Improved Bounds for Orthogonal Point Enclosure Query and Point Location in Orthogonal Subdivisions in $\mathbb{R}^3$

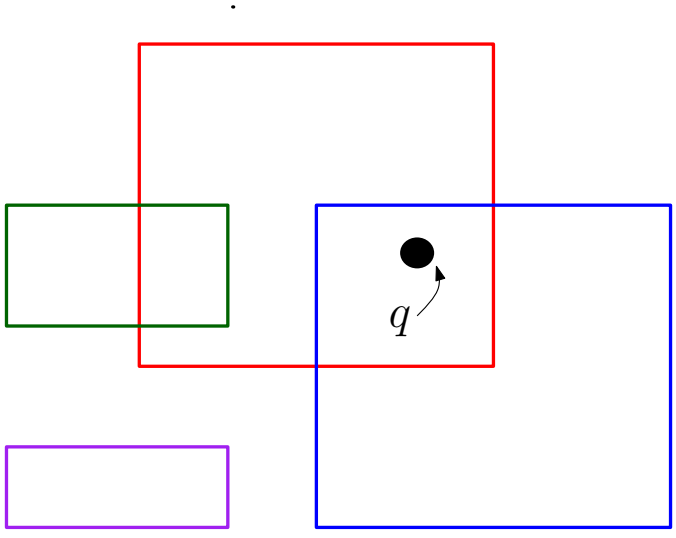
Saladi Rahul[1]

[1]Doctoral Candidate,
Dept. of Computer Science & Engg.,
University of Minnesota Twin-Cities

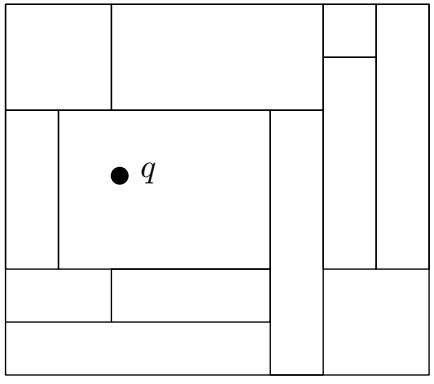
January 4, 2015

Two Fundamental Problems in Computational Geometry

Orthogonal Point Enclosure Query (*OPEQ*)



Point Location in Orthogonal Subdivisions



Orthogonal Point Enclosure Query (*OPEQ*)

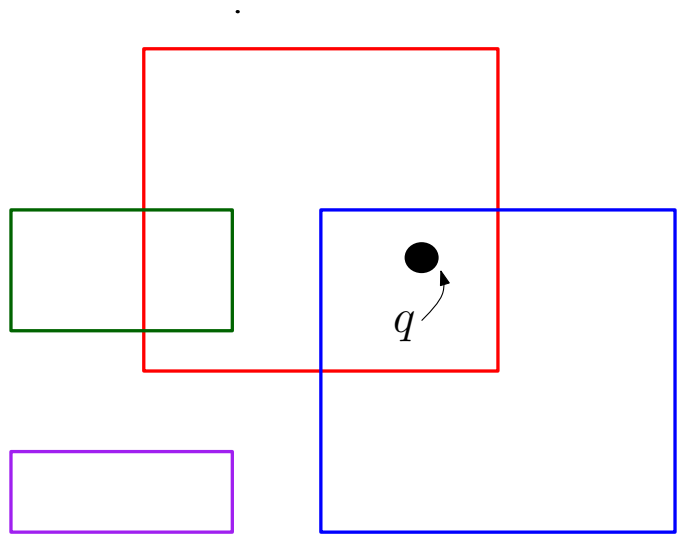
Input: A set S of n hyper-rectangles in d-dimensional space.

Query: A point q

Output: Rectangles in S containing q.

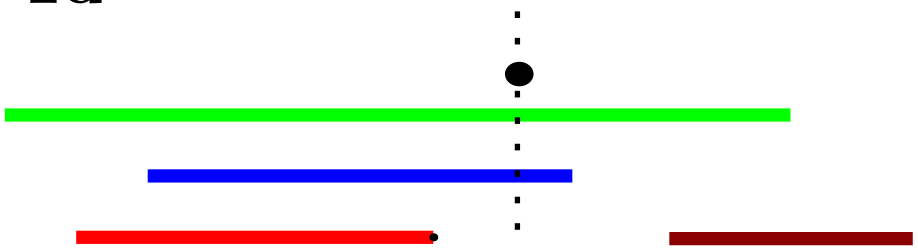
Objective:

- Minimize the space occcupied by the data structure
- Minimize query time.



