

Orthogonal Range Searching

Computer Science Day 2009

Peyman Afshani



Faculty



Post Docs



Administration



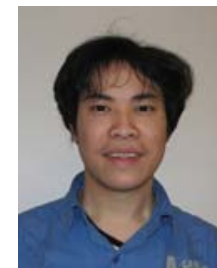
PhD students



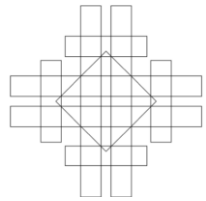
Programmers



Master Students



ACM Symposium on Computational Geometry



25th year celebration (extra invited talks)

Århus, Denmark, June 2009

followed by

Workshop on Massive Data Algorithmics

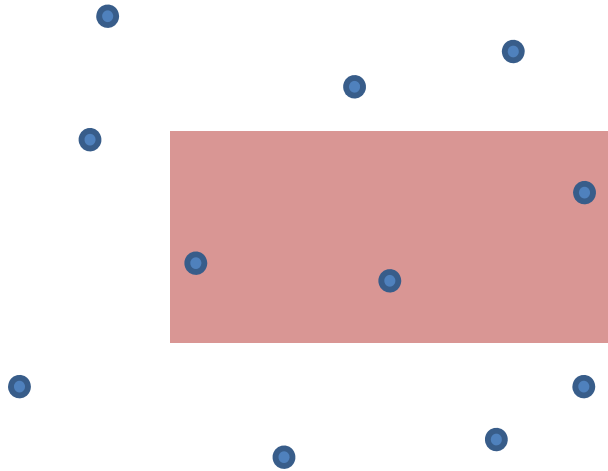




Who is inside the box?

Range Searching

- A set of points
- Process and store them

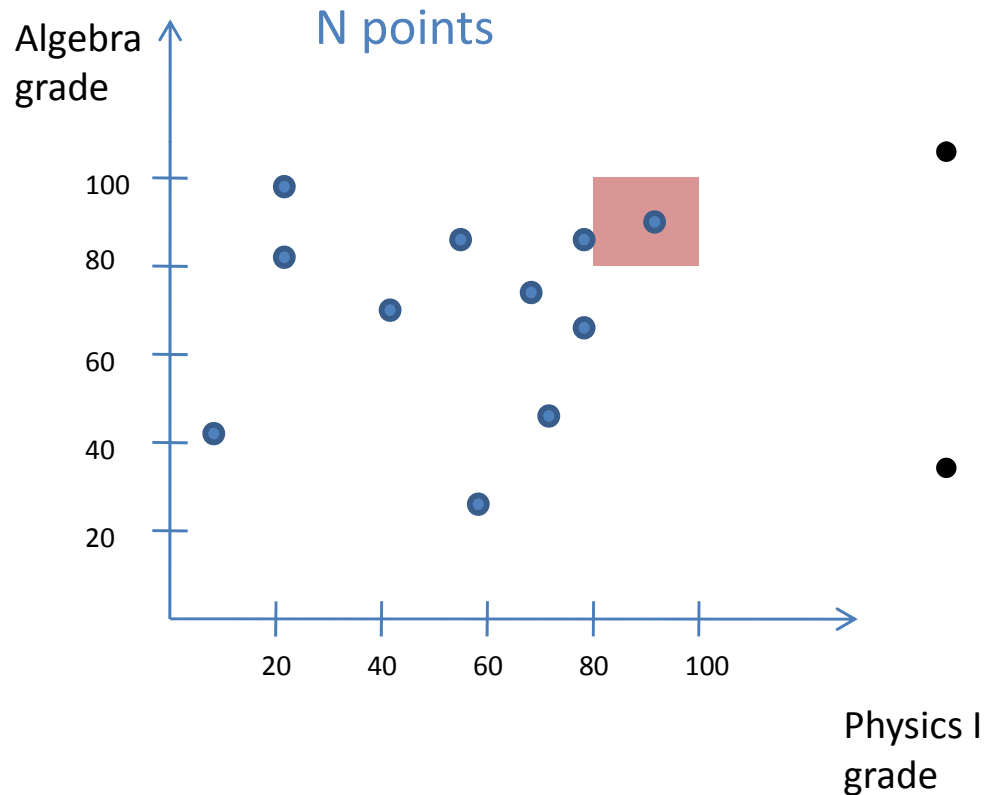


Goal:

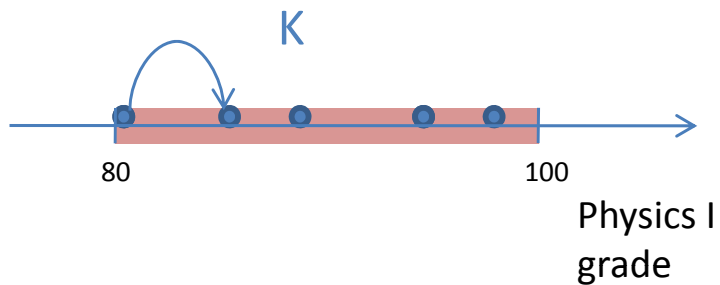
- Find points inside a region

More Than Just Finding Heads

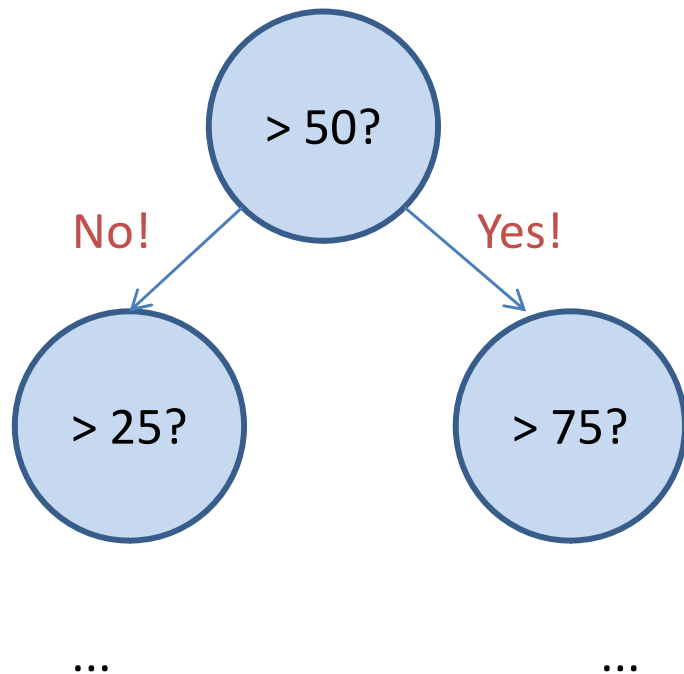
- Many applications
- Inefficient to go one by one
 - A data structure is needed
- For more than two dimensions the problem is still very interesting!
- **A Fundamental Problem**



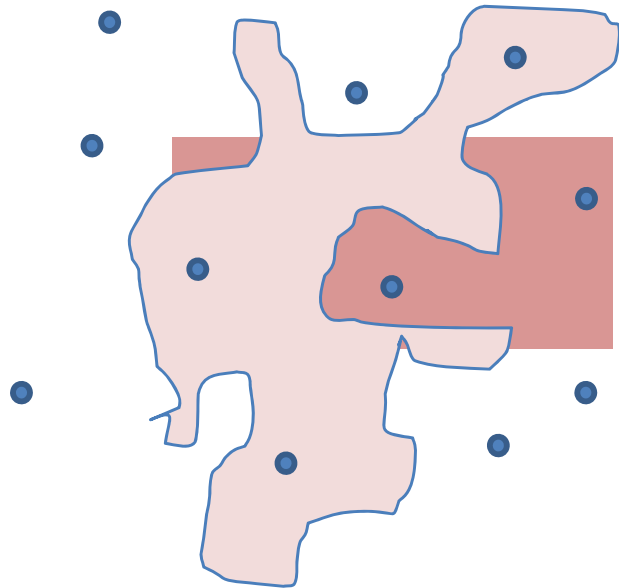
Starting Easy: One Dimension



- Each item points to the next
- Find first with binary search
- $O(\log N + K)$ time



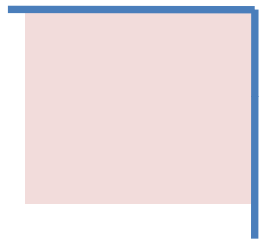
Beyond One Dimension: Already Hard



- Depends on the shape
- Let's deal with rectangles for now
- Other important shapes:
 - circles
 - halfspaces (halfplanes)
 - simplices (triangles in 2d)

Orthogonal (rectangular) Range Searching

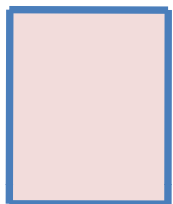
In two dimensions:



Two unbounded sides



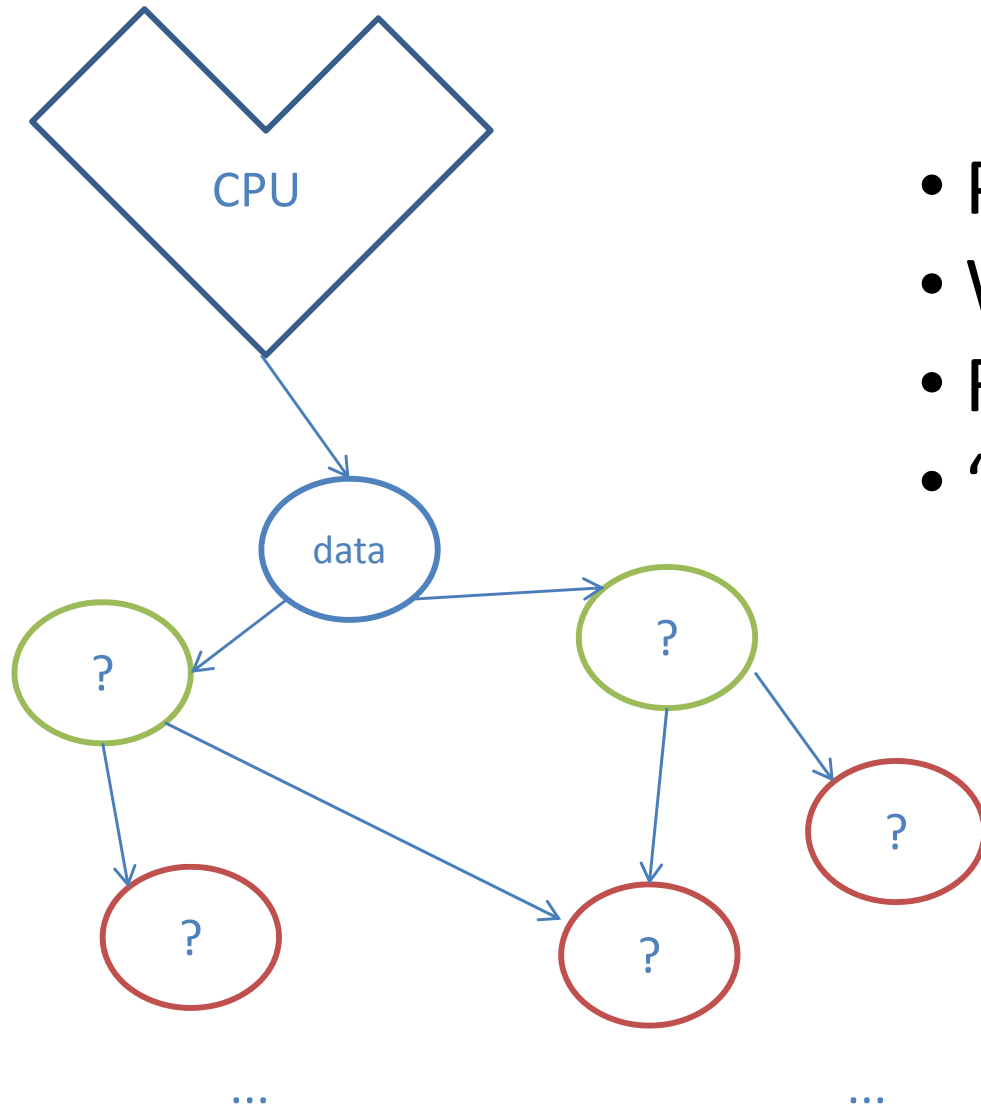
One unbounded side



No unbounded side

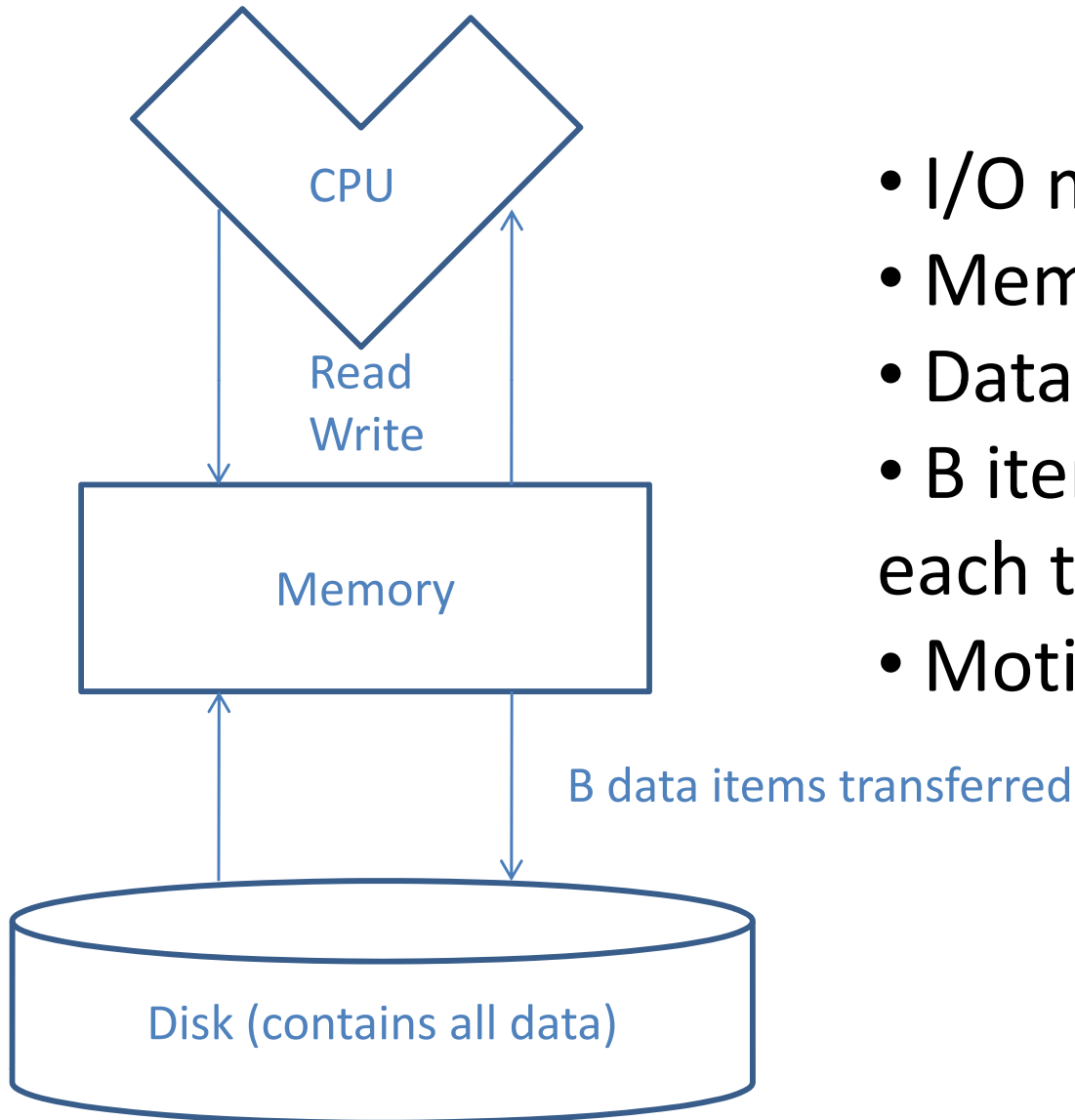
- Old problem
 - Been around for more than three decades
- Lots of variants
- Many models of computation
 - Random access machine
 - Pointer machine (no address calculation)
 - Massive data model
 - Cache-oblivious model

Pointer Machine Model



- Pointer Machine model
- We can only follow pointers
- Random jumps not allowed
- “Clean” model
 - Useful model for lower bounds

I/O Model



- I/O model
- Memory is limited
- Data is on a large disk
- B items can be transferred each time
- Motivation: massive data

Past Work

