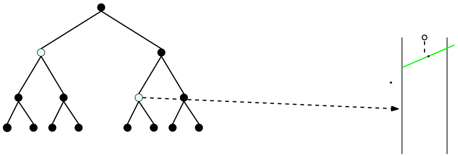
- segments in the same node can be assigned different colors
- trade-offs between update and query time (number of colors per node vs number of colors assigned to a segment)



Geometry: 2-D Dynamic Point Location

Segment Categorization:
Chan, Nekrich FOCS'15

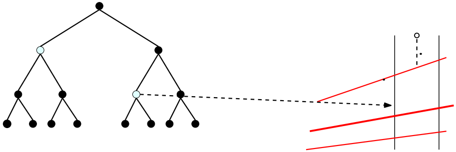
- segments in the same node can be assigned different colors
- trade-offs between update and query time (number of colors per node vs number of colors assigned to a segment)



Geometry: 2-D Dynamic Point Location

Segment Categorization:
Chan, Nekrich FOCS'15

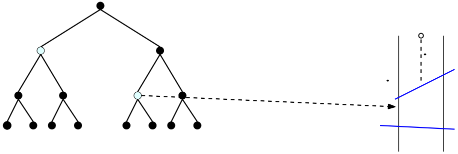
- segments in the same node can be assigned different colors
- trade-offs between update and query time (number of colors per node vs number of colors assigned to a segment)



Geometry: 2-D Dynamic Point Location

Segment Categorization:
Chan, Nekrich FOCS'15

- segments in the same node can be assigned different colors
- trade-offs between update and query time (number of colors per node vs number of colors assigned to a segment)



Geometry: 2-D Dynamic Point Location

Previous best result

$O(\log n \log\log n)$ query time (randomized)
$O(\log n \log\log n)$ update time
OR
$O(\log^{1+\varepsilon} n)$ query time
$O(\log n)$ update time
OR
$O(\log n)$ query time
$O(\log^{1+\varepsilon} n)$ update time
Chan, Nekrich, FOCS'15
all trade-offs use $O(n)$ space

can't obtain $O(\log n)$ time for updates and queries with this approach

Geometry: 2-D Dynamic Point Location

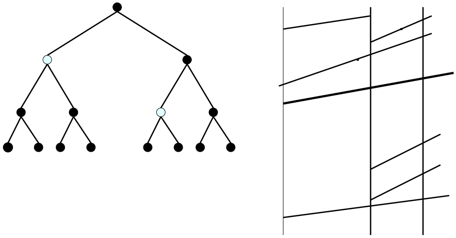
Optimal Result

$O(\log n)$ query time
$O(\log n)$ update time
$O(n)$ space
Nekrich, STOC'21

Geometry: 2-D Dynamic Point Location

Main ideas:

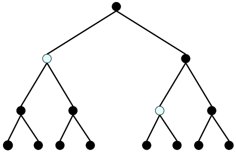
- Some segments can be inserted quickly into a slab ("good" segments)
- "bad" segments cannot be inserted quickly
- "bad" segments are stored in selected nodes only
  even in selected nodes, we don't know the total order of all bad segments

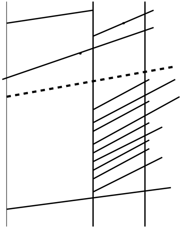


Geometry: 2-D Dynamic Point Location

Main ideas:

- Some segments can be inserted quickly into a slab ("good" segments)
- "bad" segments cannot be inserted quickly
- "bad" segments are stored in selected nodes only
  even in selected nodes, we don't know the total order of all bad segments

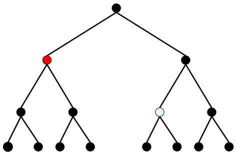


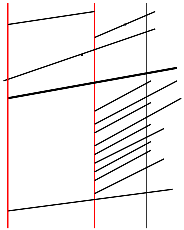


Geometry: 2-D Dynamic Point Location

Main ideas:

- Some segments can be inserted quickly into a slab ("good" segments)
- "bad" segments cannot be inserted quickly
- "bad" segments are stored in selected nodes only
  even in selected nodes, we don't know the total order of all bad segments

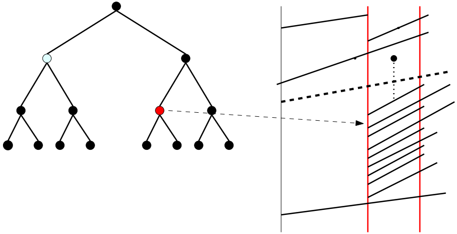




Geometry: 2-D Dynamic Point Location

Main ideas:

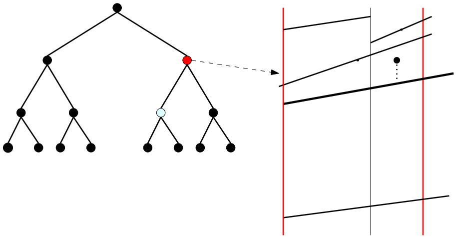
- When a query is answered in a node u, "bad" segment can be missed
- "Missed" segments are processed when the selected ancestor node is visited



Geometry: 2-D Dynamic Point Location

Main ideas:

- When a query is answered in a node u, "bad" segment can be missed
- "Missed" segments are processed when the selected ancestor node is visited



Open Question

- Point Location in 3-D:
  $O(n)$ space and $O(\log n)$ query time?
- Currently best result:
  $O(n \log n)$ space and $O(\log^2 n)$ time
  Goodrich, Tamassia; 1998

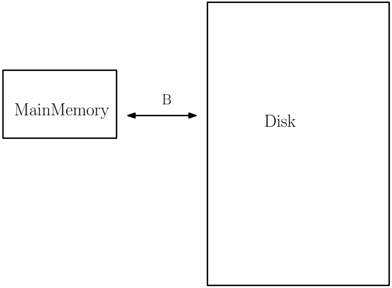
Further Research

- 3-D Point Location
- Dynamic Nearest Neighbor in 2-D

Further Research

External-Memory Model

- Limited-size main memory and potentially unlimited disk
- All operations in the main memory are *for free*
- Time complexity: number of I/Os between disk and main memory
  Each I/O reads/writes one block of size B



Further Research

- External-Memory Data Structures:
  Sorting Strings in External Memory
- Orthogonal Range Reporting, Point Location
- String Algorithms and CG

Thank You!