OPEQ solved optimally in $1d$ and $2d$

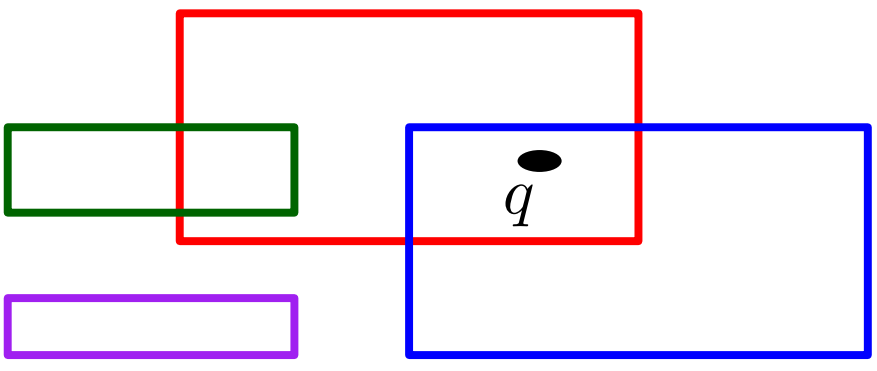
1d



Space: $O(n)$

Query Time: $O(\log n + k)$

2d



Space: $O(n)$

Query Time: $O(\log n + k)$

Comparison Model and Pointer Machine model: $\Omega(\log n + k)$

OPEQ in 3*d*

Upper Bound

$O(n)$

$O(\log^4 n + k)$

BIG GAP!

Lower Bound

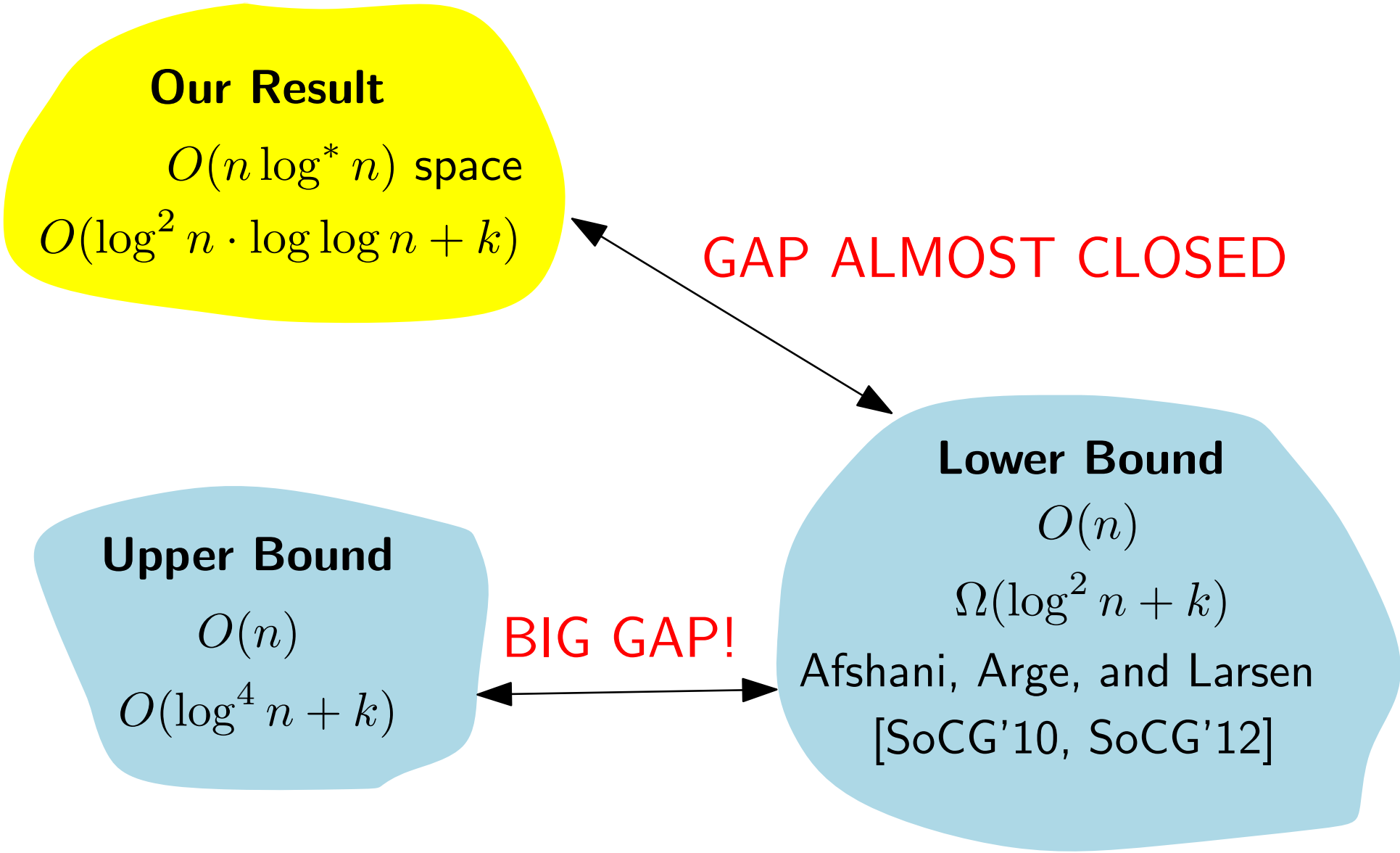
$O(n)$

$\Omega(\log^2 n + k)$

Afshani, Arge, and Larsen

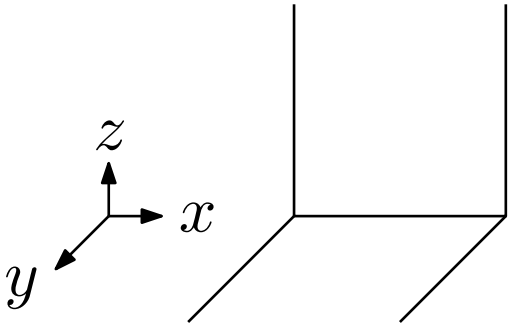
[SoCG'10, SoCG'12]

