

# Dynamic Maximum Depth of Geometric Objects

**Xiongxin Yang**

**NYU Shanghai → HKU**

**Based on joint work with**



**Subhash Suri**  
UCSB



**Jie Xue**  
NYU Shanghai



**Jiumu Zhu**  
NYU Shanghai

# Depth Problem

#objects that contain a point

# Depth Problem

#objects that contain a point

## MaxDepth

**Input:** a set of geometric objects  $\mathcal{O}$  (in  $\mathbb{R}^d$ )

**Output:** a point  $p^*$  s.t.

$$\text{dep}(p^*, \mathcal{O}) = \text{dep}(\mathcal{O}) = \max_{p \in \mathbb{R}^d} \text{dep}(p, \mathcal{O})$$

and the value of  $\text{dep}(p^*, \mathcal{O})$

# Depth Problem

#objects that contain a point

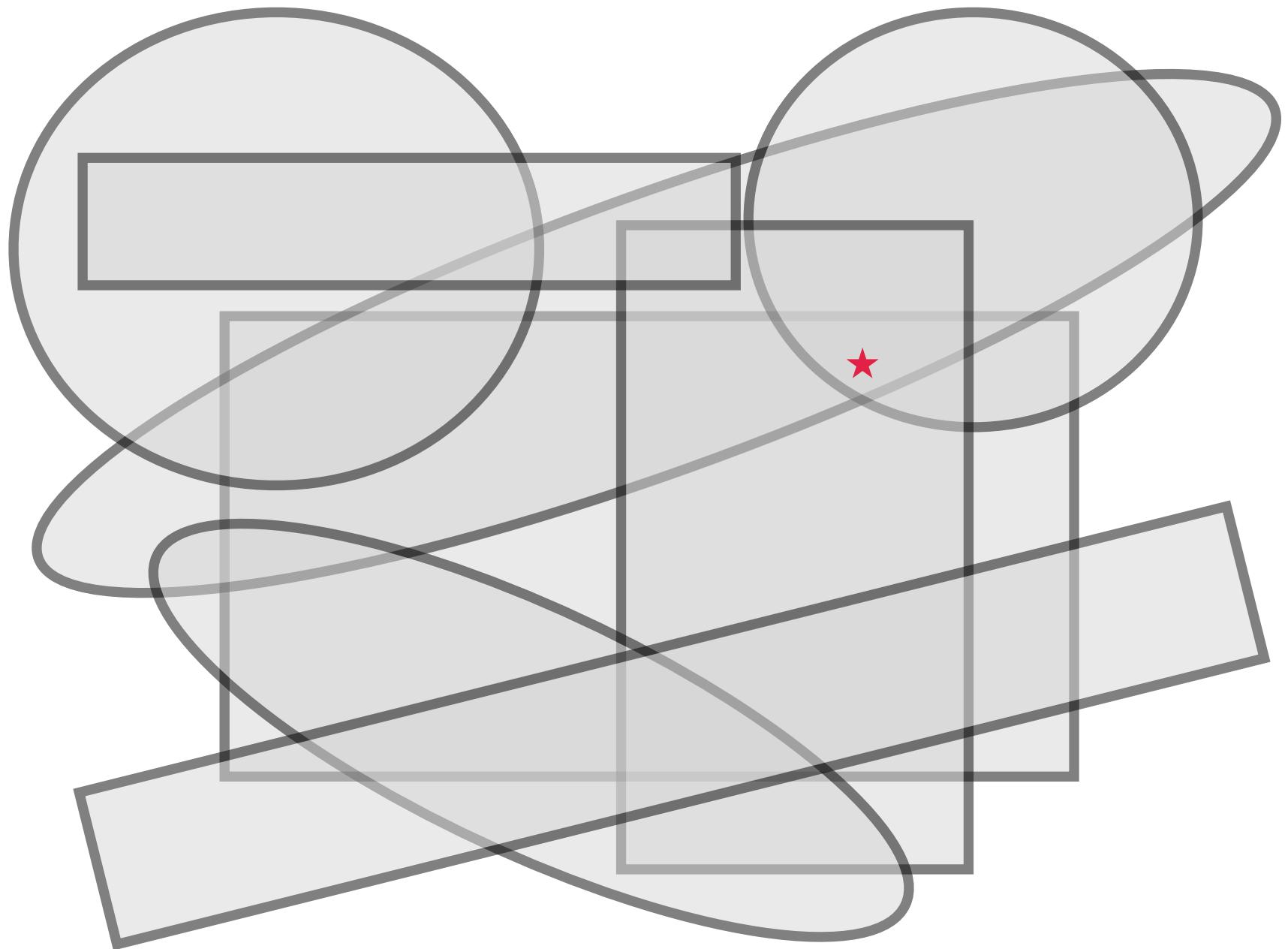
## MaxDepth

**Input:** a set of geometric objects  $\mathcal{O}$  (in  $\mathbb{R}^d$ )

**Output:** a point  $p^*$  s.t.

$$\text{dep}(p^*, \mathcal{O}) = \text{dep}(\mathcal{O}) = \max_{p \in \mathbb{R}^d} \text{dep}(p, \mathcal{O})$$

and the value of  $\text{dep}(p^*, \mathcal{O})$



# Depth Problem

#objects that contain a point

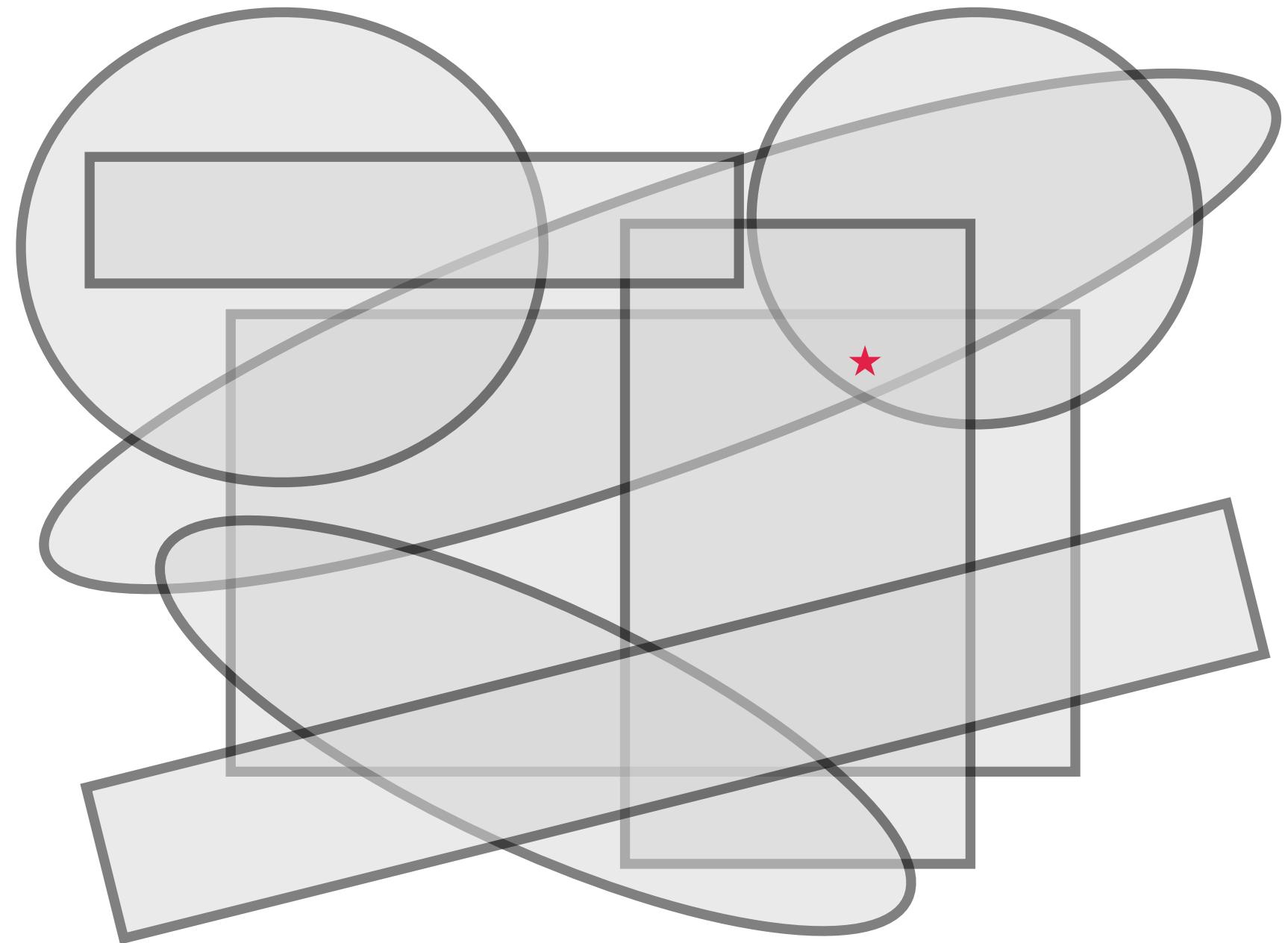
## MaxDepth

**Input:** a set of geometric objects  $\mathcal{O}$  (in  $\mathbb{R}^d$ )

**Output:** a point  $p^*$  s.t.

$$\text{dep}(p^*, \mathcal{O}) = \text{dep}(\mathcal{O}) = \max_{p \in \mathbb{R}^d} \text{dep}(p, \mathcal{O})$$

and the value of  $\text{dep}(p^*, \mathcal{O})$



## Canonical shape

interval, half-plane, rectangle (box), disk (ball), etc.

# Depth Problem

#objects that contain a point

## MaxDepth

**Input:** a set of geometric objects  $\mathcal{O}$  (in  $\mathbb{R}^d$ )

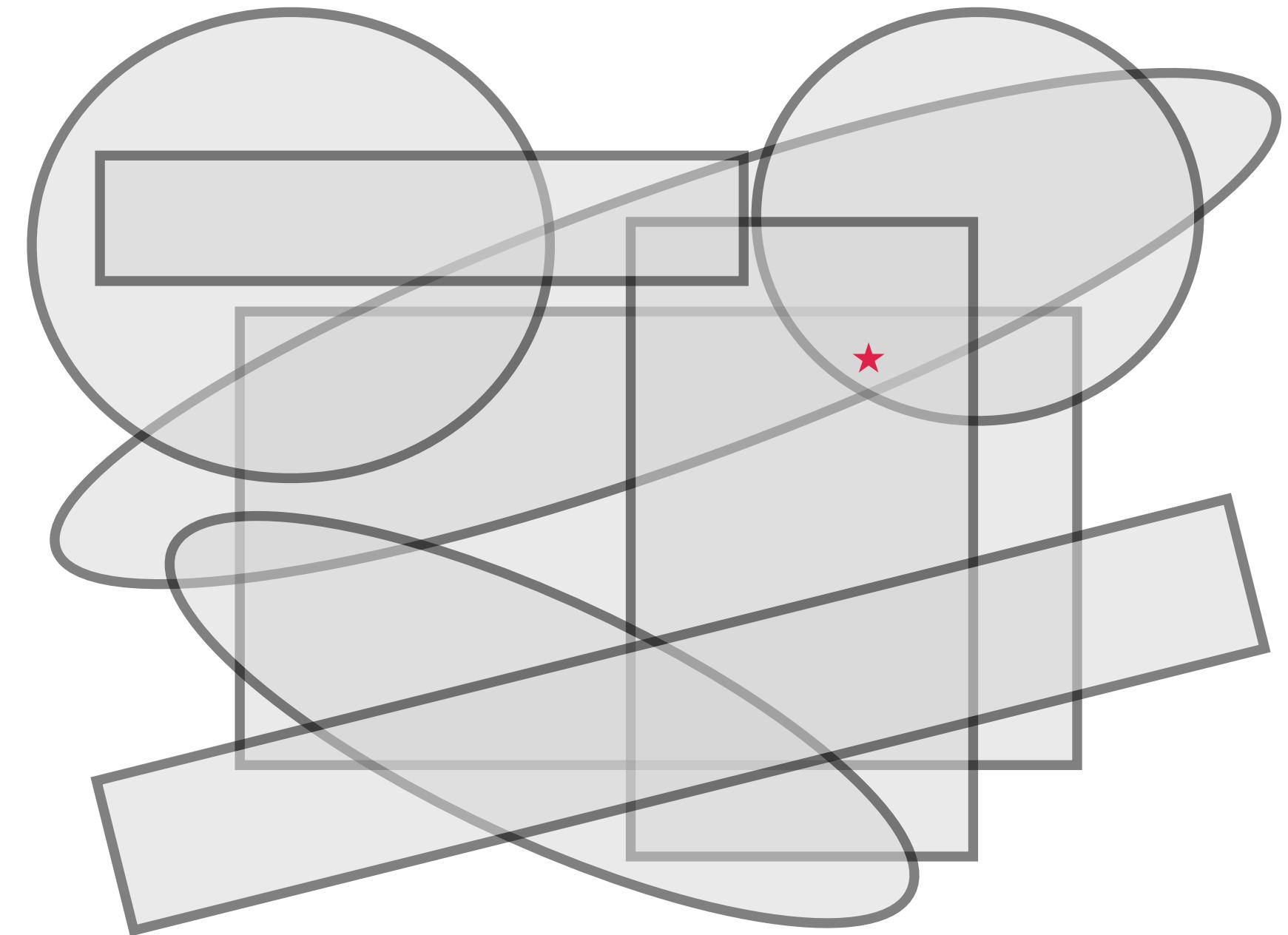
**Output:** a point  $p^*$  s.t.

$$\text{dep}(p^*, \mathcal{O}) = \text{dep}(\mathcal{O}) = \max_{p \in \mathbb{R}^d} \text{dep}(p, \mathcal{O})$$

and the value of  $\text{dep}(p^*, \mathcal{O})$

## Canonical shape

interval, half-plane, rectangle (box), disk (ball), etc.



## Dynamic version

After inserting or deleting an object,  
fast maintain the maximum depth

# **Dynamic Data Structure in CG**

**Dynamic data structure is an active topic in CG**

Dynamic + range search, point location, nearest neighbor, bichromatic closest pair,  
union of intervals, convex hull, width, geometric connectivity ...

# Dynamic Data Structure in CG

**Dynamic data structure is an active topic in CG**

Dynamic + range search, point location, nearest neighbor, bichromatic closest pair,  
union of intervals, convex hull, width, geometric connectivity ...

**Dynamic data structure for geometric optimization problem**

On geometric intersection graph:

minimum set cover [Agarwal, Chang, Suri, Xiao and Xue 2020; Chan and He 2021; Chan, He, Suri and Xue 2022]

minimum hitting set [Agarwal, Xie, Yang and Yu 2005; Agarwal, Chang, Suri, Xiao and Xue 2020]

minimum vertex cover [Bhattacharya and Kulkarni 2019; Bhore and Chan 2025]

maximum independent set [Henzinger, Neumann and Wiese 2020; Bhore and Chan 2025]

# Dynamic Data Structure in CG

**Dynamic data structure is an active topic in CG**

Dynamic + range search, point location, nearest neighbor, bichromatic closest pair,  
union of intervals, convex hull, width, geometric connectivity ...

**Dynamic data structure for geometric optimization problem**

On geometric intersection graph:

minimum set cover [Agarwal, Chang, Suri, Xie and

minimum hitting set [Agarwal, Xie, Yang and

minimum vertex cover [Bhattacharya and Kulkarni and

maximum independent set [Henzinger, Neumann and Wiese 2020; Bhore and Chan 2025]

They are all NP-hard,  
so we need approximation.

# Dynamic Data Structure in CG

**Dynamic data structure is an active topic in CG**

Dynamic + range search, point location, nearest neighbor, bichromatic closest pair,  
union of intervals, convex hull, width, geometric connectivity ...

**Dynamic data structure for geometric optimization problem**

On geometric intersection graph:

minimum set cover [Agarwal, Chang, Suri, Xie and

minimum hitting set [Agarwal, Xie, Yang and

minimum vertex cover [Bhattacharya and Kulkarni 2020]

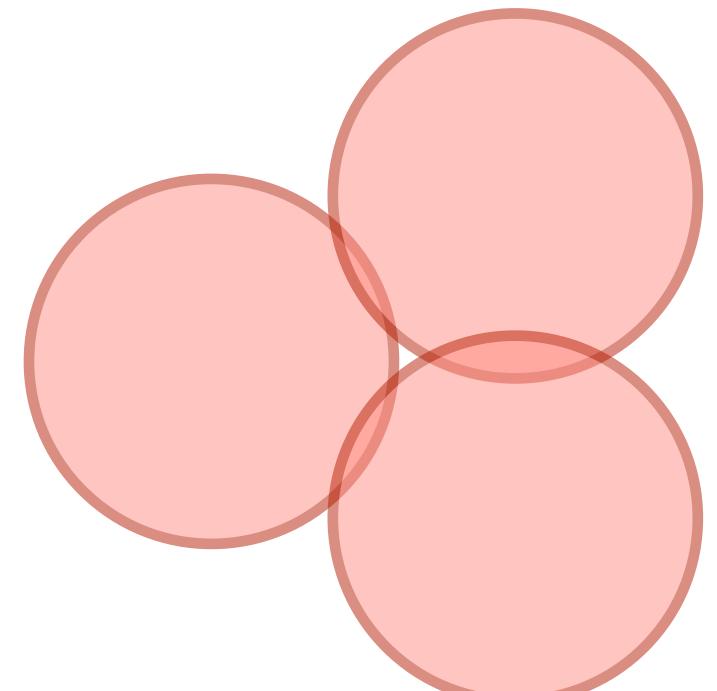
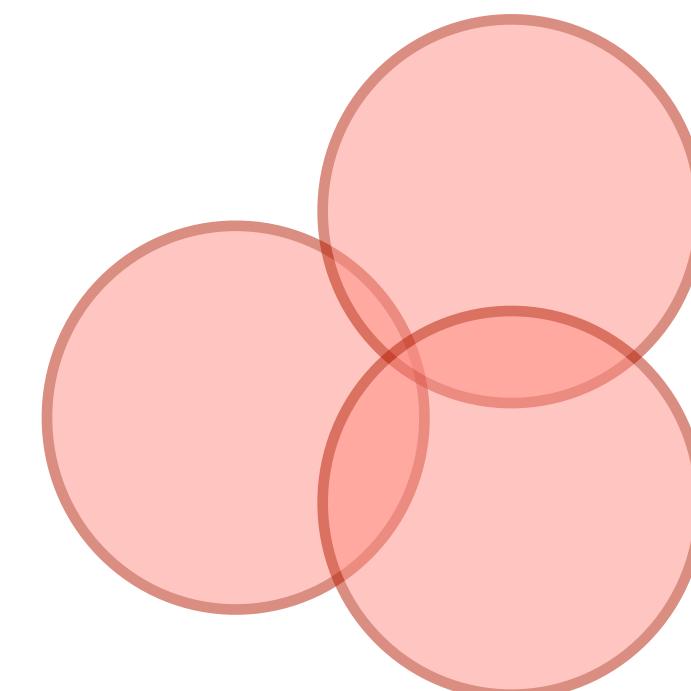
maximum independent set [Henzinger, Neumann and Wiese 2020; Bnoure and Chan 2025]

They are all NP-hard,  
so we need approximation.