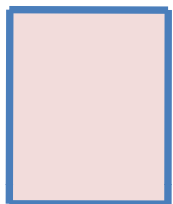
In two dimensions:



Two unbounded sides



One unbounded side



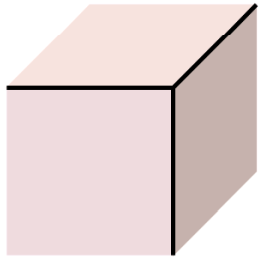
No unbounded side

Best solutions

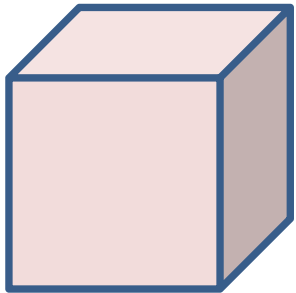
- One or two unbounded sides:
 - McCreight, 1985 (PM)
 - Lars et al., 1999 (I/O)
 - $O(N)$ space
 - $O(\log N + K)$ query (PM)
 - $O(\log_B N + K/B)$ query (I/O)
- No unbounded side:
 - Chazelle, 1986 (PM)
 - Lars et al., 1999 (I/O)
 - Query Same as above
 - $O(N \log N / \log \log N)$ space (PM)
 - $O(N \log N / \log \log_B N)$ space (I/O)

Past Work

Higher dimensions:



three dimensions
three unbounded
sides



three dimensions
no unbounded side



four dimensions

...