Our Result for *OPEQ* in 3*d*

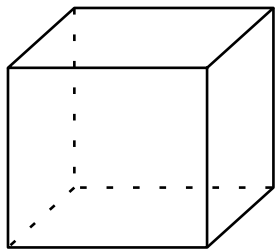


High-Level Strategy

4-sided rectangles

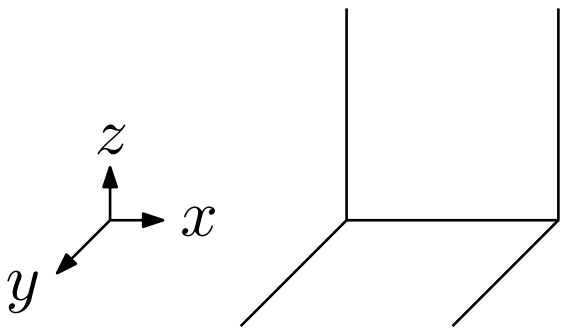
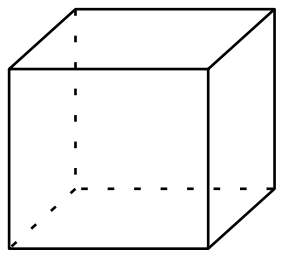


6-sided rectangles

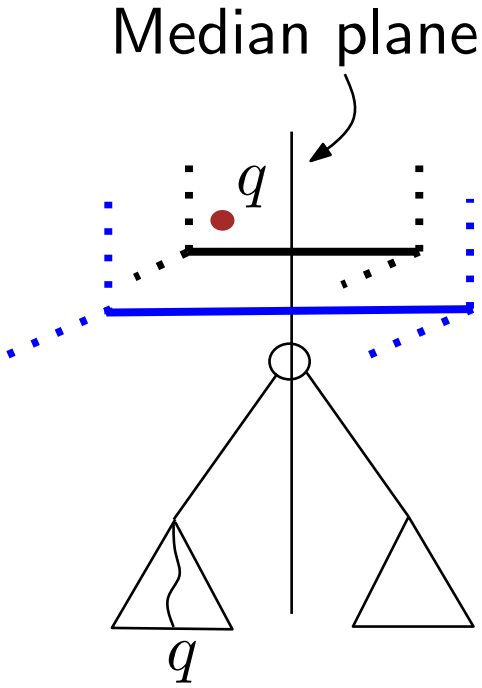


High-Level Strategy

Key Contribution

	4-sided rectangles	6-sided rectangles
		
Space	$O(n \log^* n)$	$O(n \log^* n)$
Query Time	$O(\log n + k)$	$O(\log^2 n \cdot \log \log n + k)$

OPEQ for 4-sided rectangles: Preprocessing

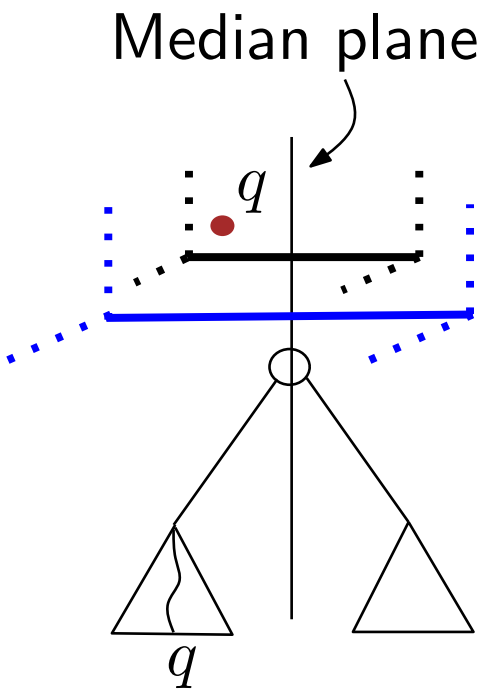


Interval Tree
on x-projections

Each rectangle assigned to exactly one node.

Find median of the $2n$ endpoints.

OPEQ for 4-sided rectangles: Preprocessing

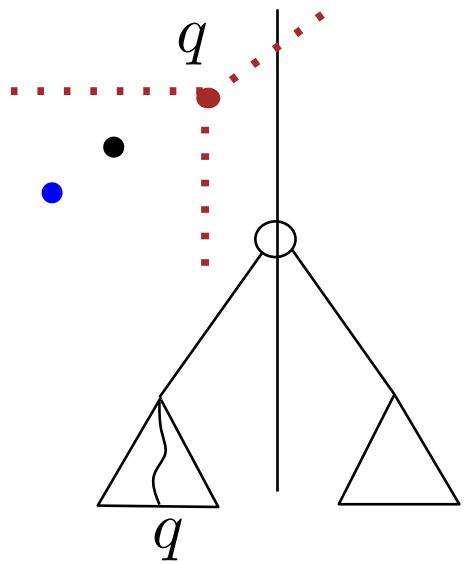


Interval Tree
on x-projections

Duality

$3d$ dominance query

Input: Points, **Query:** Octant



OPEQ for 4-sided rectangles: Query Algorithm

Query the nodes from root till leaf.