**Dynamic Maximum Depth**

Static maximum depth is polynomial time solvable

Maximum depth is a constant approx. of maximum clique



# **Known Results for MaxDep**

# Known Results for MaxDep

## Static MaxDep Problem

For  $d$ -dimensional (axis-parallel) boxes:

- best known running time:  $O((n^{d/2}/\log^{d/2} n)(\log \log n)^{O(1)})$  [Chan 2013]
- $\tilde{O}(n^{d/2})$  might be tight [Chan 2010]

For disks: MaxDep is 3SUM-hard [Aronov and Har-Peled 2008]

# Known Results for MaxDep

## Static MaxDep Problem

For  $d$ -dimensional (axis-parallel) boxes:

- best known running time:  $O((n^{d/2}/\log^{d/2} n)(\log \log n)^{O(1)})$  [Chan 2013]
- $\tilde{O}(n^{d/2})$  might be tight [Chan 2010]

For disks: MaxDep is 3SUM-hard [Aronov and Har-Peled 2008]

## Dynamic MaxDep Problem

For (weighted) intervals:  $O(\log n)$  update time [Imai and Asano 1977]

# Known Results for MaxDep

## Static MaxDep Problem

For  $d$ -dimensional (axis-parallel) boxes:

- best known running time:  $O((n^{d/2}/\log^{d/2} n)(\log \log n)^{O(1)})$  [Chan 2013]
- $\tilde{O}(n^{d/2})$  might be tight [Chan 2010]

For disks: MaxDep is 3SUM-hard [Aronov and Har-Peled 2008]

## Dynamic MaxDep Problem

For (weighted) intervals:  $O(\log n)$  update time [Imai and Asano 1977]

## This Work

For (axis-parallel) rectangles:  $(1/k)$ -approx. with  $\tilde{O}(n^{1/(k+1)})$  amortized update time

# Known Results for MaxDep

## Static MaxDep Problem

For  $d$ -dimensional (axis-parallel) boxes:

- best known running time:  $O((n^{d/2}/\log^{d/2} n)(\log \log n)^{O(1)})$  [Chan 2013]
- $\tilde{O}(n^{d/2})$  might be tight [Chan 2010]

For disks: MaxDep is 3SUM-hard [Aronov and Har-Peled 2008]

## Dynamic MaxDep Problem

For (weighted) intervals:  $O(\log n)$  update time [Imai et al. 2011]

Exact data structure with  $\tilde{O}(\sqrt{n})$  update time,  
matching the lower bound

## This Work

For (axis-parallel) rectangles:  $(1/k)$ -approx. with  $\tilde{O}(n^{1/(k+1)})$  amortized update time

# Known Results for MaxDep

## Static MaxDep Problem

For  $d$ -dimensional (axis-parallel) boxes:

- best known running time:  $O((n^{d/2}/\log^{d/2} n)(\log \log n)^{O(1)})$  [Chan 2013]
- $\tilde{O}(n^{d/2})$  might be tight [Chan 2010]

For disks: MaxDep is 3SUM-hard [Aronov and Har-Peled 2008]

## Dynamic MaxDep Problem

For (weighted) intervals:  $O(\log n)$  update time [Imai et al. 2013]

Exact date structure with  $\tilde{O}(\sqrt{n})$  update time,  
matching the lower bound

## This Work

For (axis-parallel) rectangles:  $(1/k)$ -approx. with  $\tilde{O}(n^{1/(k+1)})$  amortized update time

For disks:  $(1/2 - \varepsilon)$ -approx. with  $\tilde{O}(n^{2/3})$  update time

# Known Results for MaxDep

## Static MaxDep Problem

For  $d$ -dimensional (axis-parallel) boxes:

- best known running time:  $O((n^{d/2}/\log^{d/2} n)(\log \log n)^{O(1)})$  [Chan 2013]
- $\tilde{O}(n^{d/2})$  might be tight [Chan 2010]

For disks: MaxDep is 3SUM-hard [Aronov and Har-Peled 2008]

## Dynamic MaxDep Problem

For (weighted) intervals

Exact date structure with  $\tilde{O}(\sqrt{n})$  update time,  
matching the lower bound

## This Work

Due to 3SUM-hardness, unlikely have  
sublinear update time for exact depth

For (axis-parallel) rectangles

$\tilde{O}(n^{1/(k+1)})$  amortized update time

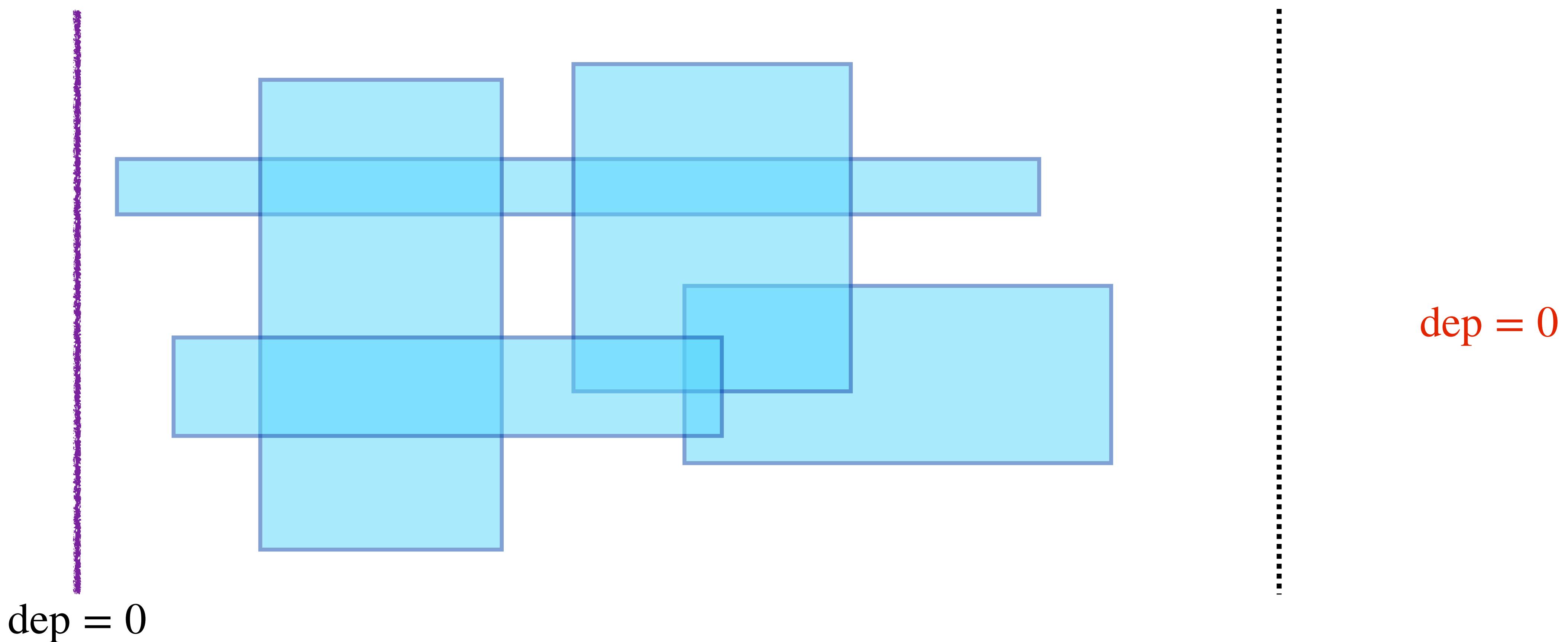
For disks:  $(1/2 - \varepsilon)$ -approx. with  $\tilde{O}(n^{2/3})$  update time

# Warm Up: Sweep-Line Algorithm for Rectangles

For rectangles:  $O(n \log n)$ -time algorithm via sweep line [Imai and Asano 1977]

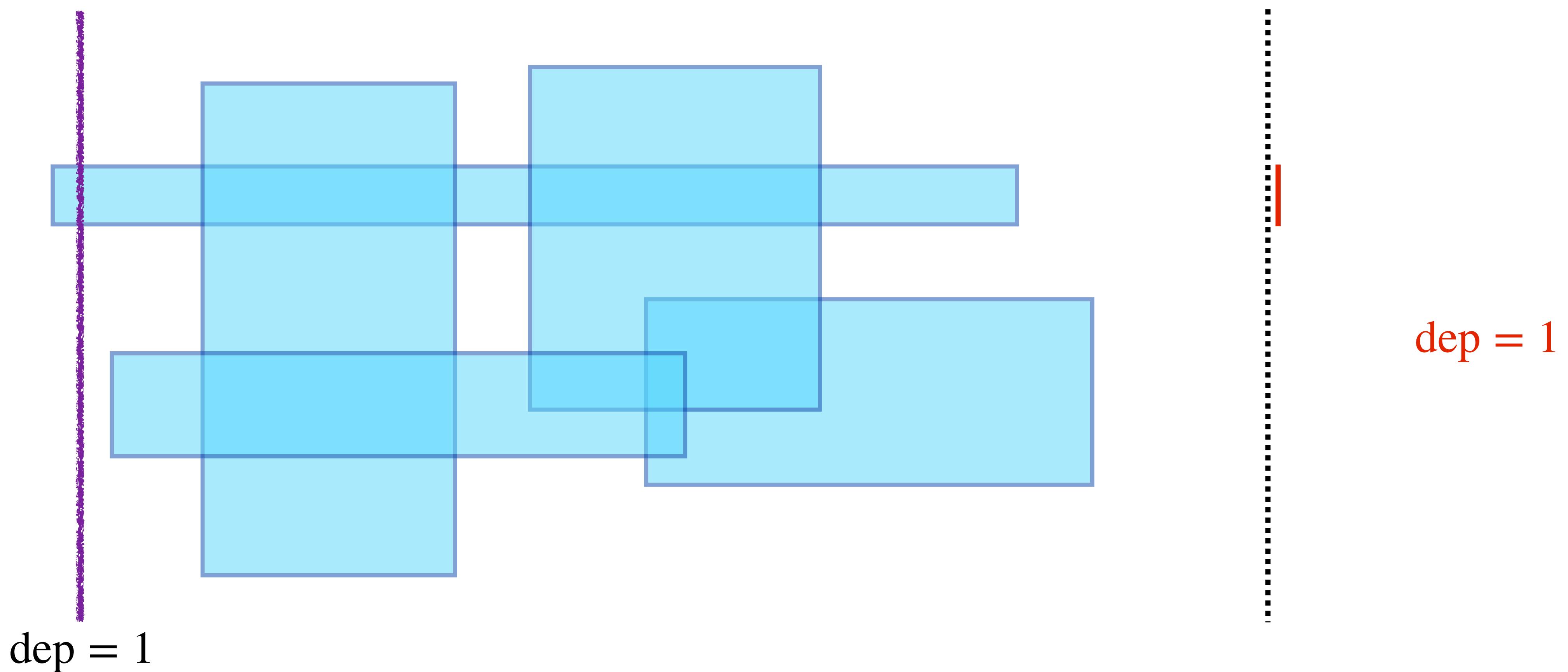
# Warm Up: Sweep-Line Algorithm for Rectangles

For rectangles:  $O(n \log n)$ -time algorithm via sweep line [Imai and Asano 1977]



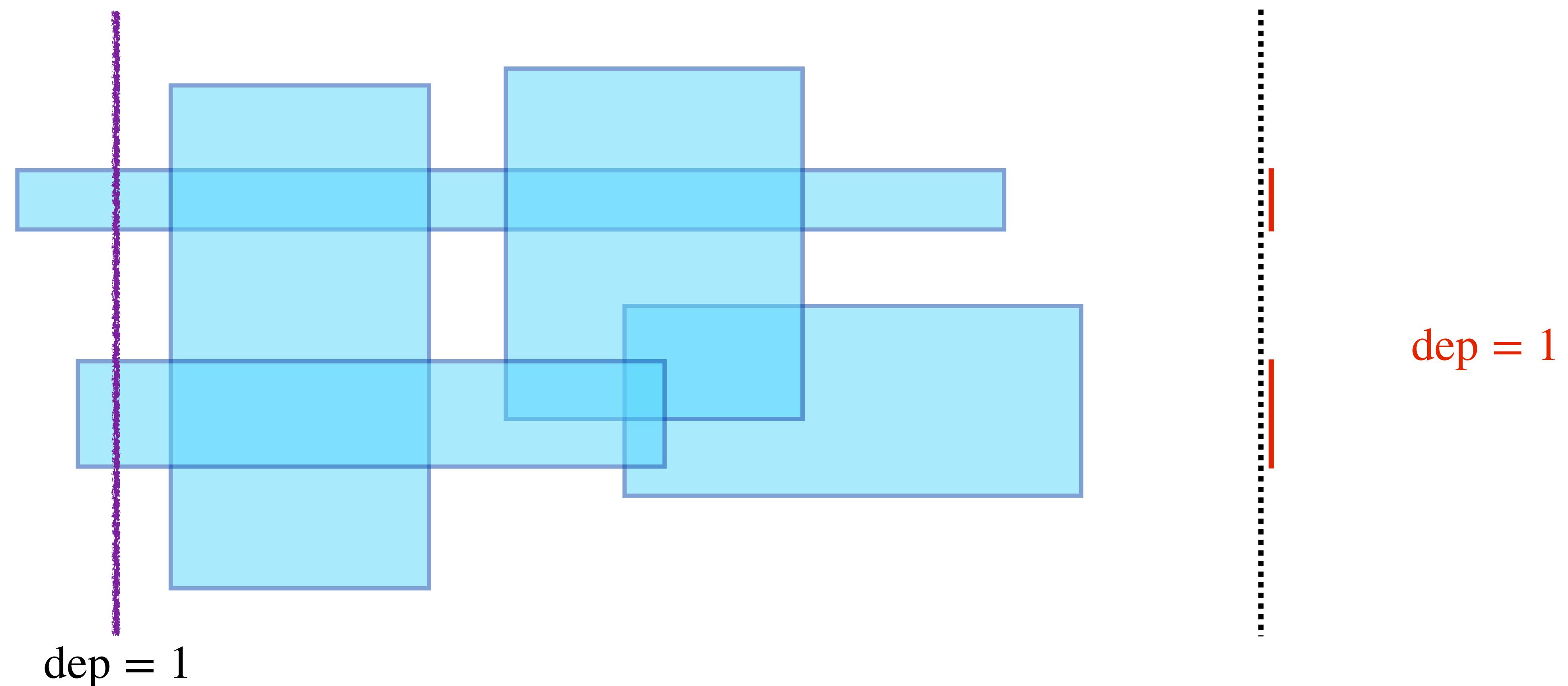
# Warm Up: Sweep-Line Algorithm for Rectangles

For rectangles:  $O(n \log n)$ -time algorithm via sweep line [Imai and Asano 1977]



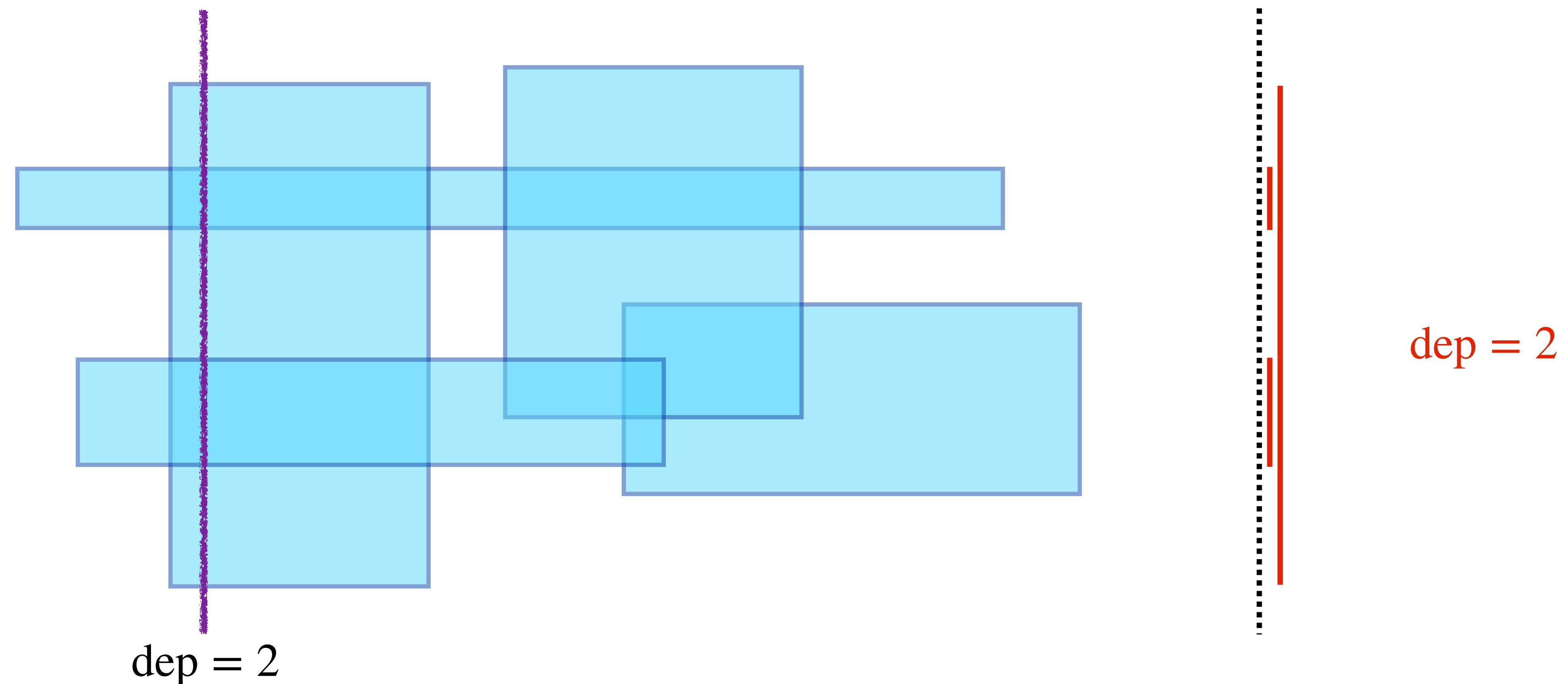
# Warm Up: Sweep-Line Algorithm for Rectangles

For rectangles:  $O(n \log n)$ -time algorithm via sweep line [Imai and Asano 1977]



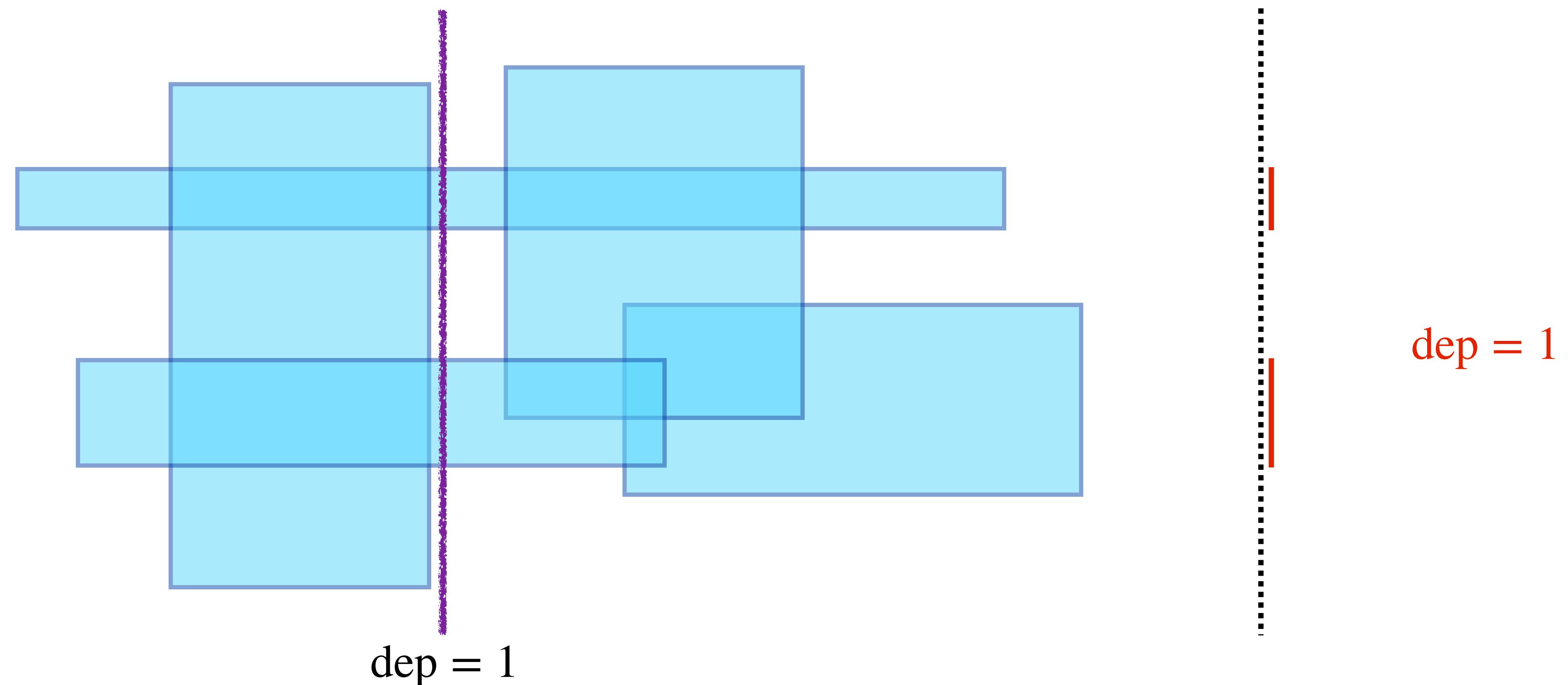
# Warm Up: Sweep-Line Algorithm for Rectangles