Best solutions

- 3D, three unbounded sides:

- Afshani, 2008 (PM, I/O)

- 3D, rest:

- Chazelle (1986)

- Chazelle (1990)

- Chazelle, Guibas (1986)

- Bozanis et al. (1997)

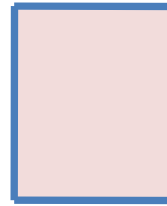
- Generalize to higher dimensions

- (more complicated bounds)

Adding Side (known technique)

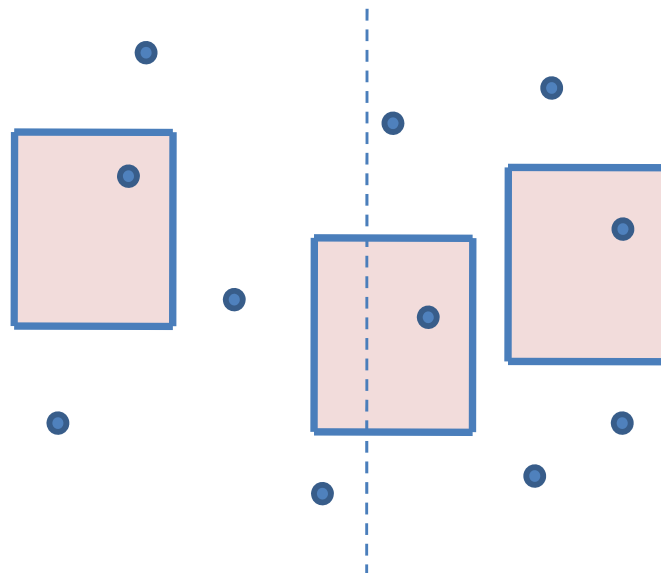


Assume we know how to answer these queries



But we want to answer these

case 1:
case 2:



Divide into left and right

Two cases

1. It does not cross
2. It does cross
 - Decomposes!

Our Improvement (Afshani, Arge, Larsen)



Assume we know how to
answer these queries

same query →



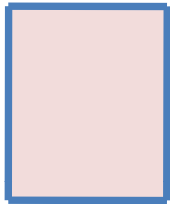
But we want to
answer these

~~log N~~ factor more space

→
log N / log log N

(In PM and I/O only)

Adding Dimension



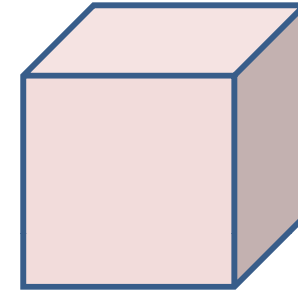
Assume we know how to answer these queries