Time spent at each node $= O(\log n + k_v)$

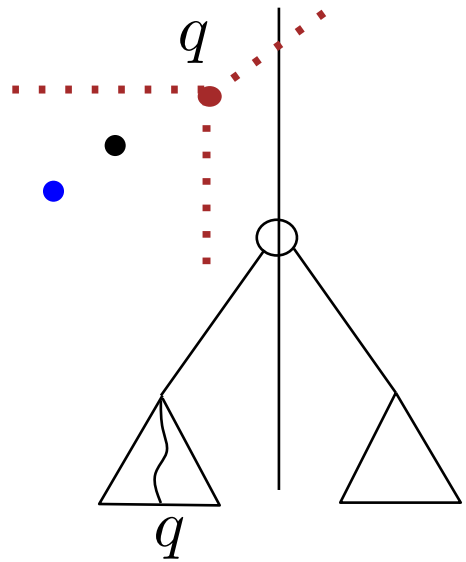
Query time $= \sum O(\log n + k_v)$

$= O(\log^2 n + k)$

Space $= O(n)$

Afshani [ESA'08]

$3d$ dominance query



OPEQ for 4-sided rectangles: Bottleneck

Current Bottleneck:

Time spent at each node $= O(\log n + k_v)$

Overall query time $= O(\log^2 n + k)$

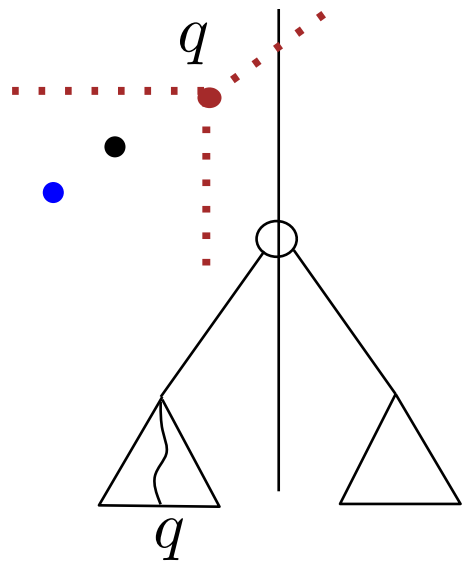
Objective:

Time spent at each node $= O(\log^* n + k_v)$

Overall query time $= O(\log n \cdot \log^* n + k)$

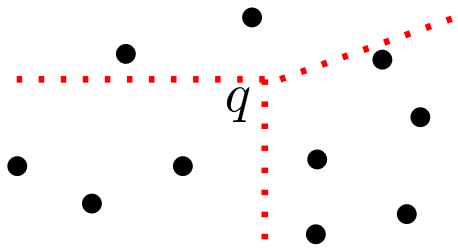
Afshani [ESA'08]

$3d$ dominance query



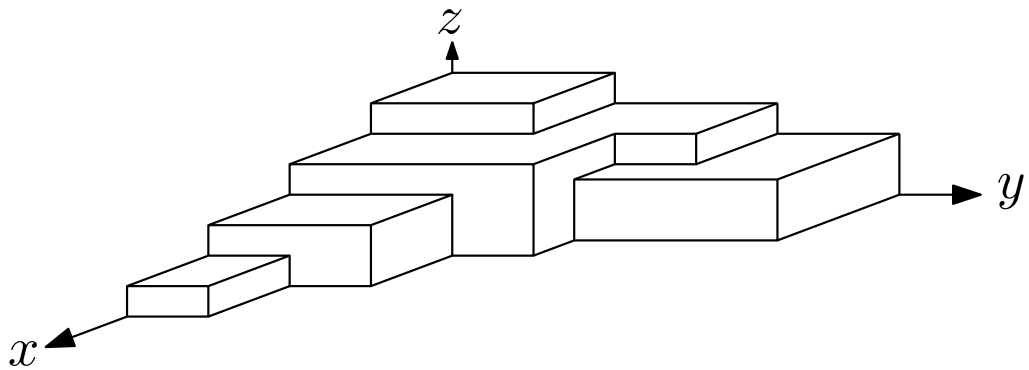
First Idea: Use Shallow Cuttings

$3d$ dominance query



Afshani[ESA'08]

$\log n$-shallow cutting



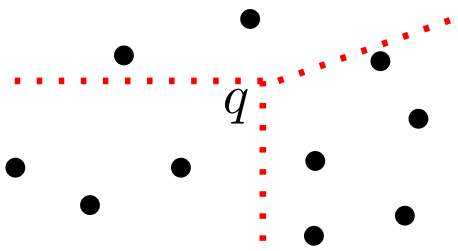
k_v= # points lying inside q

(1) If q lies inside the "surface"