For rectangles:  $O(n \log n)$ -time algorithm via sweep line [Imai and Asano 1977]



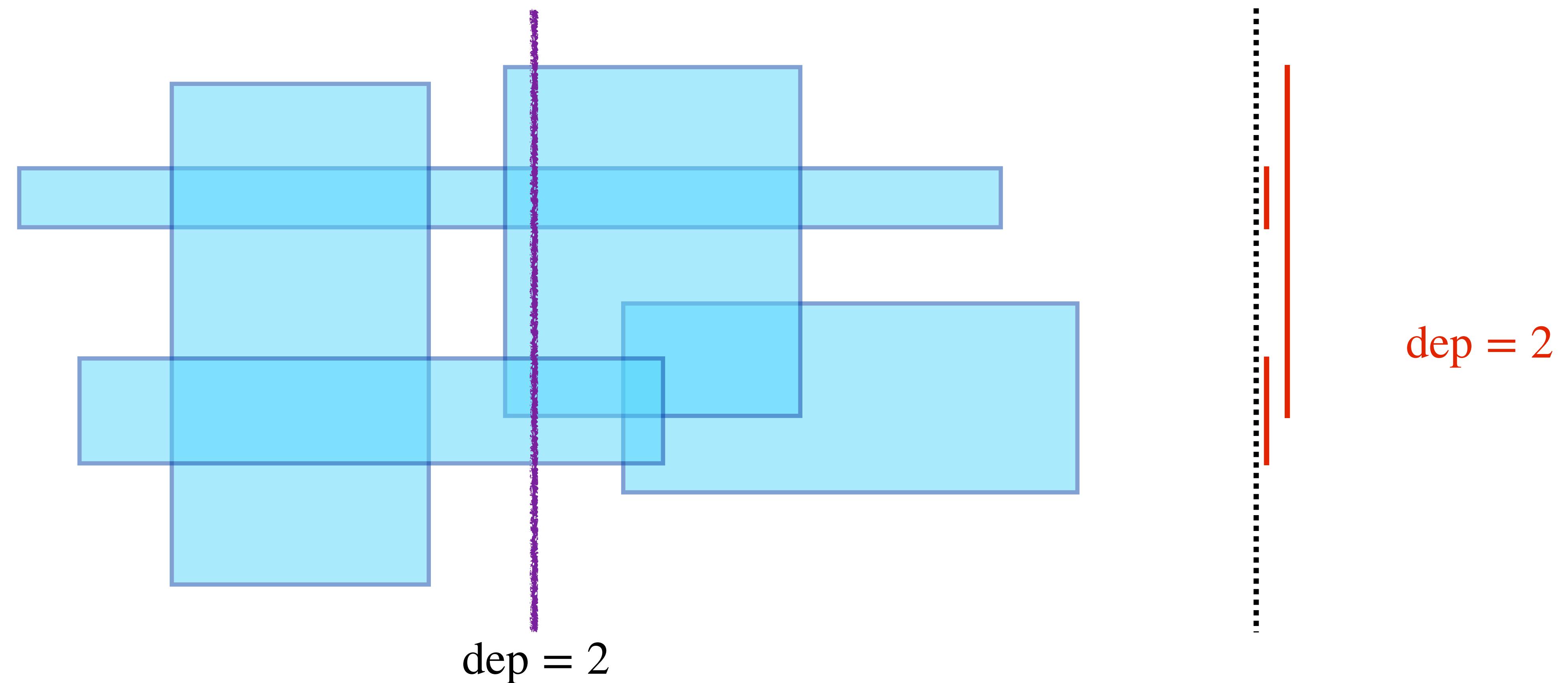
# Warm Up: Sweep-Line Algorithm for Rectangles

For rectangles:  $O(n \log n)$ -time algorithm via sweep line [Imai and Asano 1977]



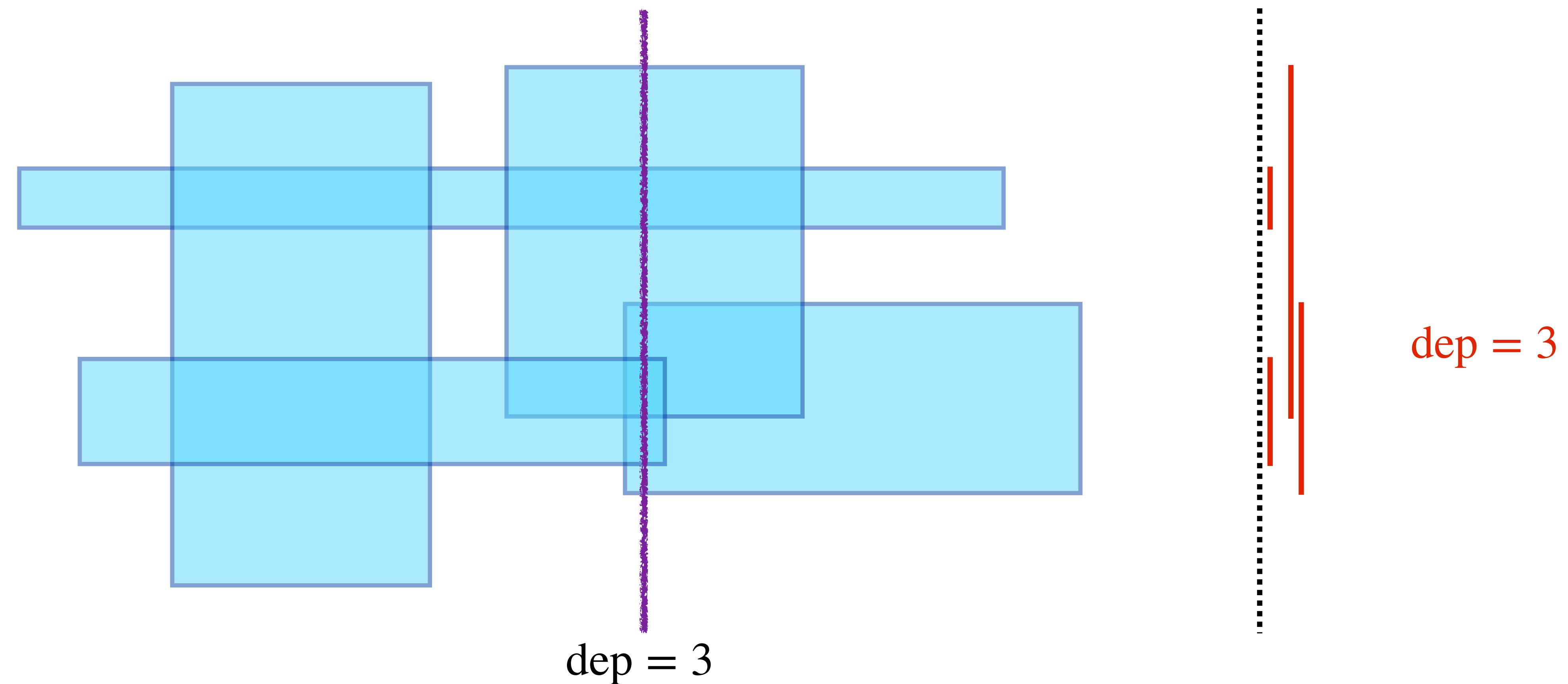
# Warm Up: Sweep-Line Algorithm for Rectangles

For rectangles:  $O(n \log n)$ -time algorithm via sweep line [Imai and Asano 1977]



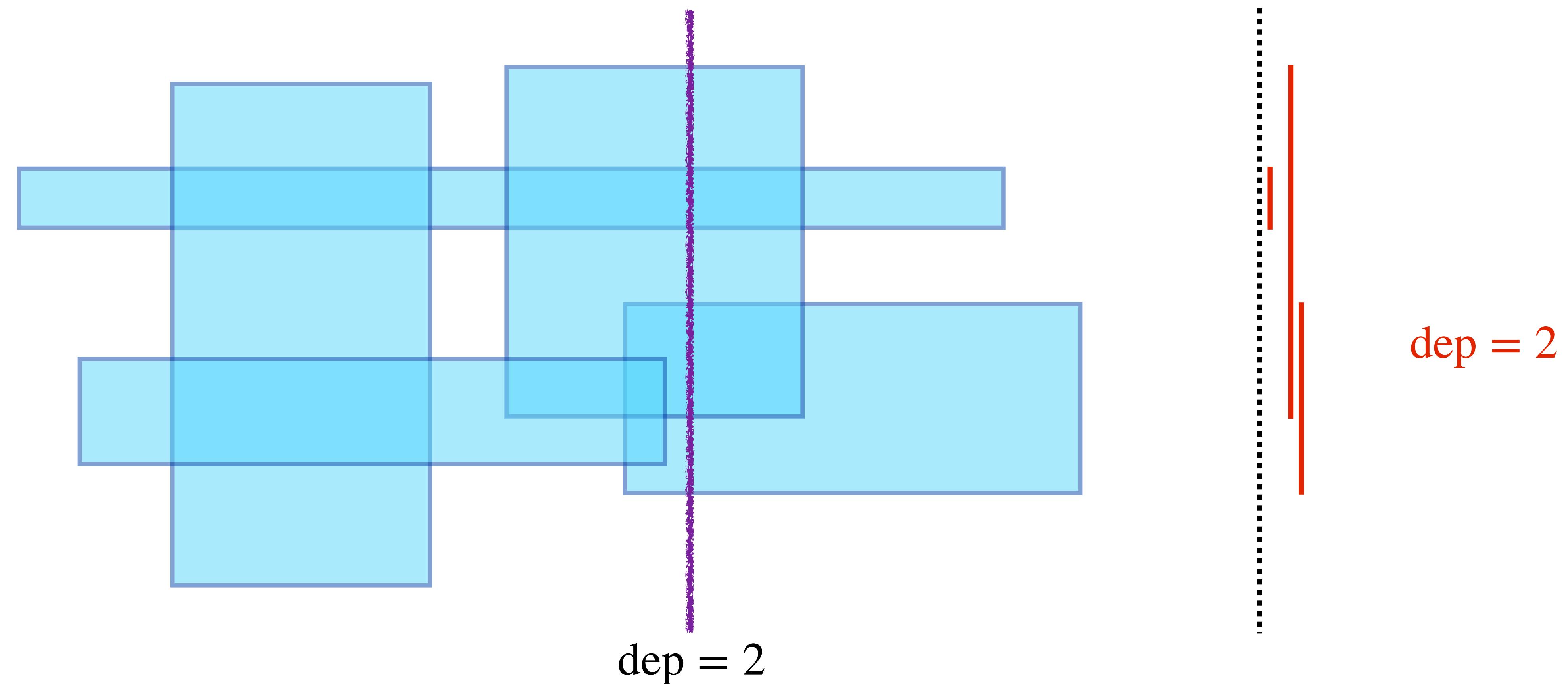
# Warm Up: Sweep-Line Algorithm for Rectangles

For rectangles:  $O(n \log n)$ -time algorithm via sweep line [Imai and Asano 1977]



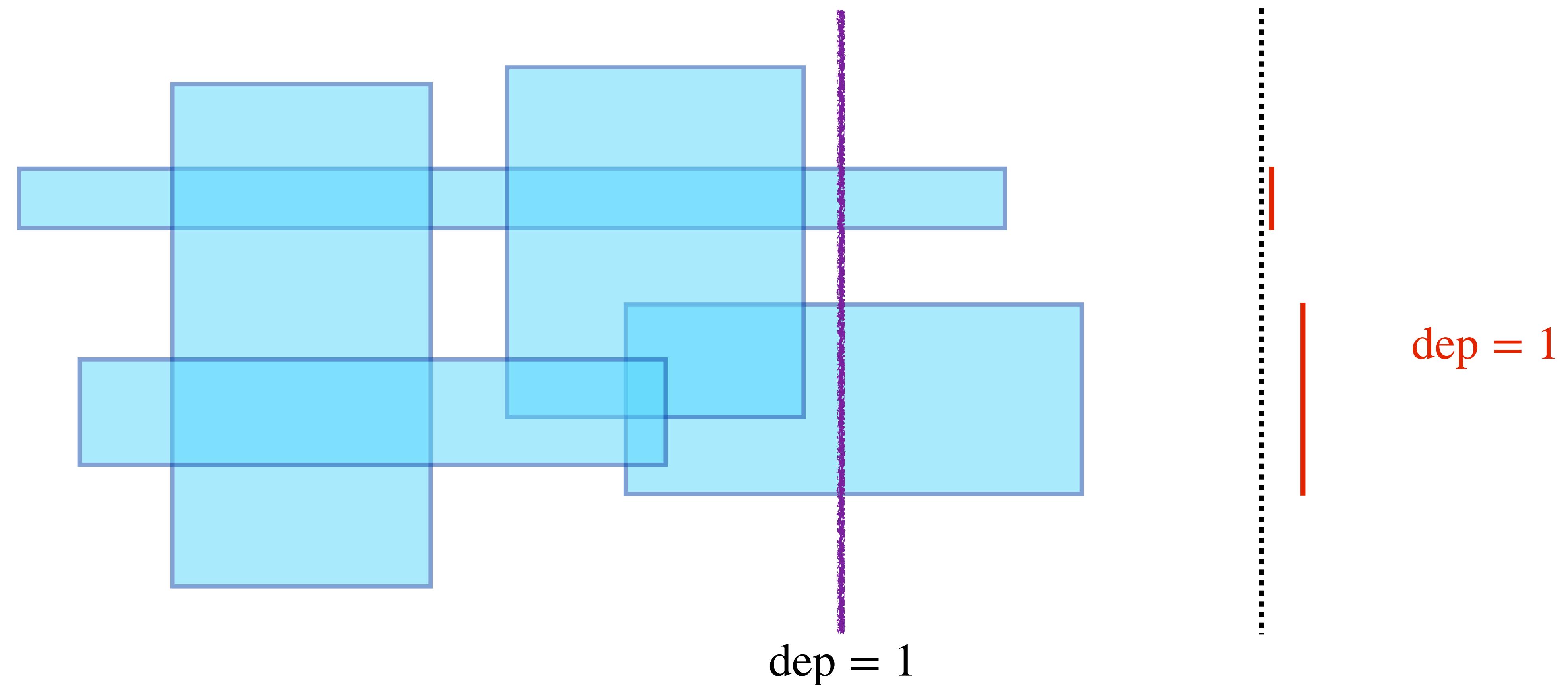
# Warm Up: Sweep-Line Algorithm for Rectangles

For rectangles:  $O(n \log n)$ -time algorithm via sweep line [Imai and Asano 1977]



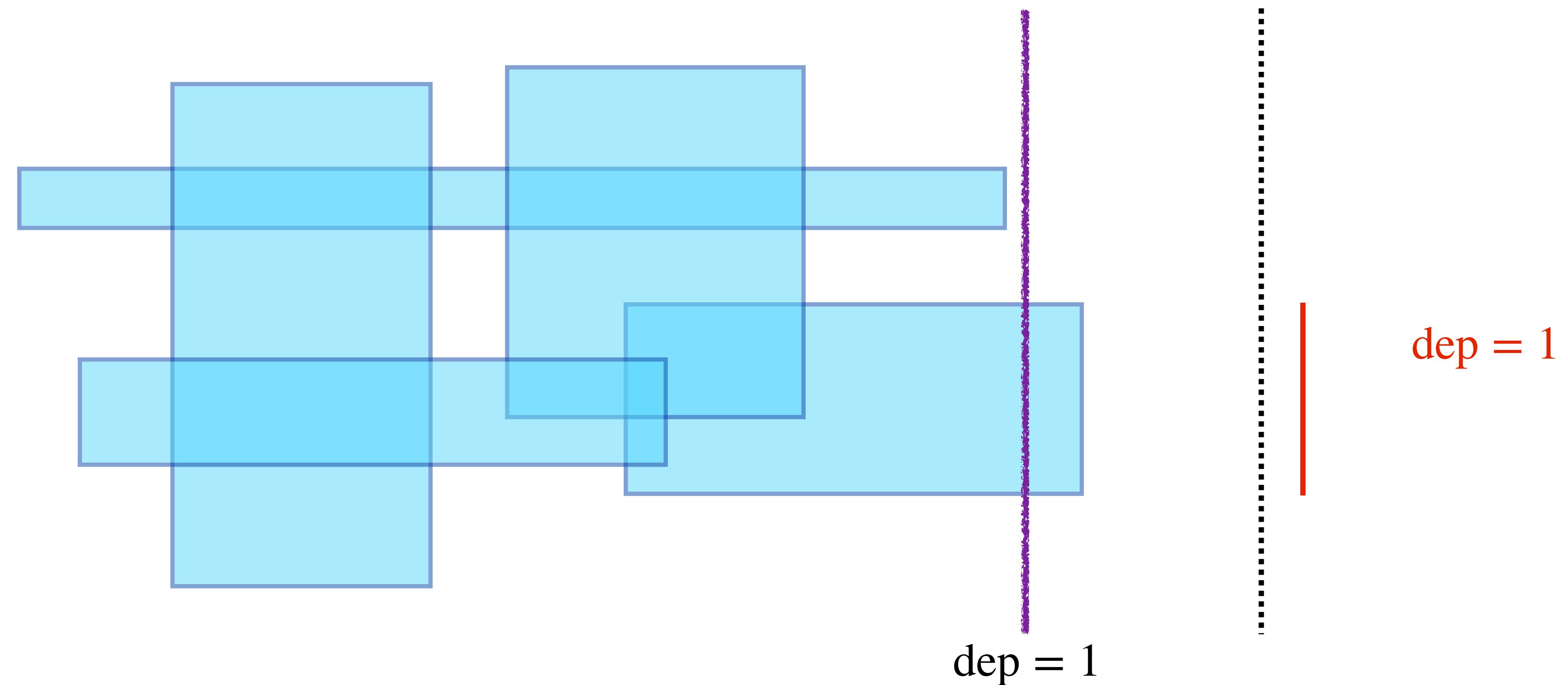
# Warm Up: Sweep-Line Algorithm for Rectangles

For rectangles:  $O(n \log n)$ -time algorithm via sweep line [Imai and Asano 1977]



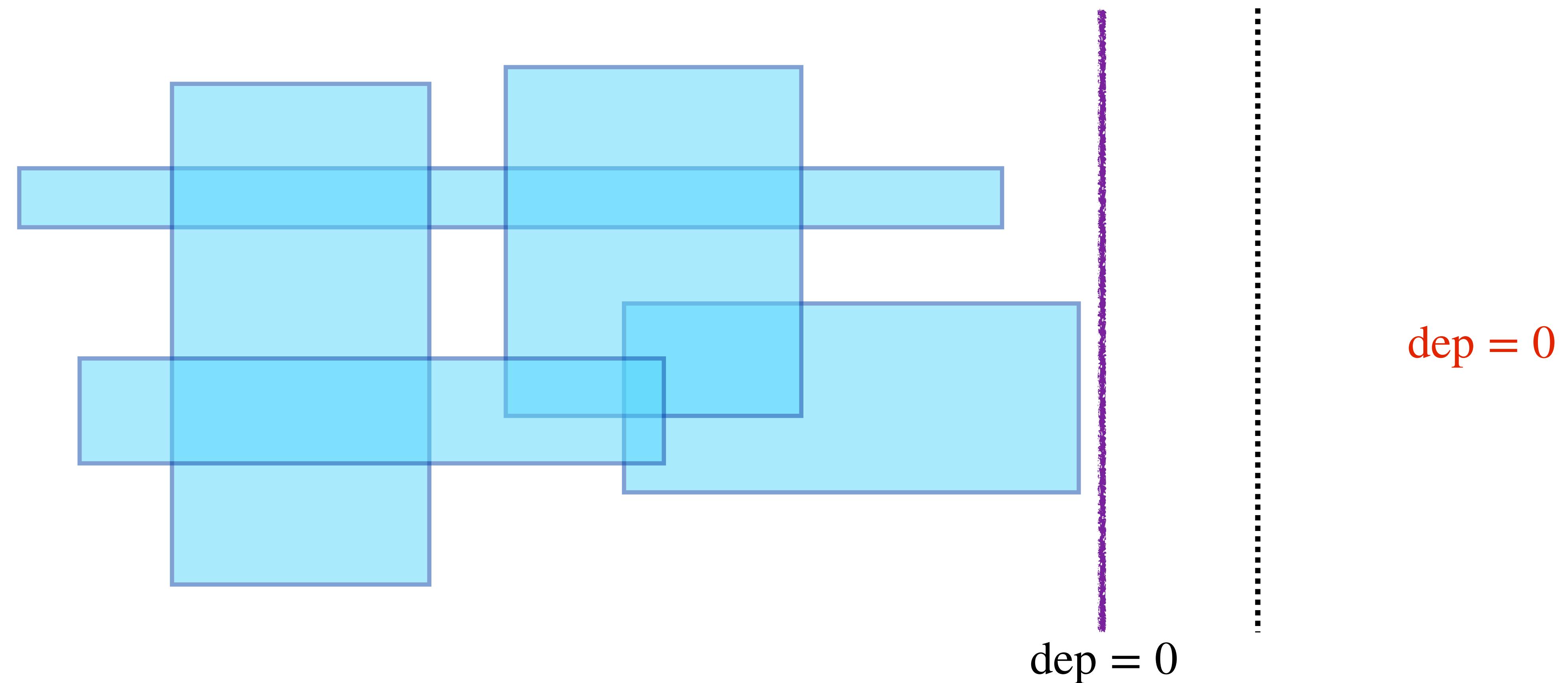
# Warm Up: Sweep-Line Algorithm for Rectangles