$\log N / \log \log N$

~~$\log N$~~ factor to query →

~~$\log N$~~ factor to space →

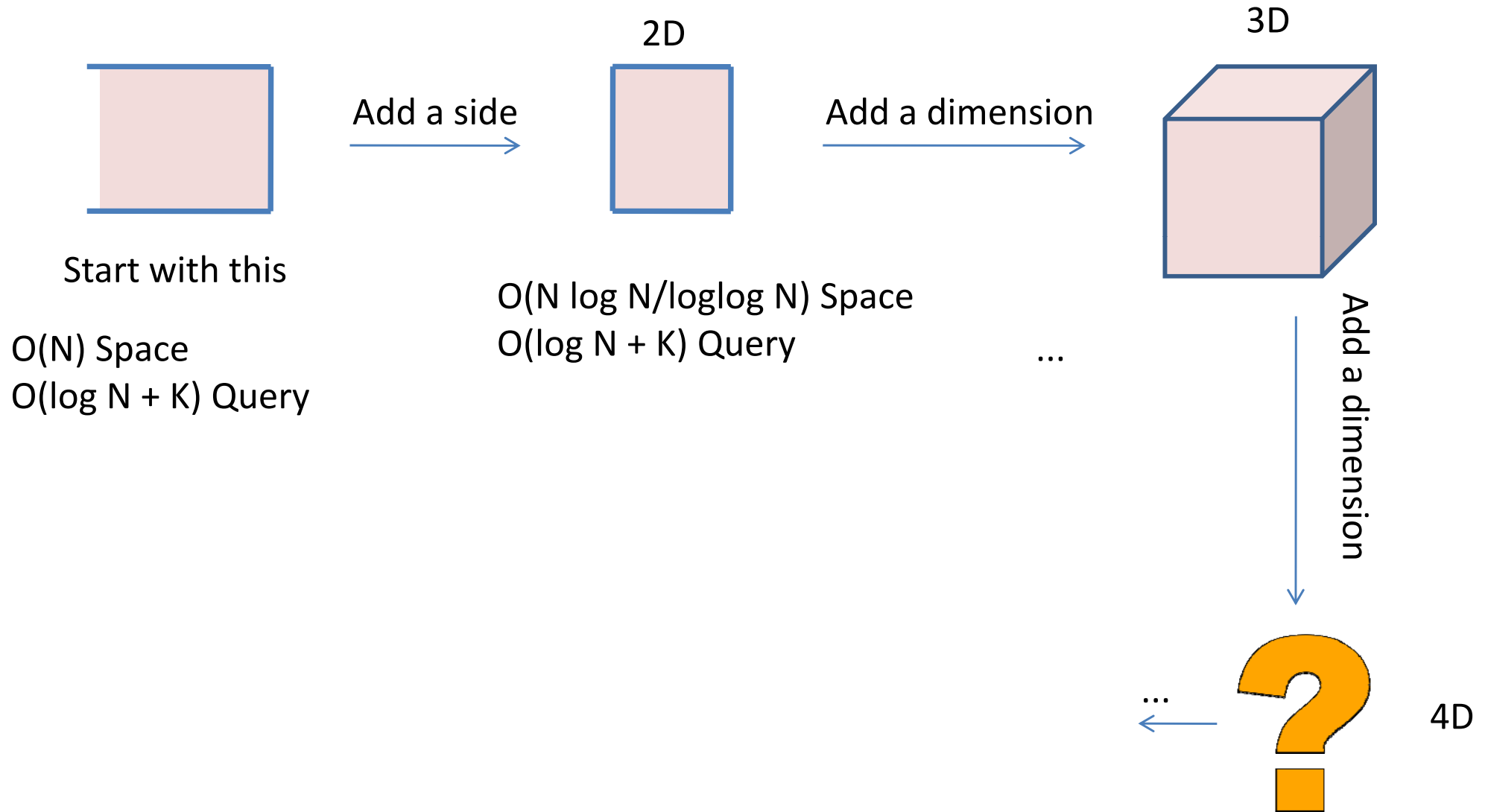
$\log N / \log \log N$



But we want to answer these

(In PM and I/O only)

Our General Approach



Our Results (Afshani, Arge, Larsen)

Space	Query	Reference	Model
$N \cdot \log^{d-2} N \cdot \log_{\log N} N$	$\log^{d-1} N + K$	Chazelle (1986)	PM
$N \cdot \log_{\log N}^{d-1} N$	$\log^{d-1+\varepsilon} N + K$	Chazelle (1990)	PM
$N \cdot \log_{\log N}^{d-1} N$	$\log N \cdot \log_{\log N}^{d-2} N + K$	Our improvement!	PM
<hr/>			
$N \cdot \log^d N$	$\log^{d-2} N + K$	Afshani (2008), Bozanis (1997), Chazelle & Guibas (1986)	PM
$N \cdot \log_{\log N}^3 N \cdot \log^{d-3} N$	$\log^{d-2} N + K$	Our improvement!	PM
<hr/>			
$N \cdot \log_{\log_B N}^{d-1} N$	$\log_B N \cdot \log_{\log_B N}^{d-2} N + K/B$	Our improvement!	I/O (totally new)

N: number input points K: number of output points B: block size

Notes

- The space bound $N \cdot \log_{\log N}^{d-1} N$ is optimal (in PM).
- We conjecture our query bound cannot be lowered, unless more space is consumed.
- We prove our space bound is optimal in I/O model as well
- Same conjecture

Conclusions

Further directions to research:

- Counting, emptiness, etc.
- Other models (i.e., cache-oblivious)
- Other shapes (halfspaces, simplices, etc.)
- Other theoretical questions
 - Lower bounds
 - Related combinatorial problems