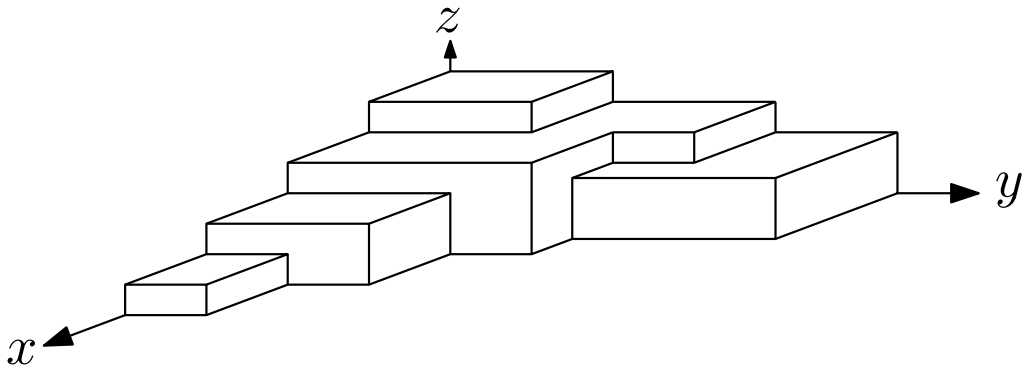
- $3d$ dominance query takes $O(\log\log n + k_v)$ time.

First Idea: Use Shallow Cuttings

$3d$ dominance query



$\log n$-shallow cutting



$k_v =$ # points lying inside q

(1) If q lies inside the "surface"

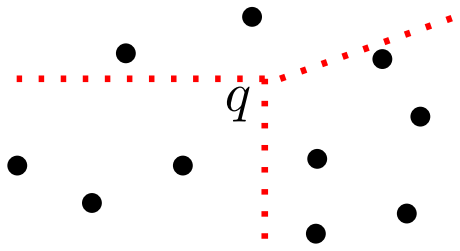
- $3d$ dominance query takes $O(\log \log n + k_v)$ time.

(2) Otherwise, $k_v \geq \log n$

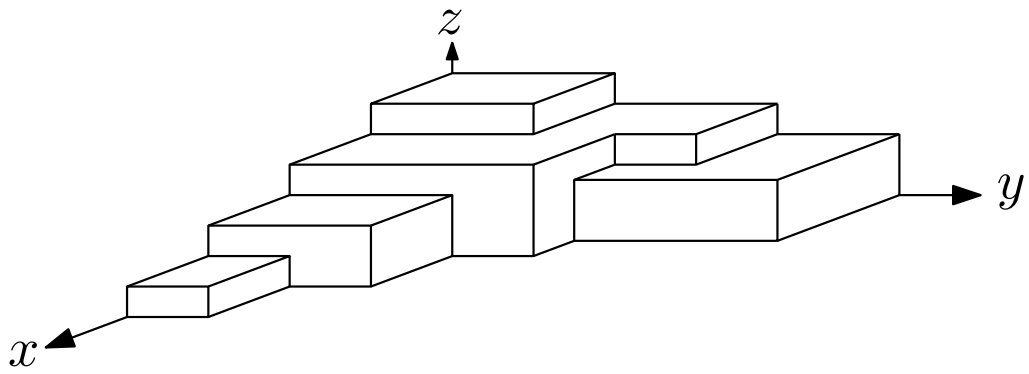
- can afford $O(\log n + k_v) = O(k_v)$ time

First Idea: Use Shallow Cuttings

$3d$ dominance query



$\log n$-shallow cutting



(1) If q lies inside the "surface"

- $3d$ dominance query takes $O(\log\log n + k_v)$ time.

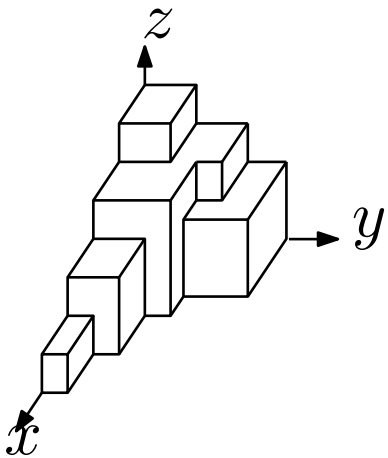
(2) Otherwise, $k_v \geq \log n$

- can afford $O(\log n + k_v) = O(k_v)$ time

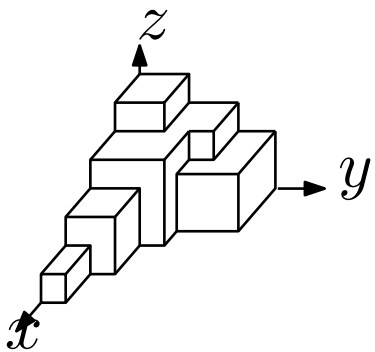
Time spent at a node $v = O(\log n + k_v) = O(\log\log n + k_v)$.

How does shallow cuttings help?