For rectangles:  $O(n \log n)$ -time algorithm via sweep line [Imai and Asano 1977]



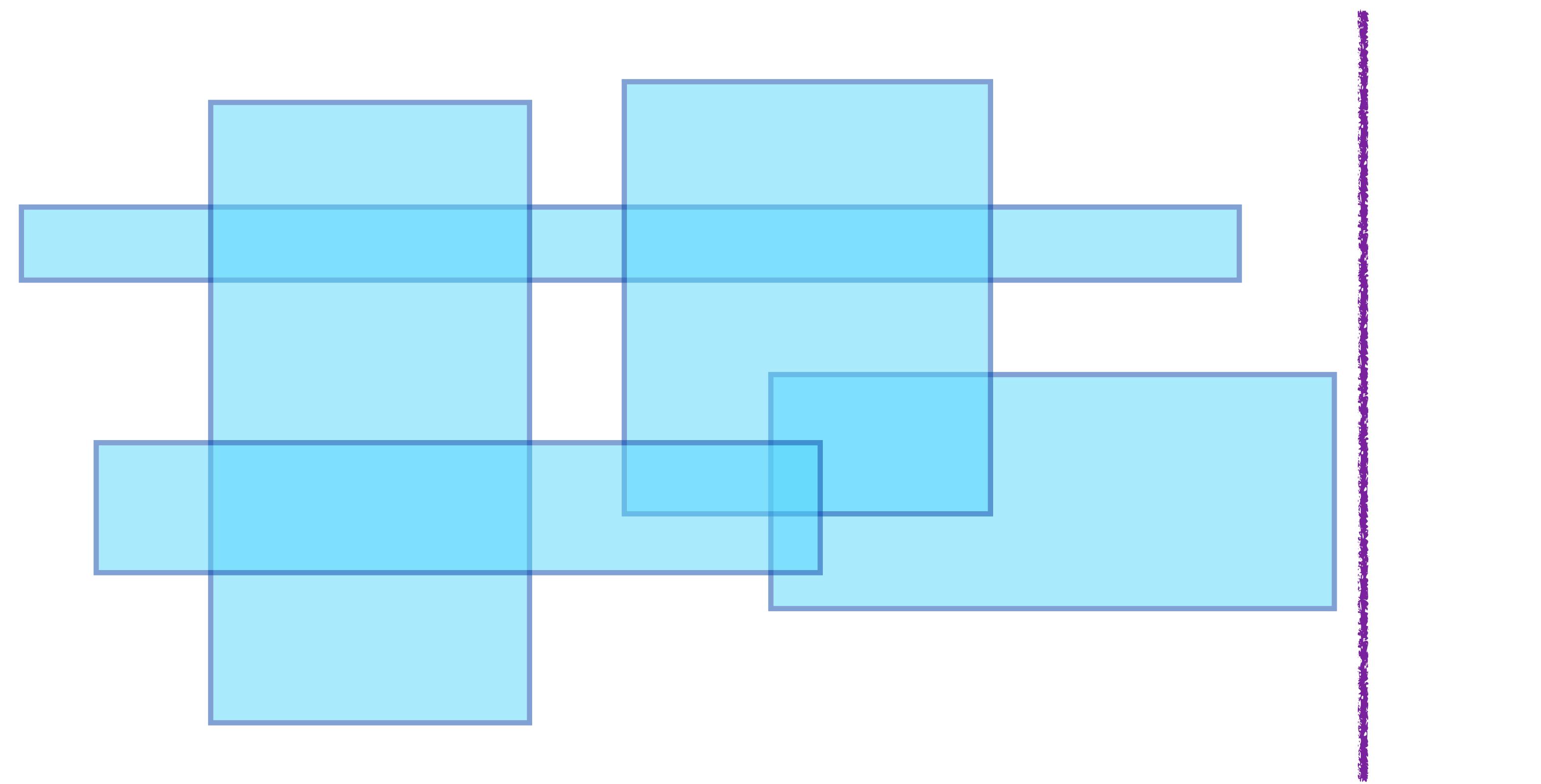
# Warm Up: Sweep-Line Algorithm for Rectangles

For rectangles:  $O(n \log n)$ -time algorithm via sweep line [Imai and Asano 1977]



# Warm Up: Sweep-Line Algorithm for Rectangles

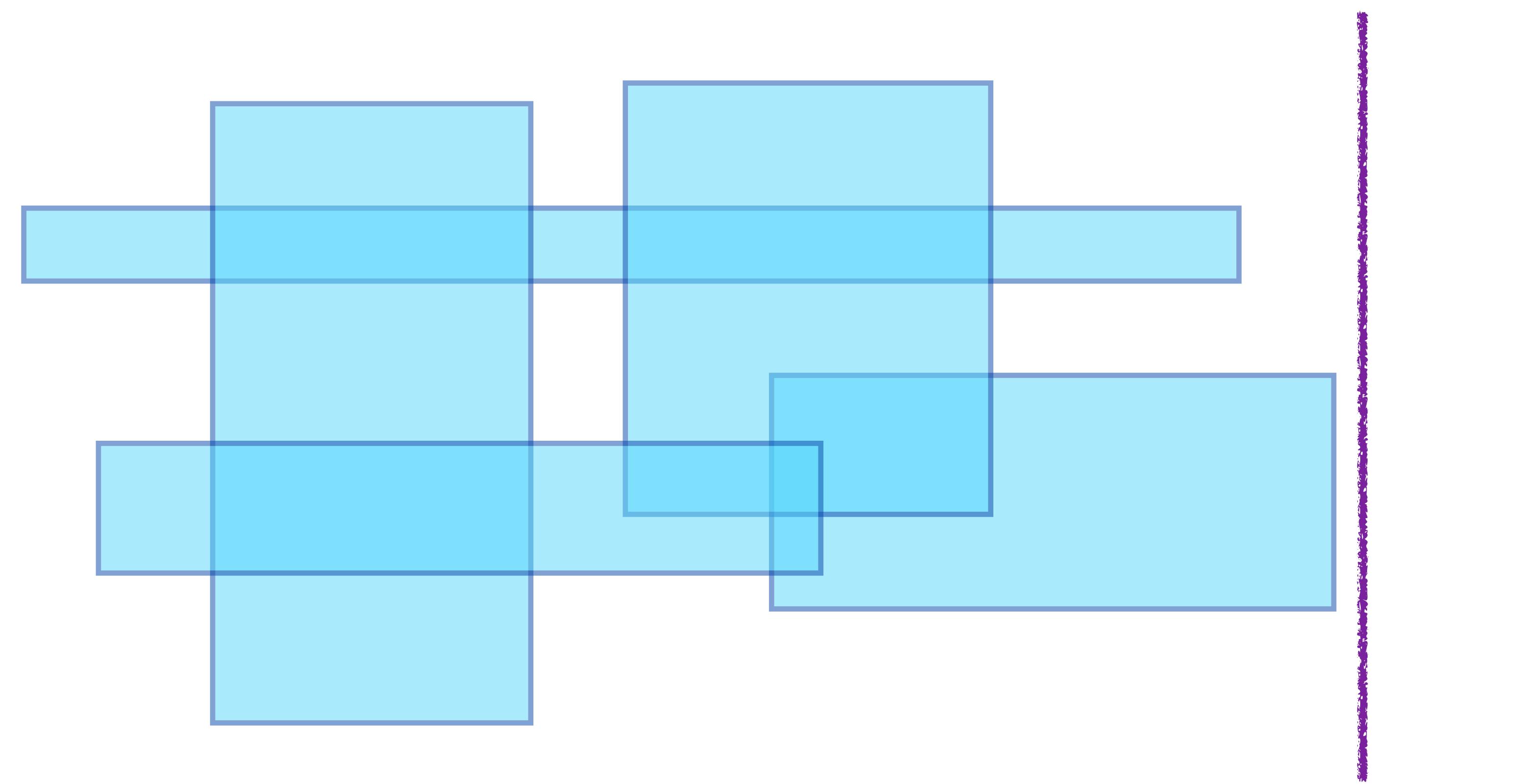
For rectangles:  $O(n \log n)$ -time algorithm via **sweep line** [Imai and Asano 1977]



Key ingredient: **dynamic** data structure for intervals with  $O(\log n)$  update time

# Warm Up: Sweep-Line Algorithm for Rectangles

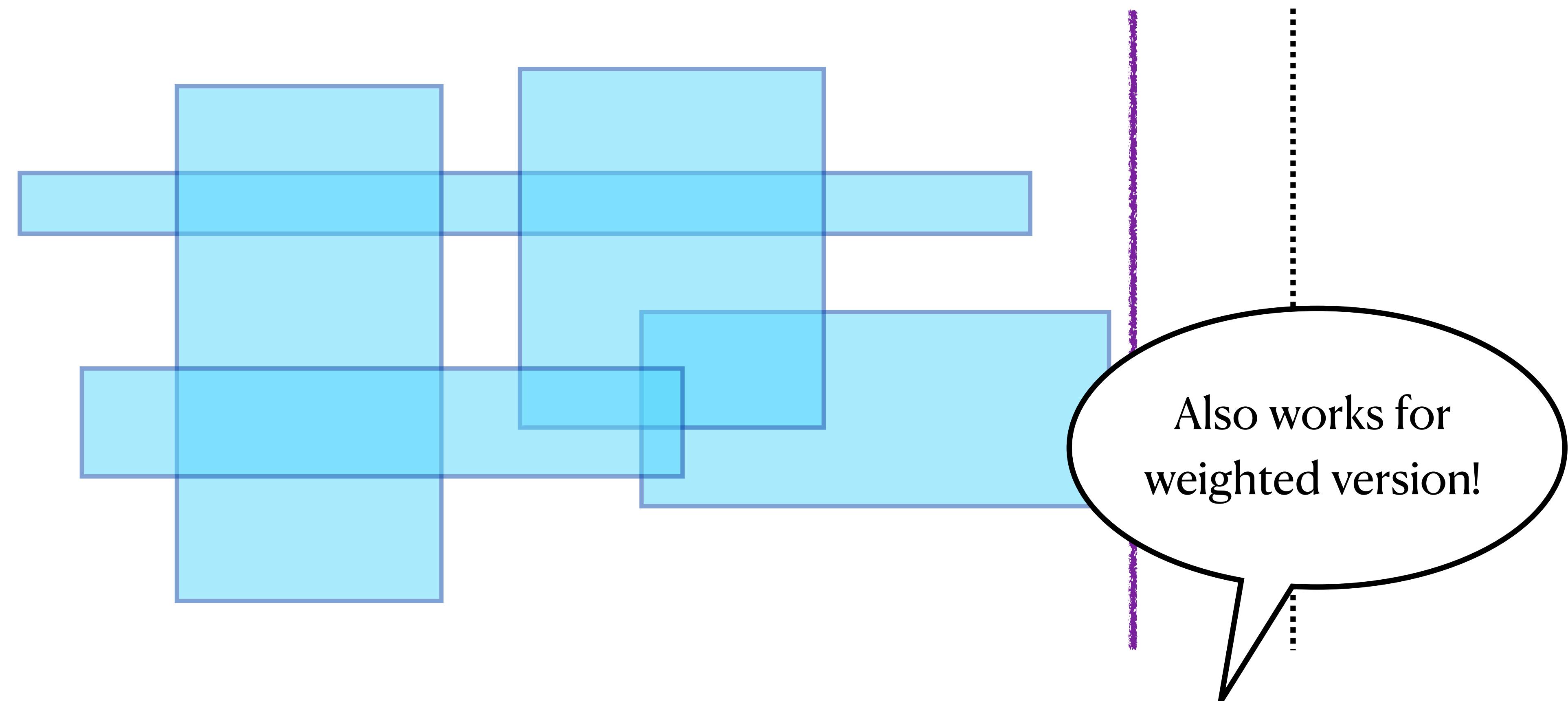
For rectangles:  $O(n \log n)$ -time algorithm via [sweep line \[Imai and Asano 1977\]](#)



Key ingredient: **dynamic** data structure for intervals with  $O(\log n)$  update time  
 $d$ -dim.  $O(n^c)$  update time yields  $(d + 1)$ -dim.  $O(n^{c+1})$  time algorithm

# Warm Up: Sweep-Line Algorithm for Rectangles

For rectangles:  $O(n \log n)$ -time algorithm via sweep line [Imai and Asano 1977]



Key ingredient: **dynamic** data structure for intervals with  $O(\log n)$  update time  
 $d$ -dim.  $O(n^c)$  update time yields  $(d + 1)$ -dim.  $O(n^{c+1})$  time algorithm

# **Rectangle: Fast Update via Partition**

# Rectangle: Fast Update via Partition

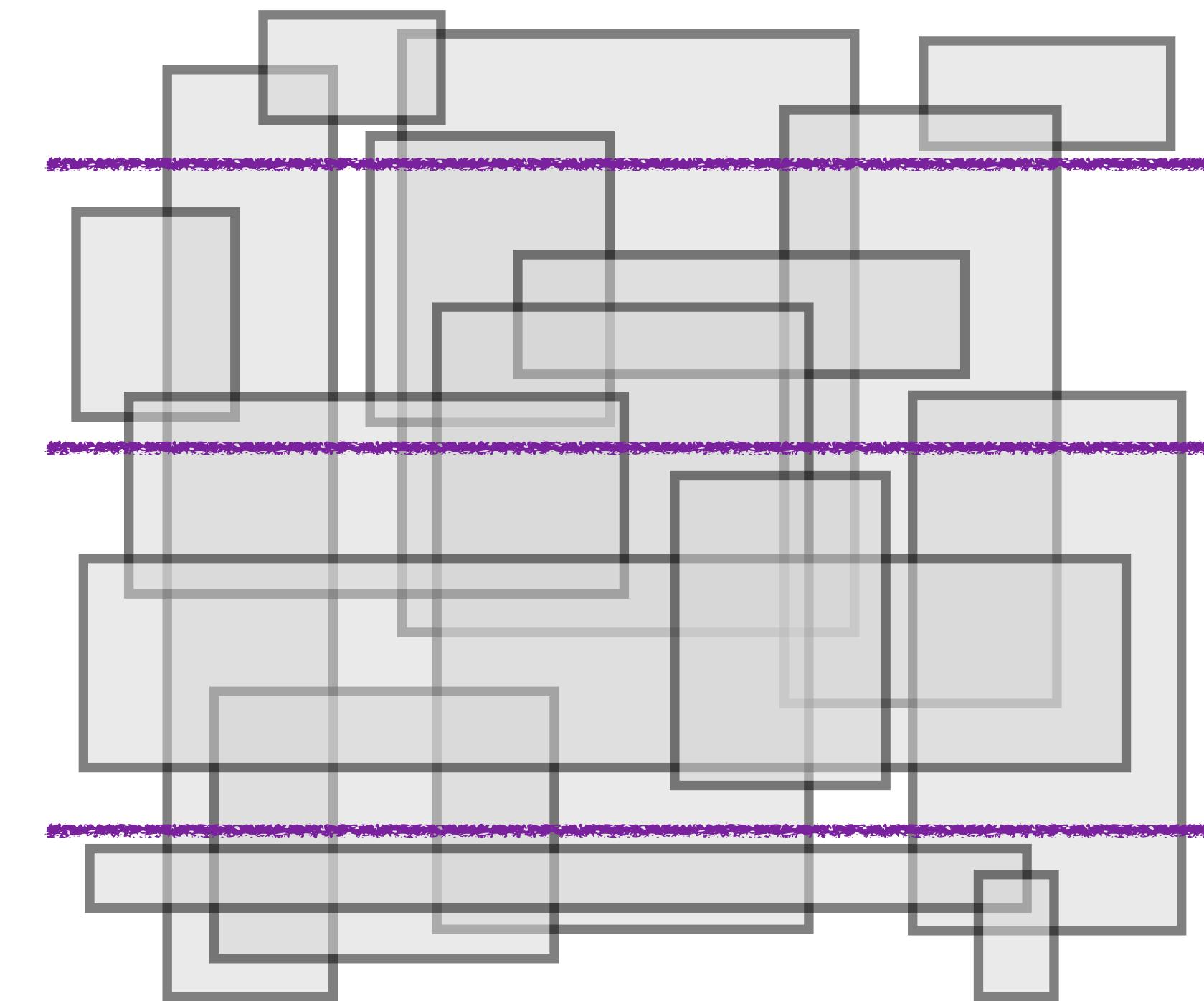
## Partition the plane

- Partition the plane into  $r$  slabs
- Each slab have  $O(n/r)$  corners of rectangles

# Rectangle: Fast Update via Partition

## Partition the plane

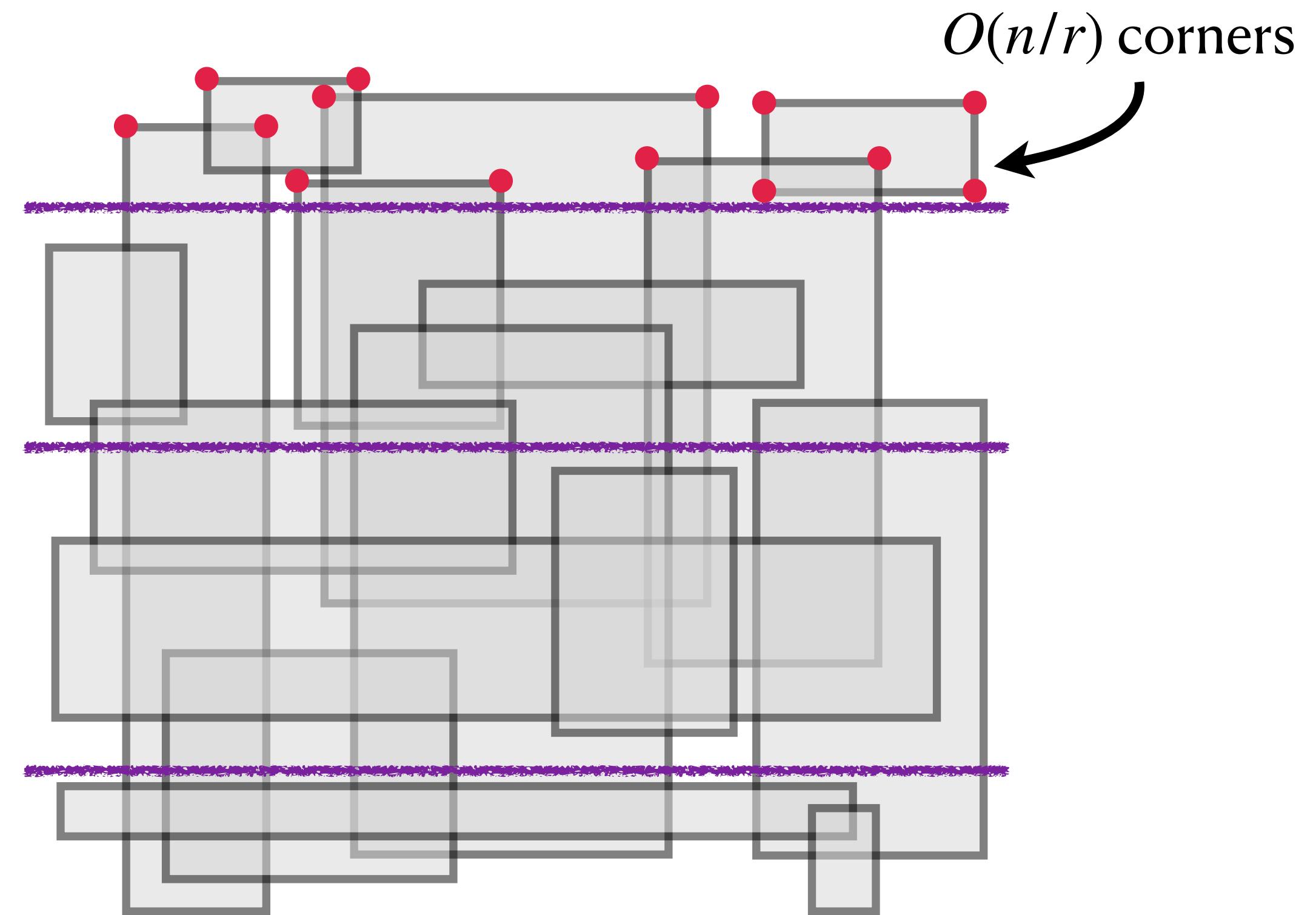
- Partition the plane into  $r$  slabs
- Each slab have  $O(n/r)$  corners of rectangles



# Rectangle: Fast Update via Partition

## Partition the plane

- Partition the plane into  $r$  slabs
- Each slab have  $O(n/r)$  corners of rectangles



# Rectangle: Fast Update via Partition

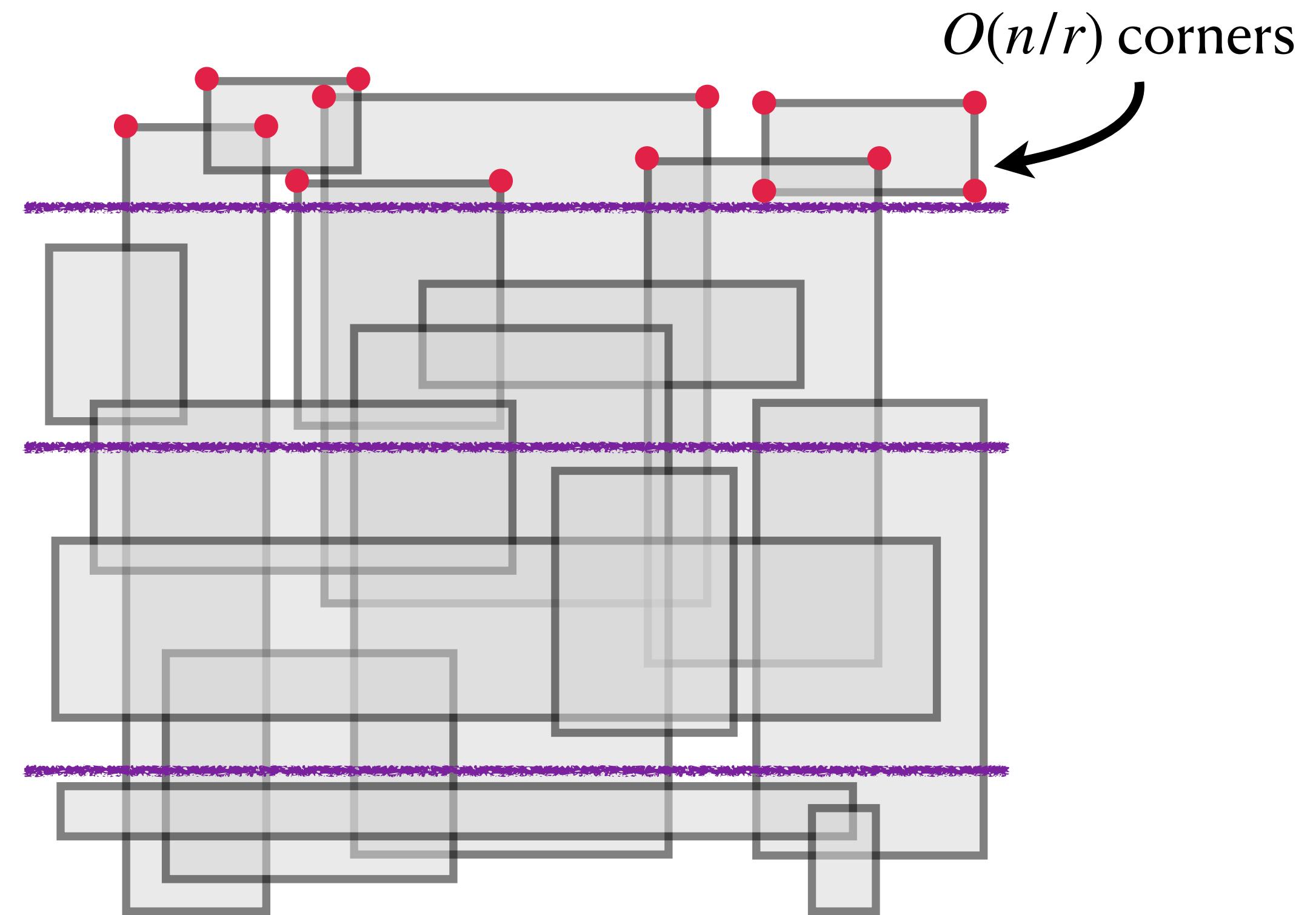
## Partition the plane

- Partition the plane into  $r$  slabs
- Each slab have  $O(n/r)$  corners of rectangles

## Update in each slab

Maintain the max depth in each slab

Return the largest one



# Rectangle: Fast Update via Partition

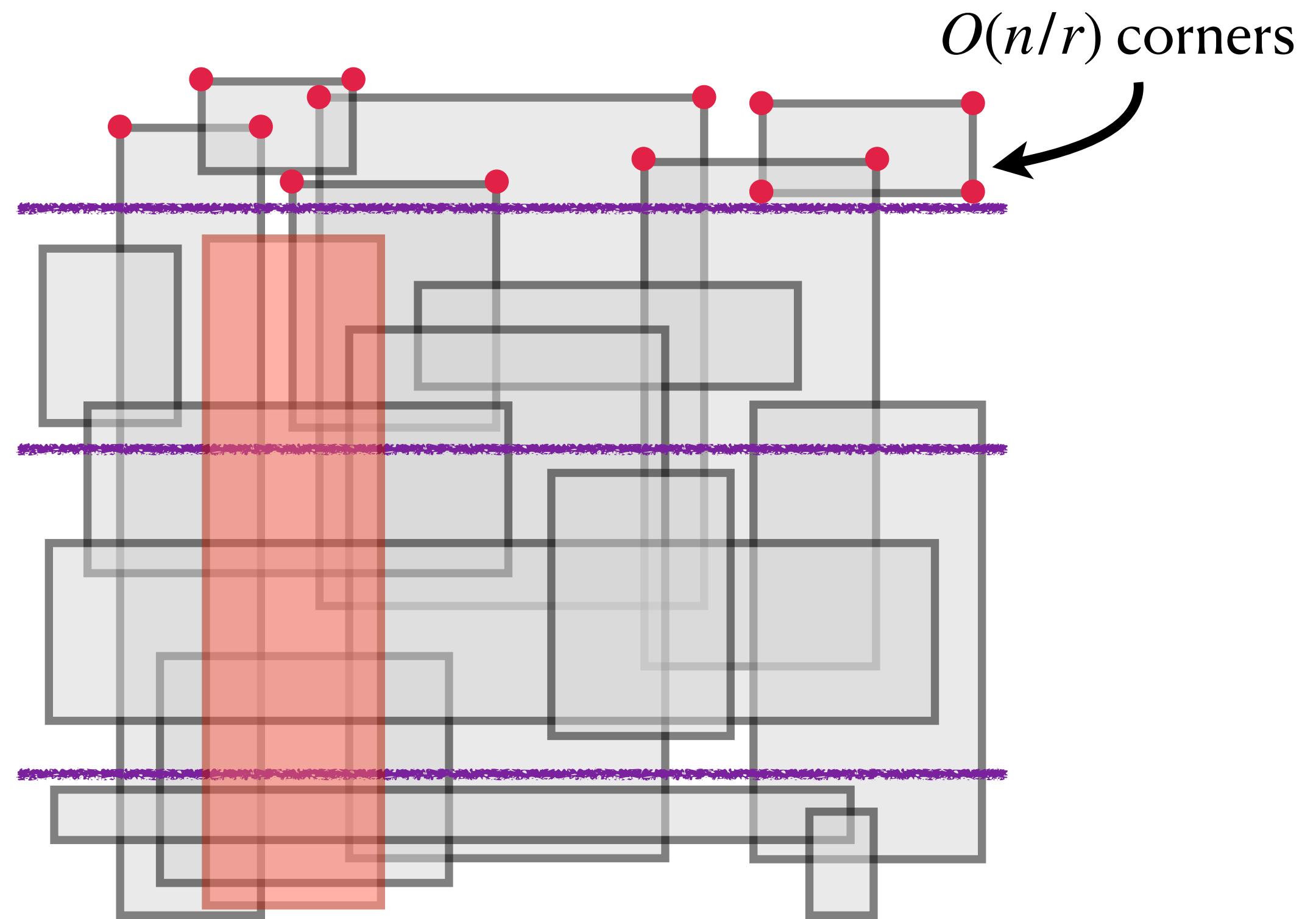
## Partition the plane

- Partition the plane into  $r$  slabs
- Each slab have  $O(n/r)$  corners of rectangles

## Update in each slab

Maintain the max depth in each slab

Return the largest one



# Rectangle: Fast Update via Partition

## Partition the plane

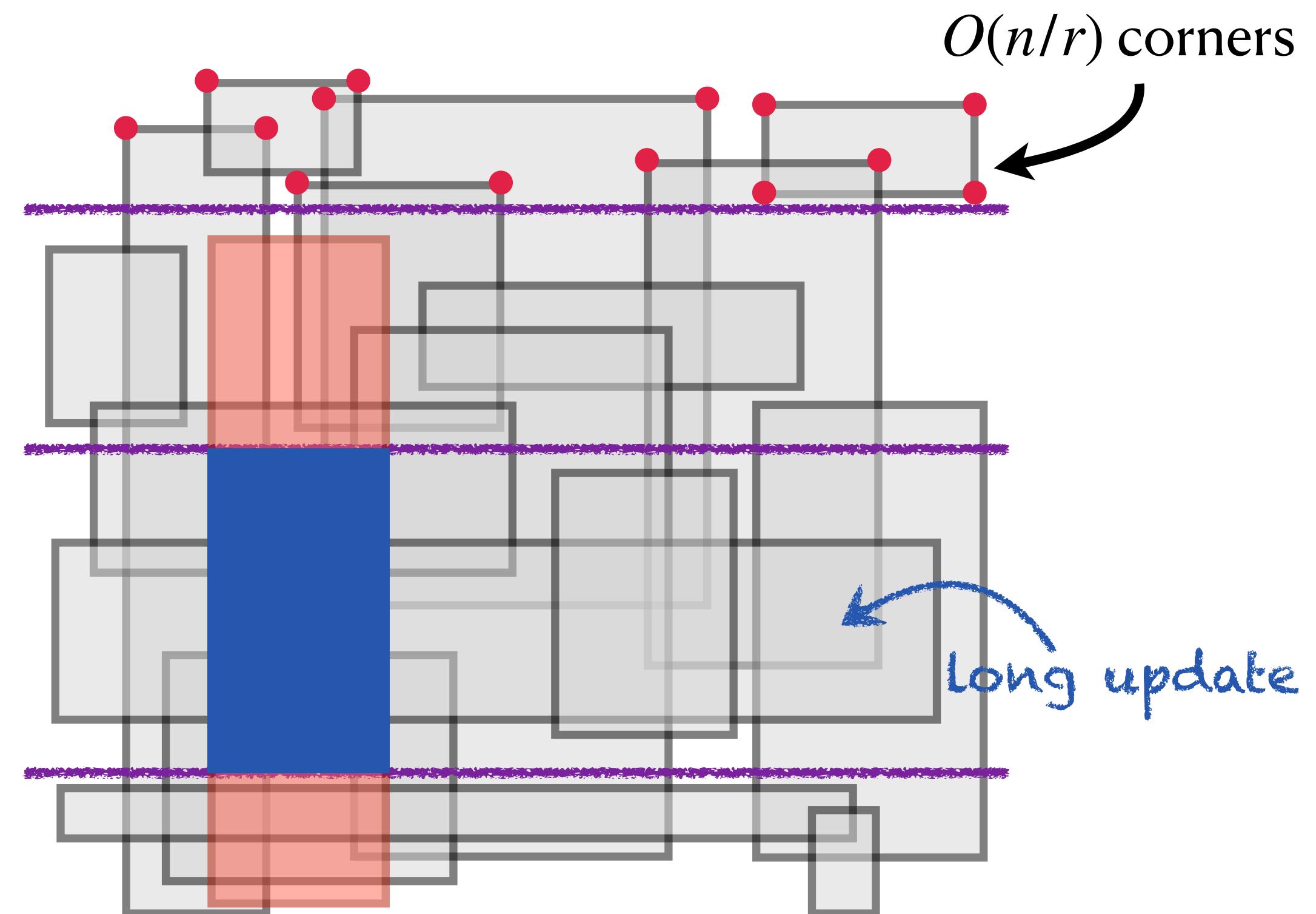
- Partition the plane into  $r$  slabs
- Each slab have  $O(n/r)$  corners of rectangles

## Update in each slab

Maintain the max depth in each slab

Return the largest one

“Long” update



# Rectangle: Fast Update via Partition

## Partition the plane

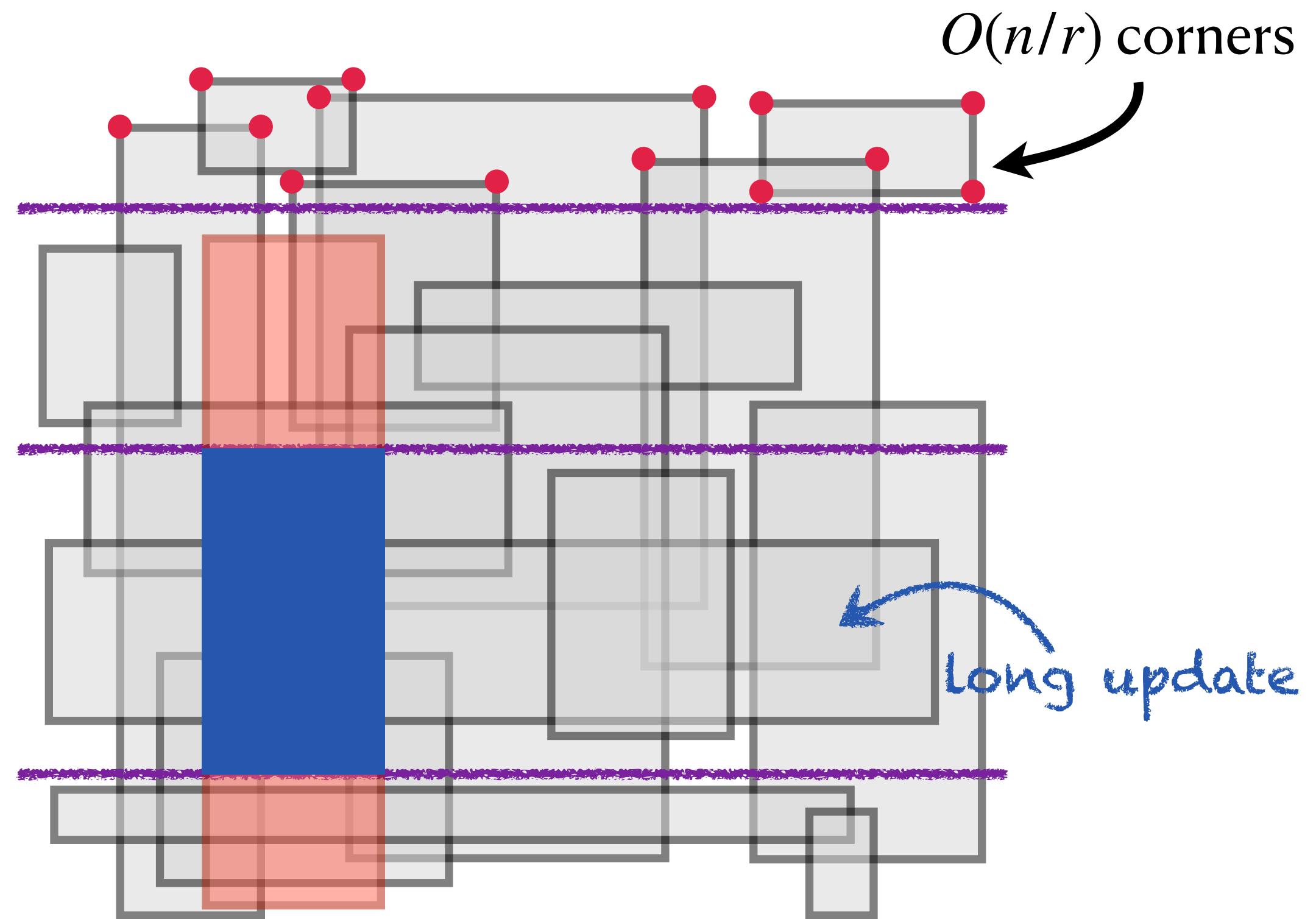
- Partition the plane into  $r$  slabs
- Each slab have  $O(n/r)$  corners of rectangles

## Update in each slab

Maintain the max depth in each slab

Return the largest one

“Long” update 😊 easy:  $O(\log n)$   
😈 many:  $O(r)$



# Rectangle: Fast Update via Partition

## Partition the plane

- Partition the plane into  $r$  slabs
- Each slab have  $O(n/r)$  corners of rectangles

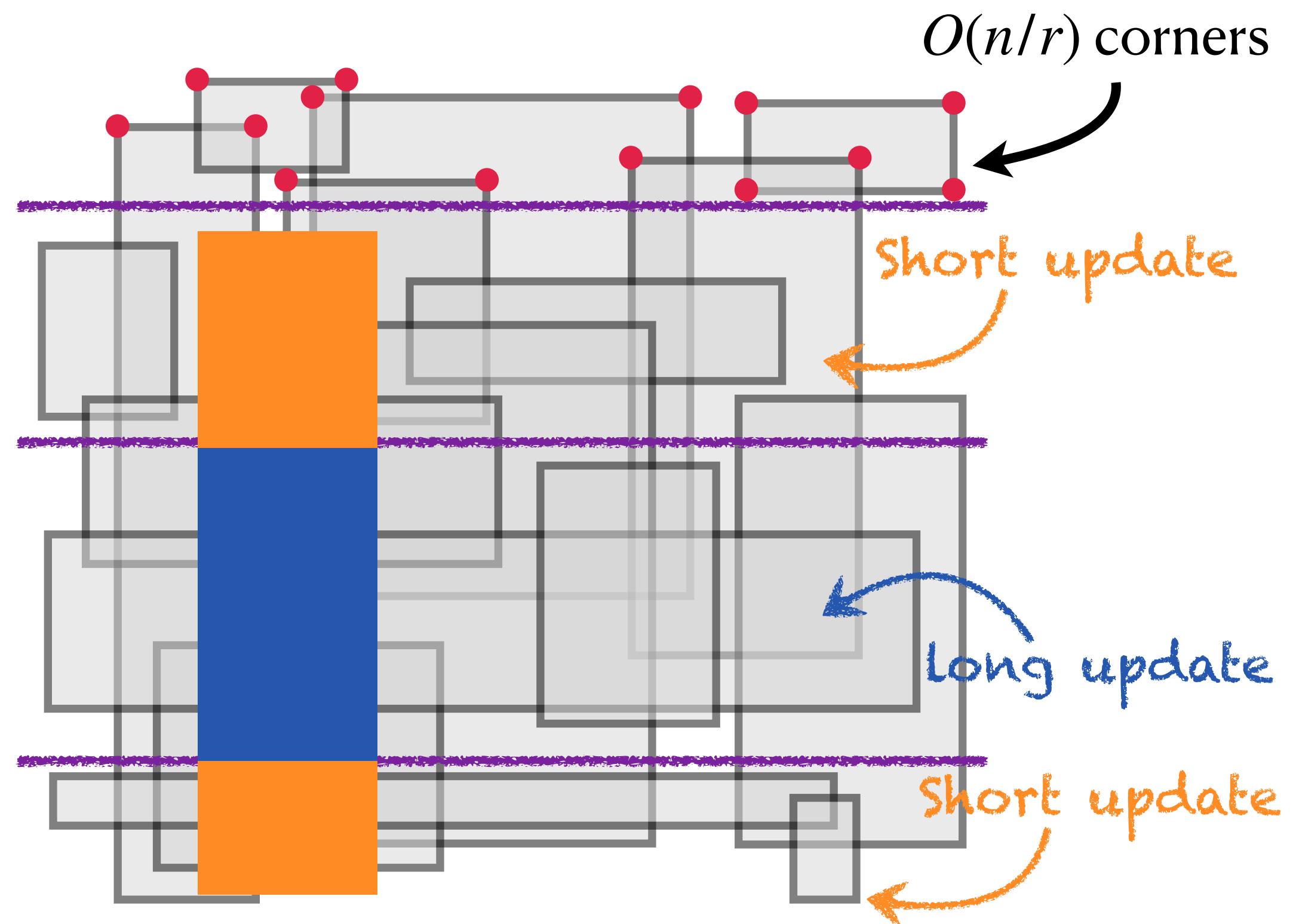
## Update in each slab

Maintain the max depth in each slab

Return the largest one

“Long” update 😊 easy:  $O(\log n)$   
😈 many:  $O(r)$

“Short” update



# Rectangle: Fast Update via Partition

## Partition the plane

- Partition the plane into  $r$  slabs
- Each slab have  $O(n/r)$  corners of rectangles