$\log n$ level

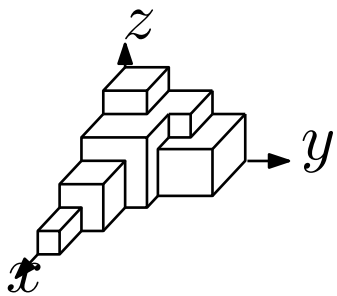


$\log \log n$ level

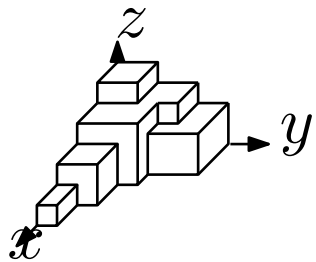


$\cdots$

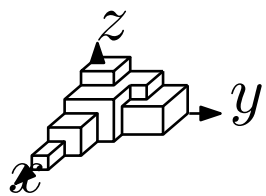
$\log^{(i)} n$ level



$\log^{(i+1)} n$ level

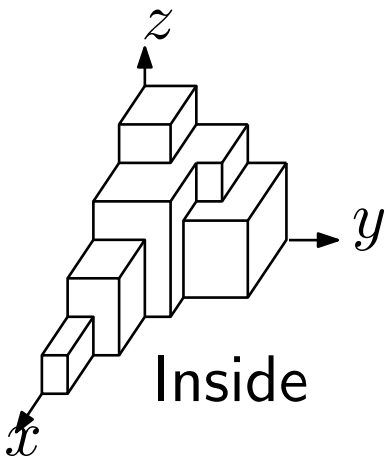


$\cdots$ 1 level



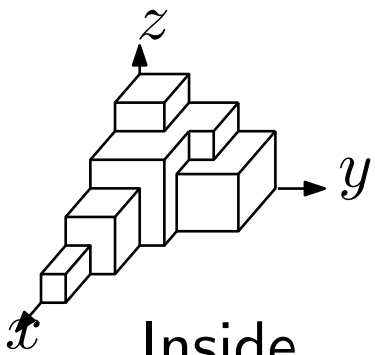
How does shallow cuttings help?

$\log n$ level



Inside

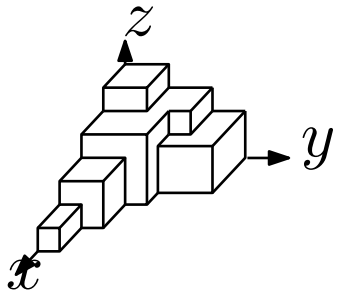
$\log \log n$ level



Inside

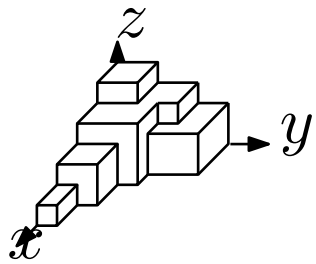
$\cdots$

$\log^{(i)} n$ level



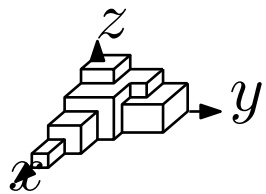
Inside

$\log^{(i+1)} n$ level



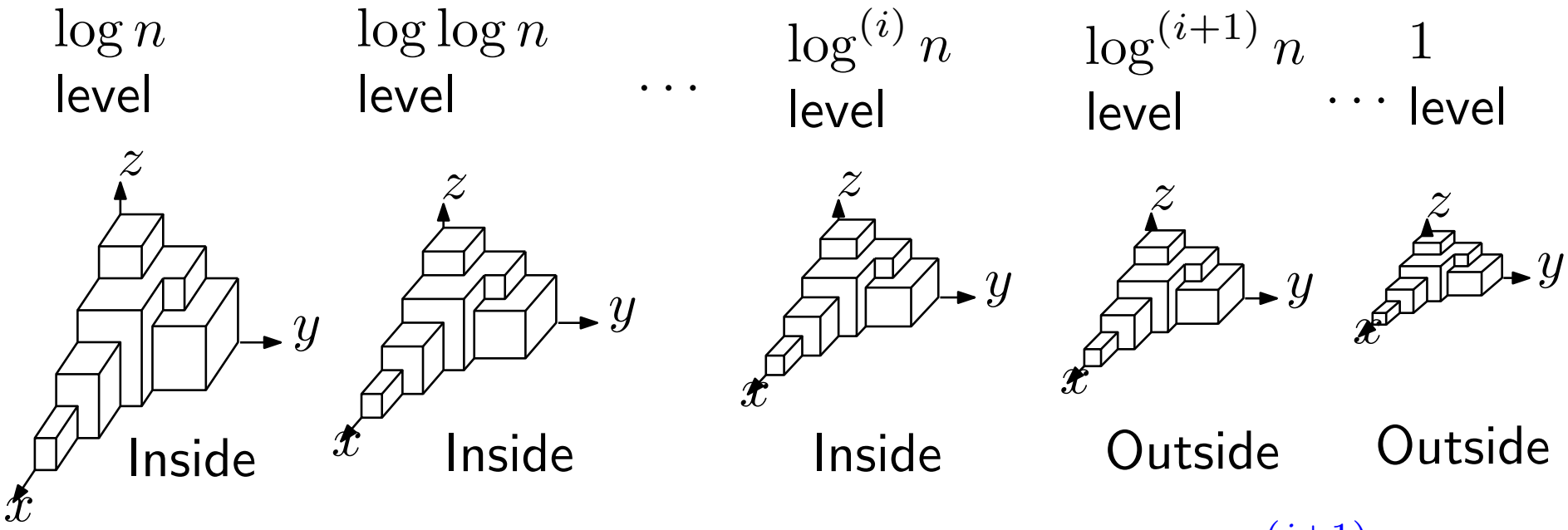
Outside

$\cdots$ 1 level



Outside

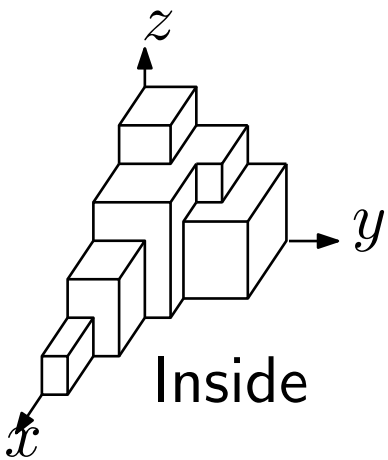
How does shallow cuttings help?