## Update in each slab

Maintain the max depth in each slab

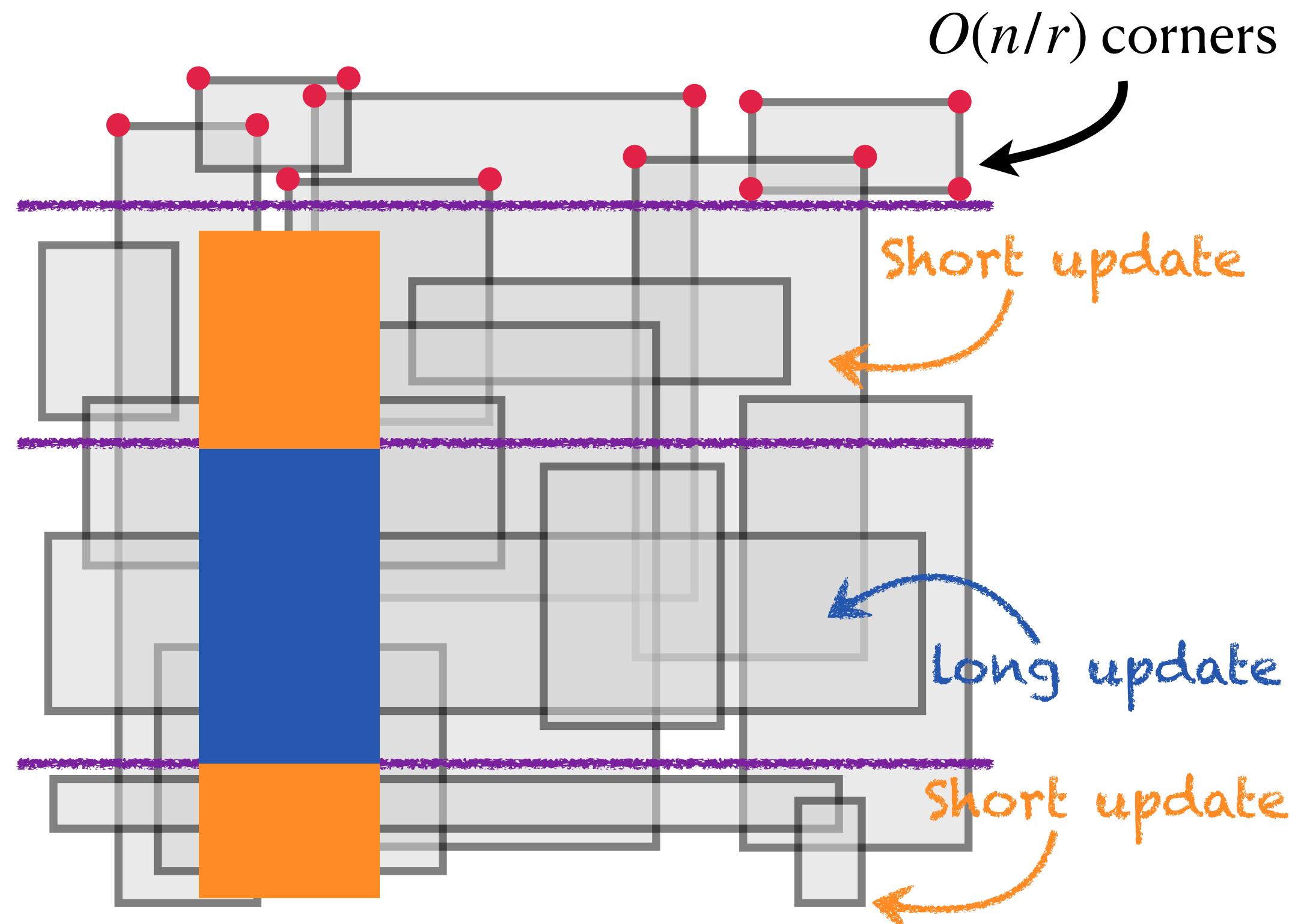
Return the largest one

“Long” update 😊 easy:  $O(\log n)$

😈 many:  $O(r)$

“Short” update 😈 hard:  $O((n/r)\log n)$

😊 rare:  $\leq 2$



# Rectangle: Fast Update via Partition

## Partition the plane

- Partition the plane into  $r$  slabs
- Each slab have  $O(n/r)$  corners of rectangles

## Update in each slab

Maintain the max depth in each slab

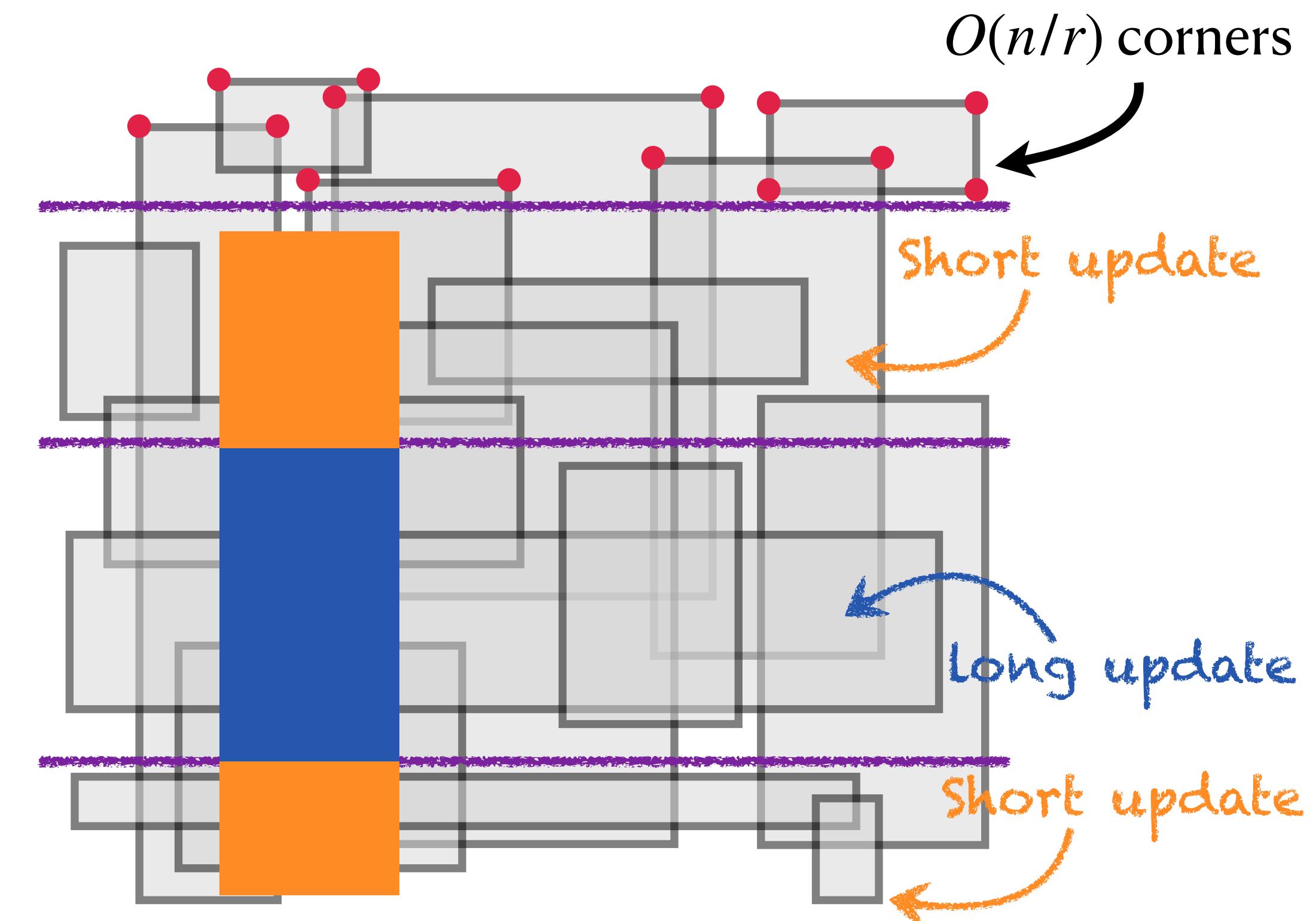
Return the largest one

“Long” update 😊 easy:  $O(\log n)$

😈 many:  $O(r)$

“Short” update 😈 hard:  $O((n/r)\log n)$

😊 rare:  $\leq 2$

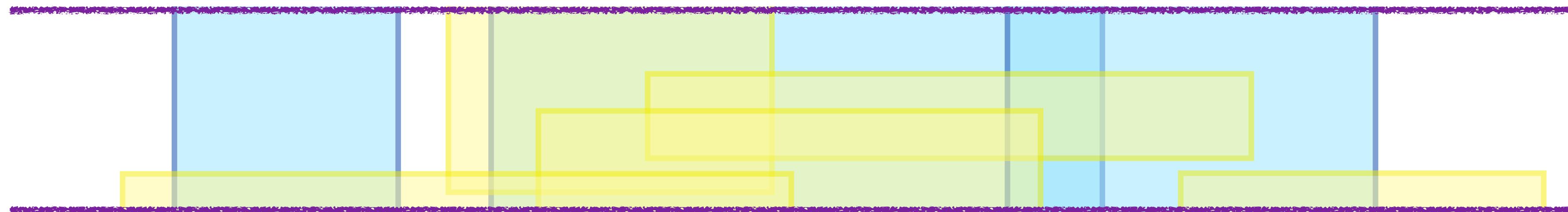


## Total update time

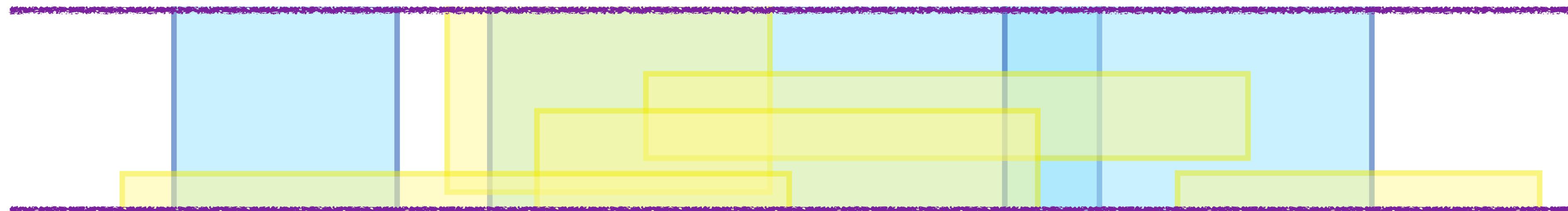
$$O\left(2 \cdot \frac{n}{r} \log n + r \cdot \log n\right) = \tilde{O}\left(\sqrt{n}\right)$$

when  $r = \sqrt{n}$

# Details of Slab Data Structure

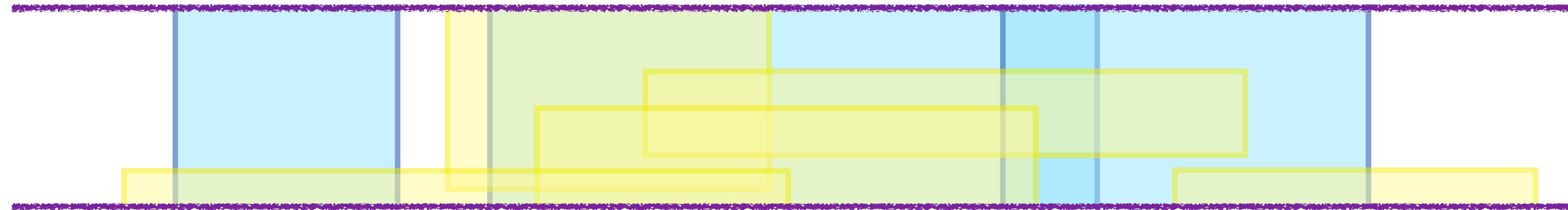


# Details of Slab Data Structure



**Reduce to intervals**

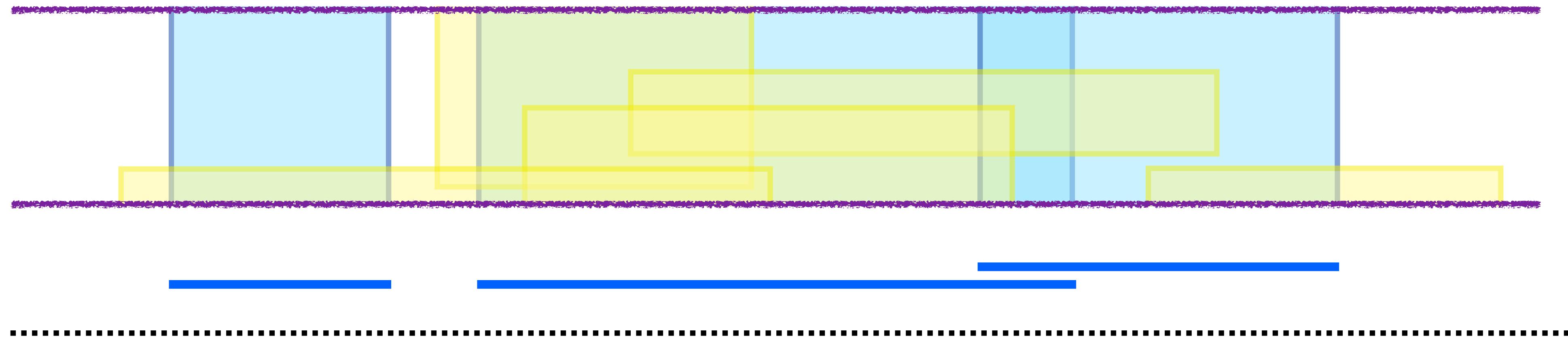
# Details of Slab Data Structure



## Reduce to intervals

$O(n)$  long rectangles       $\xrightarrow[\text{directly}]{O(n) \text{ time}}$        $O(n)$  unweight intervals

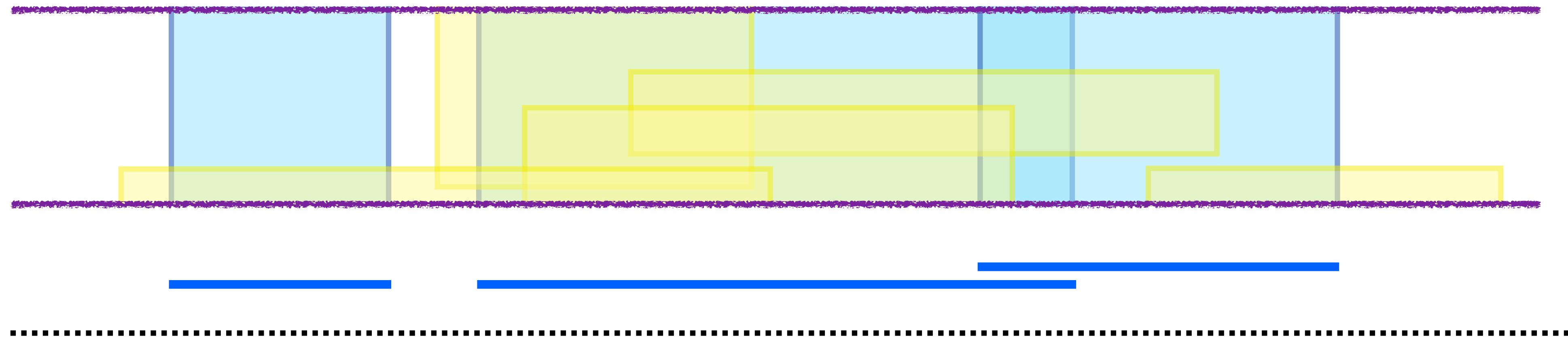
# Details of Slab Data Structure



**Reduce to intervals**

$O(n)$  long rectangles  $\xrightarrow[\mathcal{O}(n) \text{ time}]{\text{directly}} O(n)$  unweight intervals

# Details of Slab Data Structure



## Reduce to intervals

$O(n)$  long rectangles

directly  
 $O(n)$  time

$O(n)$  unweight intervals

$O\left(\frac{n}{r}\right)$  short rectangles

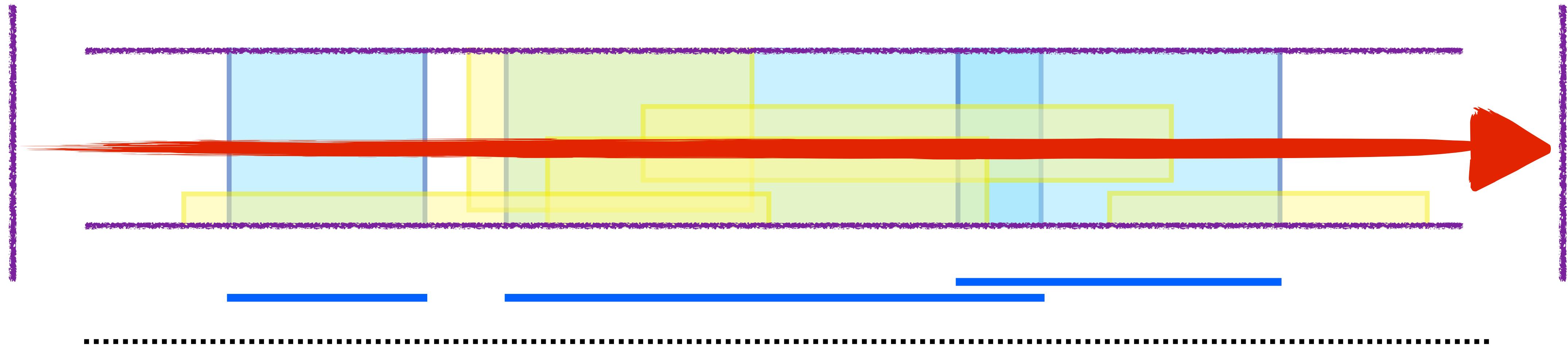
sweep line  
 $O\left(\frac{n}{r} \log \frac{n}{r}\right)$  time

$O(n/r)$  weighted intervals

interval: two adjacent boundaries

weight: max dep w.r.t. short rectangles

# Details of Slab Data Structure



## Reduce to intervals

$O(n)$  long rectangles

directly  
 $O(n)$  time

$O(n)$  unweight intervals

$O\left(\frac{n}{r}\right)$  short rectangles

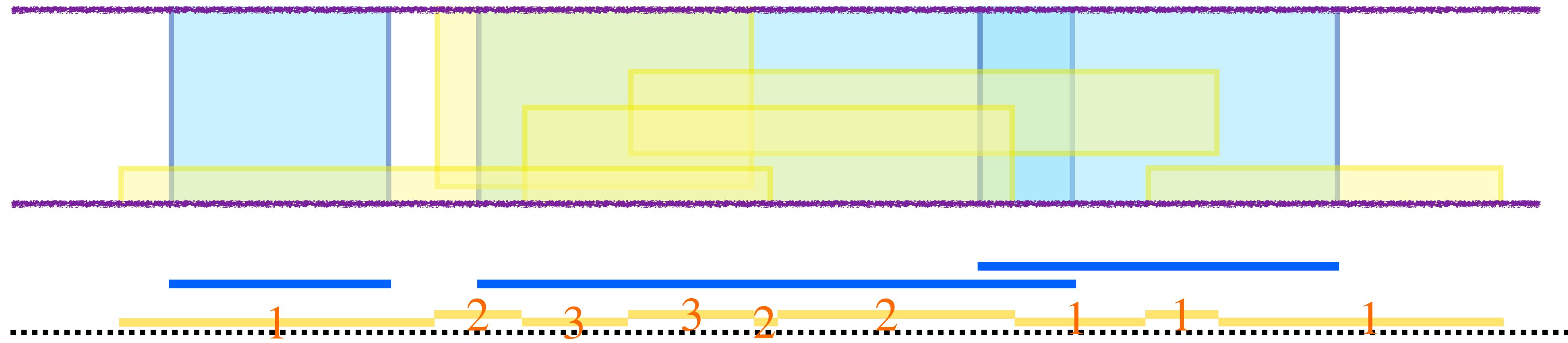
sweep line  
 $O\left(\frac{n}{r} \log \frac{n}{r}\right)$  time

$O(n/r)$  weighted intervals

interval: two adjacent boundaries

weight: max dep w.r.t. short rectangles

# Details of Slab Data Structure



## Reduce to intervals

$O(n)$  long rectangles

$\xrightarrow[\text{directly}]{O(n) \text{ time}}$   $O(n)$  unweight intervals

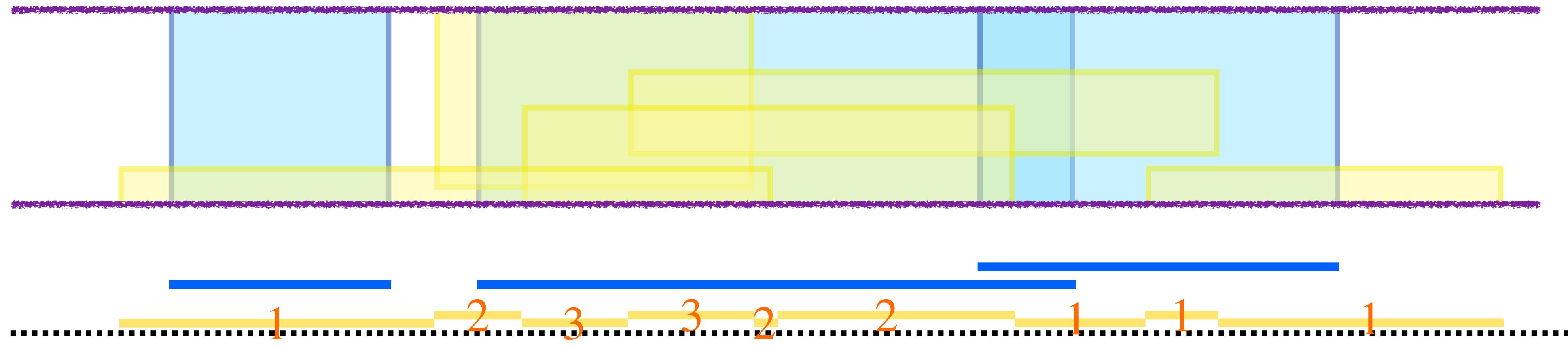
$O\left(\frac{n}{r}\right)$  short rectangles

$\xrightarrow[\text{sweep line}]{O\left(\frac{n}{r} \log \frac{n}{r}\right) \text{ time}}$   $O(n/r)$  weighted intervals

interval: two adjacent boundaries

weight: max dep w.r.t. short rectangles

# Details of Slab Data Structure



## Reduce to intervals

$O(n)$  long rectangles

$\xrightarrow[\text{directly}]{O(n) \text{ time}}$   $O(n)$  unweight intervals

$O\left(\frac{n}{r}\right)$  short rectangles

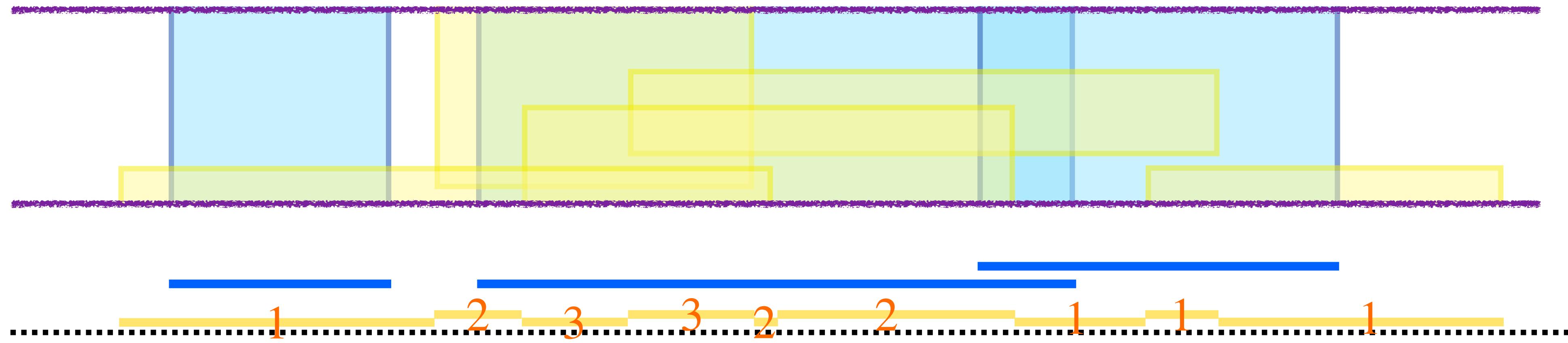
$\xrightarrow[\text{sweep line}]{O\left(\frac{n}{r} \log \frac{n}{r}\right) \text{ time}}$   $O(n/r)$  weighted intervals

interval: two adjacent boundaries

weight: max dep w.r.t. short rectangles

Total preprocessing time:  $O(n \log n)$

# Details of Slab Data Structure



## Reduce to intervals

$O(n)$  long rectangles

directly  
 $O(n)$  time

$O(n)$  unweight intervals

$O\left(\frac{n}{r}\right)$  short rectangles

sweep line  
 $O\left(\frac{n}{r} \log \frac{n}{r}\right)$  time

$O(n/r)$  weighted intervals

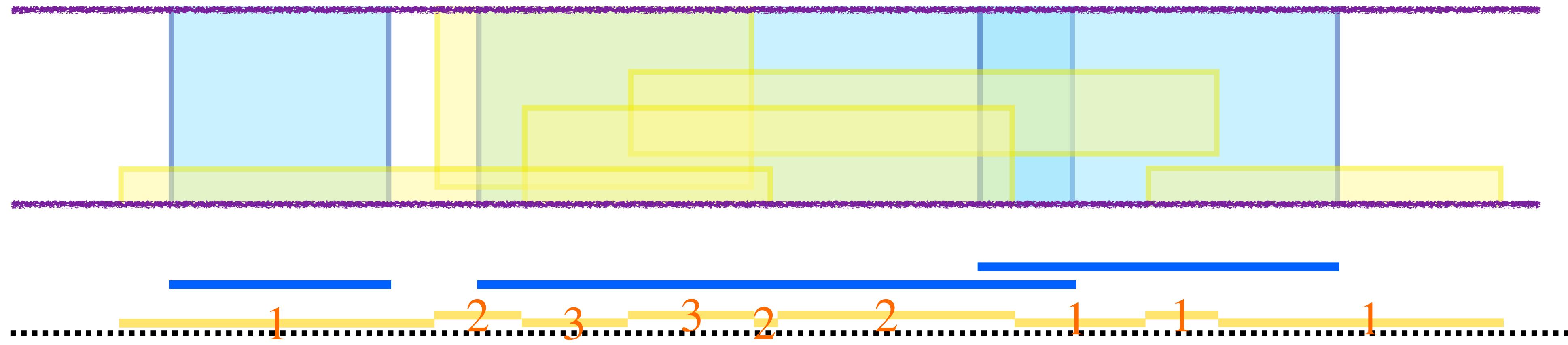
interval: two adjacent boundaries

weight: max dep w.r.t. short rectangles

Total preprocessing time:  $O(n \log n)$

## Update

# Details of Slab Data Structure



## Reduce to intervals

$O(n)$  long rectangles

directly  
 $O(n)$  time

$O(n)$  unweight intervals

$O\left(\frac{n}{r}\right)$  short rectangles

sweep line  
 $O\left(\frac{n}{r} \log \frac{n}{r}\right)$  time

$O(n/r)$  weighted intervals

interval: two adjacent boundaries

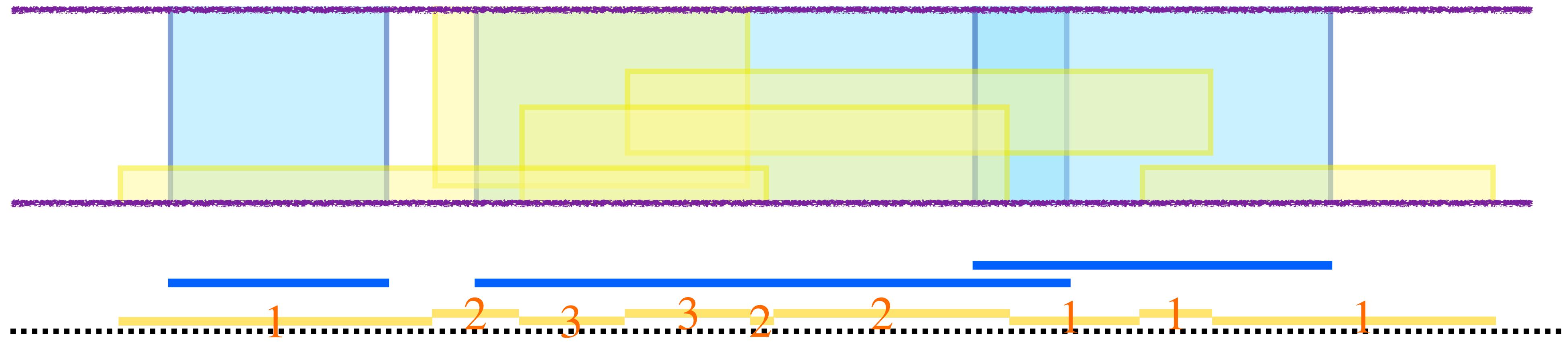
weight: max dep w.r.t. short rectangles

Total preprocessing time:  $O(n \log n)$

## Update

Long update:  $O(\log n)$

# Details of Slab Data Structure



## Reduce to intervals

$O(n)$  long rectangles

directly  
 $O(n)$  time

$O(n)$  unweight intervals

$O\left(\frac{n}{r}\right)$  short rectangles

sweep line  
 $O\left(\frac{n}{r} \log \frac{n}{r}\right)$  time

$O(n/r)$  weighted intervals

interval: two adjacent boundaries

weight: max dep w.r.t. short rectangles

Total preprocessing time:  $O(n \log n)$

## Update

Long update:  $O(\log n)$

Short update:

- Delete all weighted intervals
- Recompute weighted intervals
- Insert all weighted intervals

$O((n/r) \cdot \log n)$

# Update Time vs Approx. Ratio



Exact data structure: balancing long and short rectangles

# Update Time vs Approx. Ratio



Exact data structure: balancing long and short rectangles

💡 Speed-up for approx.: maintain long and short rectangles separately

$D_1$ : exact data structure

$I$ : interval data structure

$\mathcal{L}$ : set of long rect.

$\mathcal{S}$ : set of short rect.



# Update Time vs Approx. Ratio

Exact data structure: balancing long and short rectangles

Speed-up for approx.: maintain long and short rectangles separately

## Construction

$D_2$  : Build  $D_1$  on  $\mathcal{S}$ ,  $I$  on  $\mathcal{L}$

$\mathbf{D}_1$ : exact data structure

$\mathbf{I}$ : interval data structure

$\mathcal{L}$ : set of long rect.

$\mathcal{S}$ : set of short rect.



# Update Time vs Approx. Ratio



Exact data structure: balancing long and short rectangles

Construction

$\mathbf{D}_2$  : Build  $\mathbf{D}_1$  on  $\mathcal{S}$ ,  $\mathbf{I}$  on  $\mathcal{L}$

Approx. ratio

$$\max \{\text{dep}(\mathcal{S}), \text{dep}(\mathcal{L})\} \geq \frac{1}{2} \text{dep}(\mathcal{O})$$

$\mathbf{D}_1$ : exact data structure

$\mathbf{I}$ : interval data structure

$\mathcal{L}$ : set of long rect.

$\mathcal{S}$ : set of short rect.



# Update Time vs Approx. Ratio

Exact data structure: balancing long and short rectangles

Speed-up for approx.: maintain long and short rectangles separately

Construction

$\mathbf{D}_2$  : Build  $\mathbf{D}_1$  on  $\mathcal{S}$ ,  $\mathbf{I}$  on  $\mathcal{L}$

Approx. ratio

$$\max \{\text{dep}(\mathcal{S}), \text{dep}(\mathcal{L})\} \geq \frac{1}{2} \text{dep}(\mathcal{O})$$

Update time

$$\tilde{O} \left( 2 \cdot (n/r)^{\frac{1}{2}} + r \right) = \tilde{O} \left( n^{\frac{1}{3}} \right)$$

$r = n^{1/3}$

$\mathbf{D}_1$ : exact data structure

$\mathbf{I}$ : interval data structure

$\mathcal{L}$ : set of long rect.

$\mathcal{S}$ : set of short rect.



# Update Time vs Approx. Ratio

Speed-up for approx.: maintain long and short rectangles separately

Construction

$\mathbf{D}_2$  : Build  $\mathbf{D}_1$  on  $\mathcal{S}$ ,  $\mathbf{I}$  on  $\mathcal{L}$

Approx. ratio

$$\max \{\text{dep}(\mathcal{S}), \text{dep}(\mathcal{L})\} \geq \frac{1}{2} \text{dep}(\mathcal{O})$$

Update time

$$\tilde{O} \left( 2 \cdot (n/r)^{\frac{1}{2}} + r \right) = \tilde{O} \left( n^{\frac{1}{3}} \right)$$

$r = n^{1/3}$

$\mathbf{D}_3$  : Build  $\mathbf{D}_2$  on  $\mathcal{S}$ ,  $\mathbf{I}$  on  $\mathcal{L}$

$D_1$ : exact data structure

$I$ : interval data structure

$\mathcal{L}$ : set of long rect.

$\mathcal{S}$ : set of short rect.



# Update Time vs Approx. Ratio



Exact data structure: balancing long and short rectangles

Speed-up for approx.: maintain long and short rectangles separately

Construction

$D_2$  : Build  $D_1$  on  $\mathcal{S}$ ,  $I$  on  $\mathcal{L}$

Approx. ratio

$$\max \left\{ \text{dep}(\mathcal{S}), \text{dep}(\mathcal{L}) \right\} \geq \frac{1}{2} \text{dep}(\mathcal{O})$$

Update time

$$\tilde{O} \left( 2 \cdot (n/r)^{\frac{1}{2}} + r \right) = \tilde{O} \left( n^{\frac{1}{3}} \right)$$

$r = n^{1/3}$

$D_3$  : Build  $D_2$  on  $\mathcal{S}$ ,  $I$  on  $\mathcal{L}$

$$\max \left\{ \frac{1}{2} \text{dep}(\mathcal{S}), \text{dep}(\mathcal{L}) \right\} \geq \frac{1}{3} \text{dep}(\mathcal{O})$$

$\mathbf{D}_1$ : exact data structure

$\mathbf{I}$ : interval data structure

$\mathcal{L}$ : set of long rect.

$\mathcal{S}$ : set of short rect.



# Update Time vs Approx. Ratio



Exact data structure: balancing long and short rectangles

Construction

$\mathbf{D}_2$  : Build  $\mathbf{D}_1$  on  $\mathcal{S}$ ,  $\mathbf{I}$  on  $\mathcal{L}$

$$\text{Approx. ratio} \quad \max \left\{ \text{dep}(\mathcal{S}), \text{dep}(\mathcal{L}) \right\} \geq \frac{1}{2} \text{dep}(\mathcal{O})$$

Update time

$$\tilde{\mathcal{O}} \left( 2 \cdot (n/r)^{\frac{1}{2}} + r \right) = \tilde{\mathcal{O}} \left( n^{\frac{1}{3}} \right)$$

$r = n^{1/3}$

$\mathbf{D}_3$  : Build  $\mathbf{D}_2$  on  $\mathcal{S}$ ,  $\mathbf{I}$  on  $\mathcal{L}$

$$\text{Approx. ratio} \quad \max \left\{ \frac{1}{2} \text{dep}(\mathcal{S}), \text{dep}(\mathcal{L}) \right\} \geq \frac{1}{3} \text{dep}(\mathcal{O})$$

$$\tilde{\mathcal{O}} \left( 2 \cdot (n/r)^{\frac{1}{3}} + r \right) = \tilde{\mathcal{O}} \left( n^{\frac{1}{4}} \right)$$

$r = n^{1/4}$

$D_1$ : exact data structure

$I$ : interval data structure

$\mathcal{L}$ : set of long rect.

$\mathcal{S}$ : set of short rect.



# Update Time vs Approx. Ratio



Exact data structure: balancing long and short rectangles

Speed-up for approx.: maintain long and short rectangles separately

Construction

$D_2$  : Build  $D_1$  on  $\mathcal{S}$ ,  $I$  on  $\mathcal{L}$

Approx. ratio

$$\max \{ \text{dep}(\mathcal{S}), \text{dep}(\mathcal{L}) \} \geq \frac{1}{2} \text{dep}(\mathcal{O})$$

Update time

$$\tilde{O} \left( 2 \cdot (n/r)^{\frac{1}{2}} + r \right) = \tilde{O} \left( n^{\frac{1}{3}} \right)$$

$D_3$  : Build  $D_2$  on  $\mathcal{S}$ ,  $I$  on  $\mathcal{L}$

$$\max \left\{ \frac{1}{2} \text{dep}(\mathcal{S}), \text{dep}(\mathcal{L}) \right\} \geq \frac{1}{3} \text{dep}(\mathcal{O})$$

$$\tilde{O} \left( 2 \cdot (n/r)^{\frac{1}{3}} + r \right) = \tilde{O} \left( n^{\frac{1}{4}} \right)$$

• • •

• • •

• • •

$D_1$ : exact data structure

$I$ : interval data structure

$\mathcal{L}$ : set of long rect.

$\mathcal{S}$ : set of short rect.



# Update Time vs Approx. Ratio



Speed-up for approx.: maintain long and short rectangles separately

Construction

$D_2$  : Build  $D_1$  on  $\mathcal{S}$ ,  $I$  on  $\mathcal{L}$

Approx. ratio

$$\max \left\{ \text{dep}(\mathcal{S}), \text{dep}(\mathcal{L}) \right\} \geq \frac{1}{2} \text{dep}(\mathcal{O})$$

Update time

$$\tilde{O} \left( 2 \cdot \frac{(n/r)^{\frac{1}{2}} + r}{r = n^{1/3}} \right) = \tilde{O} \left( n^{\frac{1}{3}} \right)$$

$D_3$  : Build  $D_2$  on  $\mathcal{S}$ ,  $I$  on  $\mathcal{L}$

$$\max \left\{ \frac{1}{2} \text{dep}(\mathcal{S}), \text{dep}(\mathcal{L}) \right\} \geq \frac{1}{3} \text{dep}(\mathcal{O})$$

$$\tilde{O} \left( 2 \cdot \frac{(n/r)^{\frac{1}{3}} + r}{r = n^{1/4}} \right) = \tilde{O} \left( n^{\frac{1}{4}} \right)$$

• • •

• • •

• • •

$D_k$  : Build  $D_{k-1}$  on  $\mathcal{S}$ ,  $I$  on  $\mathcal{L}$

$$\max \left\{ \frac{1}{k-1} \text{dep}(\mathcal{S}), \text{dep}(\mathcal{L}) \right\} \geq \frac{1}{k} \text{dep}(\mathcal{O})$$

$$\tilde{O} \left( 2 \cdot \frac{(n/r)^{\frac{1}{k}} + r}{r = n^{1/(k+1)}} \right) = \tilde{O} \left( n^{\frac{1}{k+1}} \right)$$

$D_1$ : exact data structure

$I$ : interval data structure

$\mathcal{L}$ : set of long rect.