$$k_v = \Omega(\log^{(i+1)} n)$$

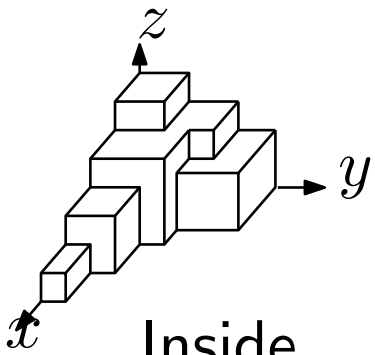
How does shallow cuttings help?

$\log n$ level



Inside

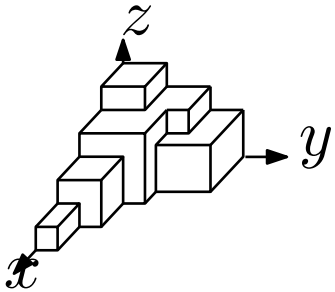
$\log \log n$ level



Inside

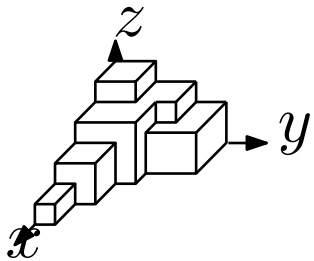
$\cdots$

$\log^{(i)} n$ level



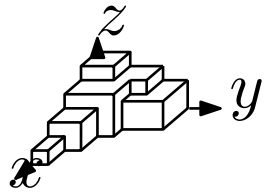
Inside

$\log^{(i+1)} n$ level



Outside

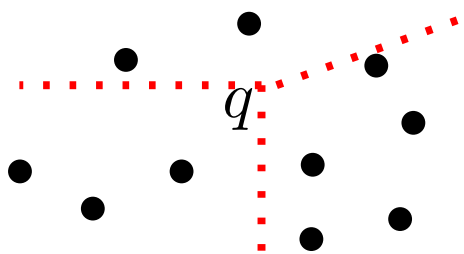
$\cdots$ 1 level



Outside

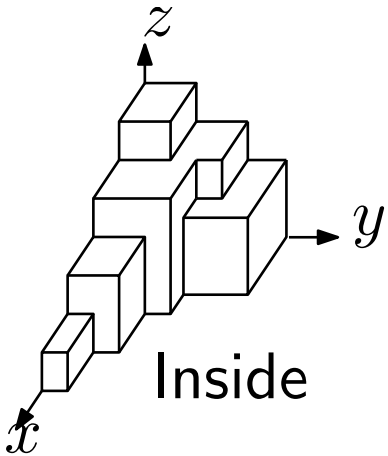
$k_v = \Omega(\log^{(i+1)} n)$

$3d$ dominance query

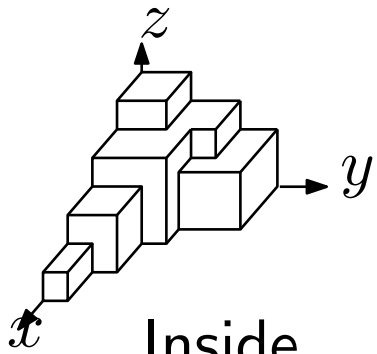


$$O(\log(\log^{(i)} n) + k_v)$$
$$= O(\log^{(i+1)} n + k_v)$$
$$= O(k_v)$$

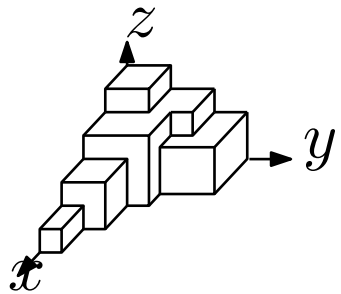
How to identify Inside/Outside?



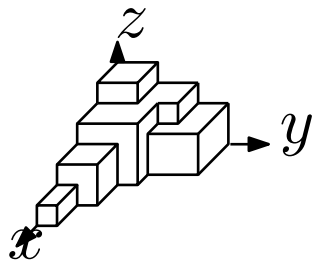
Inside



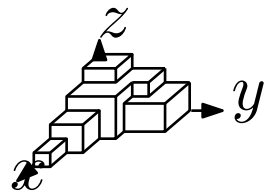
Inside



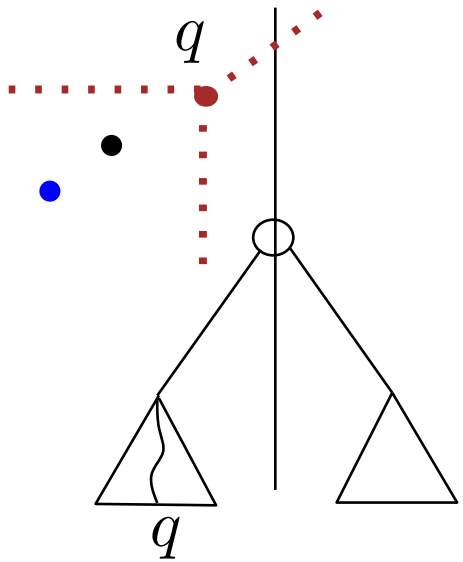
Inside



Outside



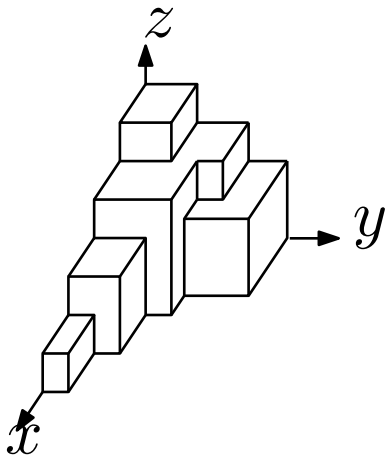
Outside



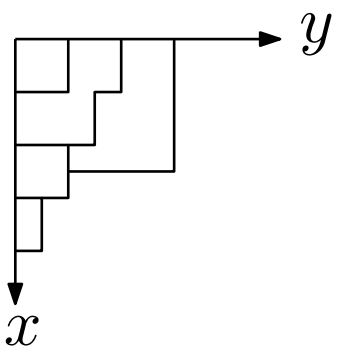
Naive Implementation $= O(\log^2 n \cdot \log^* n)$ time.

Second Idea: "Parallel" Point Location

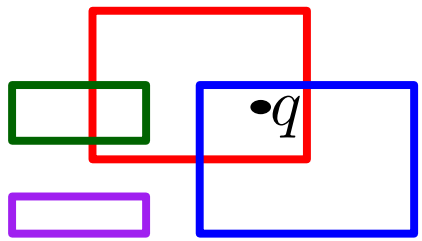
Inside or Outside?

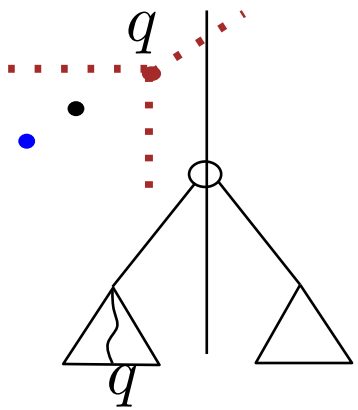


$2d$ subdivision



$OPEQ$ in $2d$





Combine $2d$ subdivisions from all nodes in the tree

Finding cells $= O(\log^2 n \cdot \log^* n)$

$= O(\log n \cdot \log^* n)$ time.

OPEQ for 4-sided rectangles: Summary

$O(n \log^* n)$ space

$O(\log n \cdot \log^* n + k)$

Non-trivial combination of

- Shallow Cuttings
- $OPEQ$ in $2d$
- Interval Tree

OPEQ for 4-sided rectangles: Summary

$O(n \log^* n)$ space

$O(\log n + k)$

Extra Tools Needed !!

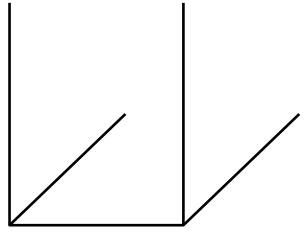
- Larger fanout of $2^{\log^* n}$.
- Segment Tree
- Persistence
- ...

High-Level Strategy

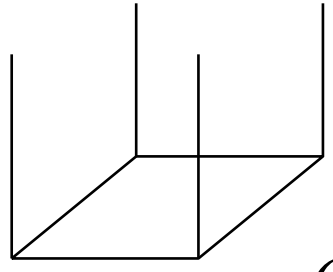
4-sided rectangles

$O(n \log^* n)$

$O(\log n + k)$



Grid Technique
Alstrup, Brodal, and Rauhe [FOCS'00]

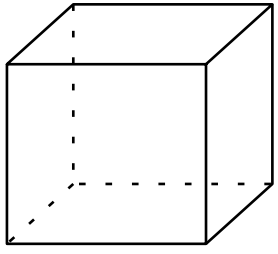


$O(n \log^* n)$

$O(\log n \cdot \log \log n + k)$

5-sided rectangles

Trivial



$O(n \log^* n)$

$O(\log^2 n \cdot \log \log n + k)$

6-sided rectangles

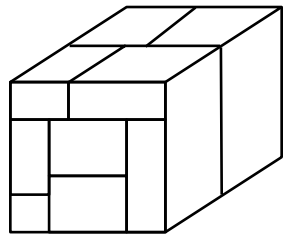
Orthogonal Point Location in $3d$: Pointer Machine

Previous Results:

- Edelsbrunner, Haring and Hilbert [The Computer Journal 1986]
  Space $= O(n)$
  Query Time $= O(\log^2 n)$.
- Afshani, Arge and Larsen [SoCG'10]
  Space $= O(n)$
  Query Time $= O(\log^2 n / \log\log n)$.

Our Result:

- Space $= O(n)$
  Query Time $= O(\log^{3/2} n)$.



Key Ideas:
Random Sampling on cuboids

$2d$ point location on "small" subsets.

$OPEQ$ in $3d$.

Questions?