$\mathcal{S}$ : set of short rect.



# Update Time vs Approx. Ratio



Exact data structure: balancing long and short rectangles

Speed-up for approx.: maintain long and short rectangles separately

Construction

$D_2$  : Build  $D_1$  on  $\mathcal{S}$ ,  $I$  on  $\mathcal{L}$

Approx. ratio

$$\max \{ \text{dep}(\mathcal{S}), \text{dep}(\mathcal{L}) \} \geq \frac{1}{2} \text{dep}(\mathcal{O})$$

Update time

$$\tilde{O} \left( 2 \cdot (n/r)^{\frac{1}{2}} + r \right) = \tilde{O} \left( n^{\frac{1}{3}} \right)$$

$D_3$  : Build  $D_2$  on  $\mathcal{S}$ ,  $I$  on  $\mathcal{L}$

$$\max \left\{ \frac{1}{2} \text{dep}(\mathcal{S}), \text{dep}(\mathcal{L}) \right\} \geq \frac{1}{3} \text{dep}(\mathcal{O})$$

$$\tilde{O} \left( 2 \cdot (n/r)^{\frac{1}{3}} + r \right) = \tilde{O} \left( n^{\frac{1}{4}} \right)$$

• • •

• • •

• • •

$D_k$  : Build  $D_{k-1}$  on  $\mathcal{S}$ ,  $I$  on  $\mathcal{L}$

$$\max \left\{ \frac{1}{k-1} \text{dep}(\mathcal{S}), \text{dep}(\mathcal{L}) \right\} \geq \frac{1}{k} \text{dep}(\mathcal{O})$$

$$\tilde{O} \left( 2 \cdot (n/r)^{\frac{1}{k}} + r \right) = \tilde{O} \left( n^{\frac{1}{k+1}} \right)$$

## Summary

$\forall k \in \mathbb{Z}^+, \exists \frac{1}{k}$ -approx. data structure for dynamic MaxDep with  $\tilde{O} \left( n^{\frac{1}{k+1}} \right)$  update time

# **Approximate via Discretization**

# Approximate via Discretization

## DisMaxDepth

**Input:** a set of geometric objects  $\mathcal{O}$

a set of points  $\mathcal{P}$

**Output:** a point  $p^* \in \mathcal{P}$  s.t.

$$\text{dep}(p^*, \mathcal{O}) = \text{dep}(\mathcal{P}, \mathcal{O}) = \max_{p \in \mathcal{P}} \text{dep}(p, \mathcal{O})$$

and the value of  $\text{dep}(p^*, \mathcal{O})$

# Approximate via Discretization

## DisMaxDepth

**Input:** a set of geometric objects  $\mathcal{O}$   
a set of points  $\mathcal{P}$

**Output:** a point  $p^* \in \mathcal{P}$  s.t.

$$\text{dep}(p^*, \mathcal{O}) = \text{dep}(\mathcal{P}, \mathcal{O}) = \max_{p \in \mathcal{P}} \text{dep}(p, \mathcal{O})$$

and the value of  $\text{dep}(p^*, \mathcal{O})$

## Companion points

Insert/delete some “well-chosen” points  $\Gamma(O)$   
while insert/delete geometric object  $O$

# Approximate via Discretization

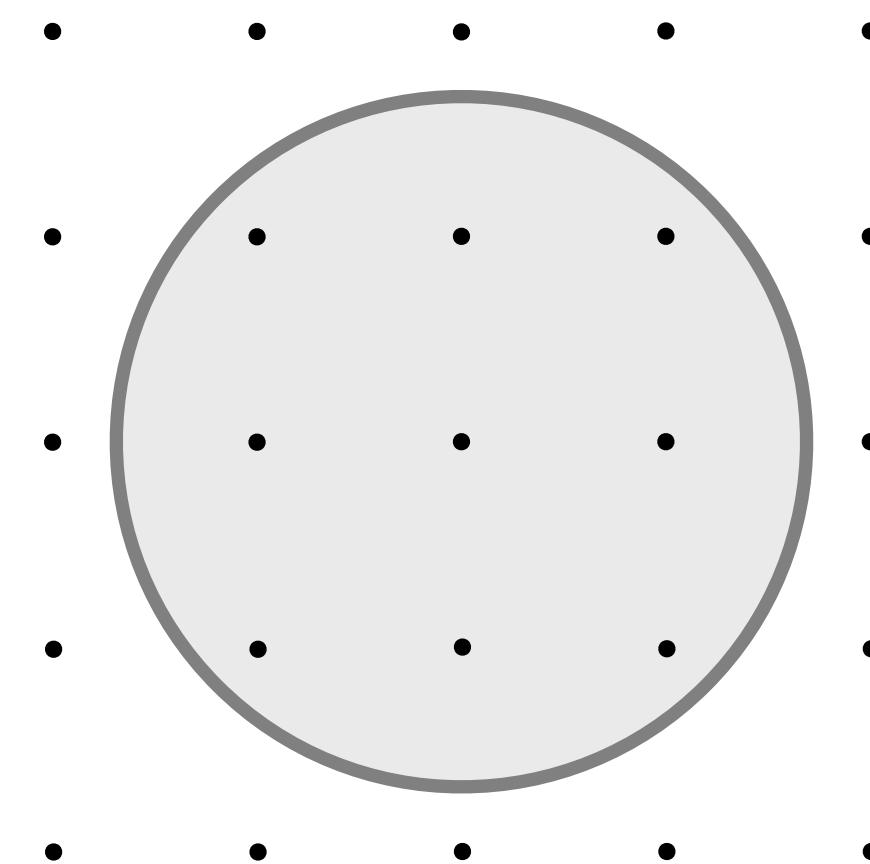
## DisMaxDepth

**Input:** a set of geometric objects  $\mathcal{O}$   
a set of points  $\mathcal{P}$

**Output:** a point  $p^* \in \mathcal{P}$  s.t.

$$\text{dep}(p^*, \mathcal{O}) = \text{dep}(\mathcal{P}, \mathcal{O}) = \max_{p \in \mathcal{P}} \text{dep}(p, \mathcal{O})$$

and the value of  $\text{dep}(p^*, \mathcal{O})$



## Companion points

Insert/delete some “well-chosen” points  $\Gamma(O)$   
while insert/delete geometric object  $O$

# Approximate via Discretization

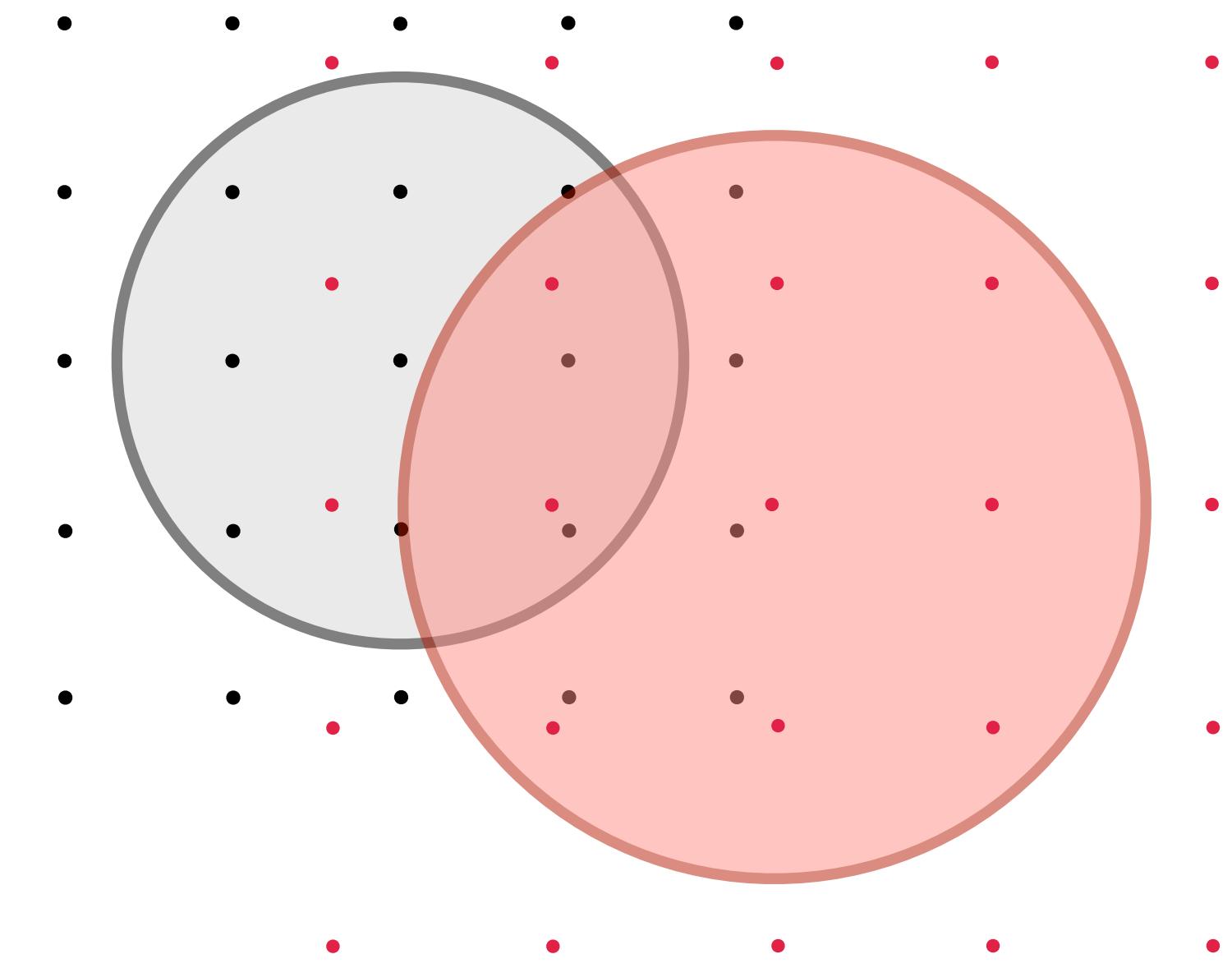
## DisMaxDepth

**Input:** a set of geometric objects  $\mathcal{O}$   
a set of points  $\mathcal{P}$

**Output:** a point  $p^* \in \mathcal{P}$  s.t.

$$\text{dep}(p^*, \mathcal{O}) = \text{dep}(\mathcal{P}, \mathcal{O}) = \max_{p \in \mathcal{P}} \text{dep}(p, \mathcal{O})$$

and the value of  $\text{dep}(p^*, \mathcal{O})$



## Companion points

Insert/delete some “well-chosen” points  $\Gamma(O)$

while insert/delete geometric object  $O$

# Approximate via Discretization

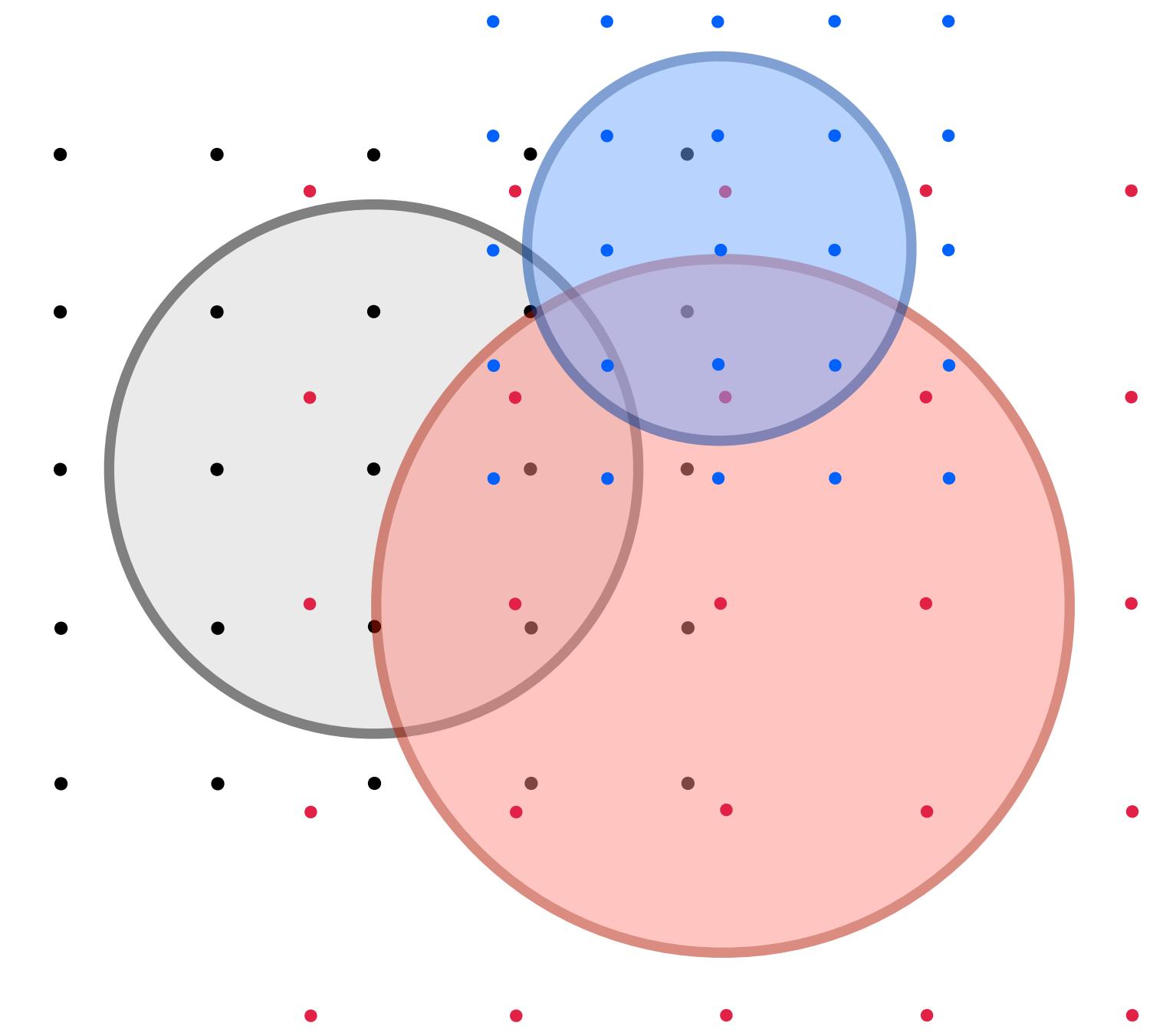
## DisMaxDepth

**Input:** a set of geometric objects  $\mathcal{O}$   
a set of points  $\mathcal{P}$

**Output:** a point  $p^* \in \mathcal{P}$  s.t.

$$\text{dep}(p^*, \mathcal{O}) = \text{dep}(\mathcal{P}, \mathcal{O}) = \max_{p \in \mathcal{P}} \text{dep}(p, \mathcal{O})$$

and the value of  $\text{dep}(p^*, \mathcal{O})$



## Companion points

Insert/delete some “well-chosen” points  $\Gamma(O)$

while insert/delete geometric object  $O$

# Approximate via Discretization

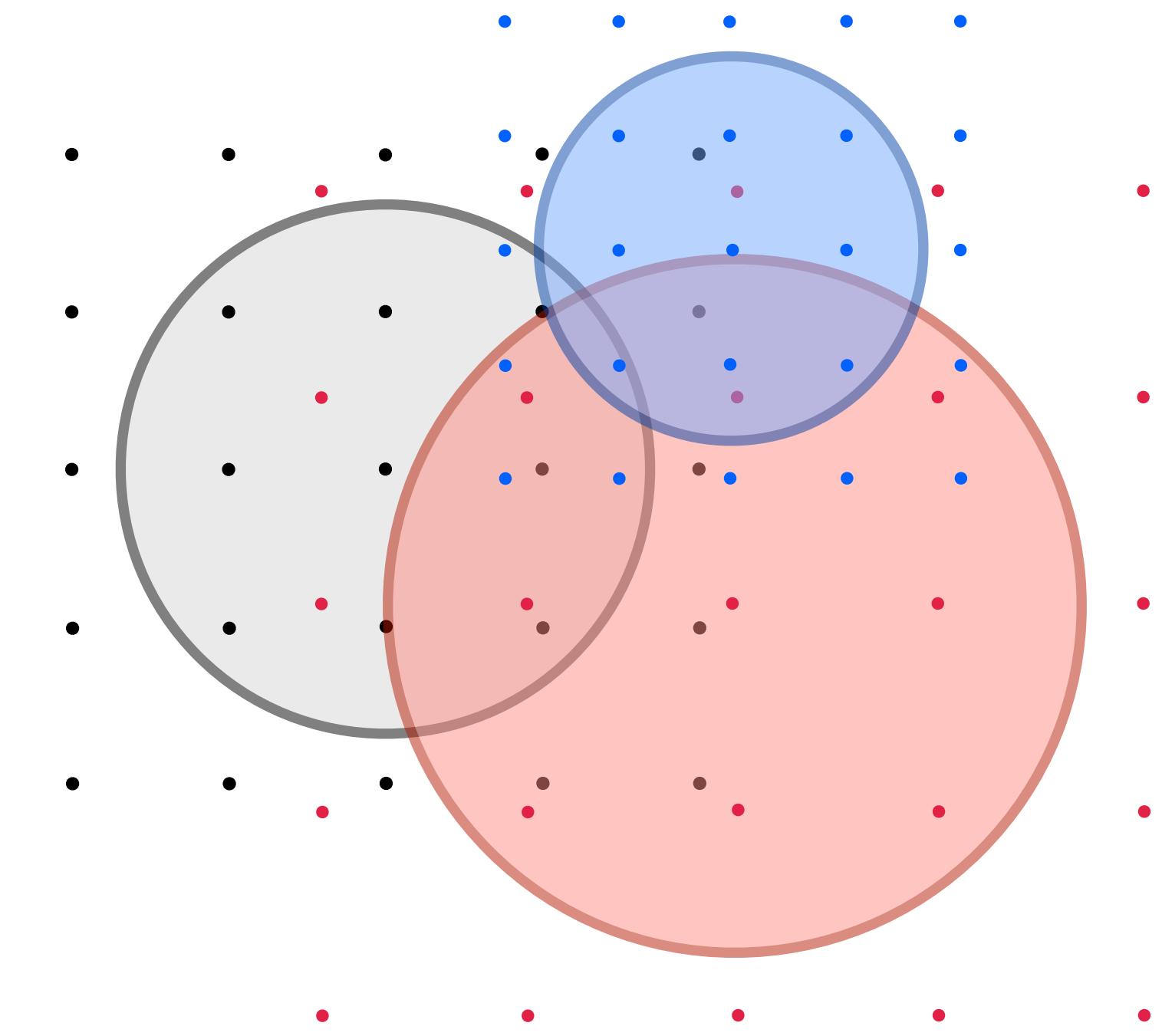
## DisMaxDepth

**Input:** a set of geometric objects  $\mathcal{O}$   
a set of points  $\mathcal{P}$

**Output:** a point  $p^* \in \mathcal{P}$  s.t.

$$\text{dep}(p^*, \mathcal{O}) = \text{dep}(\mathcal{P}, \mathcal{O}) = \max_{p \in \mathcal{P}} \text{dep}(p, \mathcal{O})$$

and the value of  $\text{dep}(p^*, \mathcal{O})$



## Companion points

Insert/delete some “well-chosen” points  $\Gamma(O)$   
while insert/delete geometric object  $O$

## “Well-chosen”

Hopefully,  $\text{dep}(\Gamma, \mathcal{O}) \geq \frac{1}{k} \cdot \text{dep}(\mathcal{O})$

# Approximation for Disks and Fat Objects

# Approximation for Disks and Fat Objects

## Approx. for disks

$$\forall \varepsilon > 0, \exists \Gamma, \text{s.t. } \text{dep}(\Gamma, \mathcal{O}) \geq \left( \frac{1}{2} - \varepsilon \right) \text{dep}(\mathcal{O})$$

# Approximation for Disks and Fat Objects

## Approx. for disks

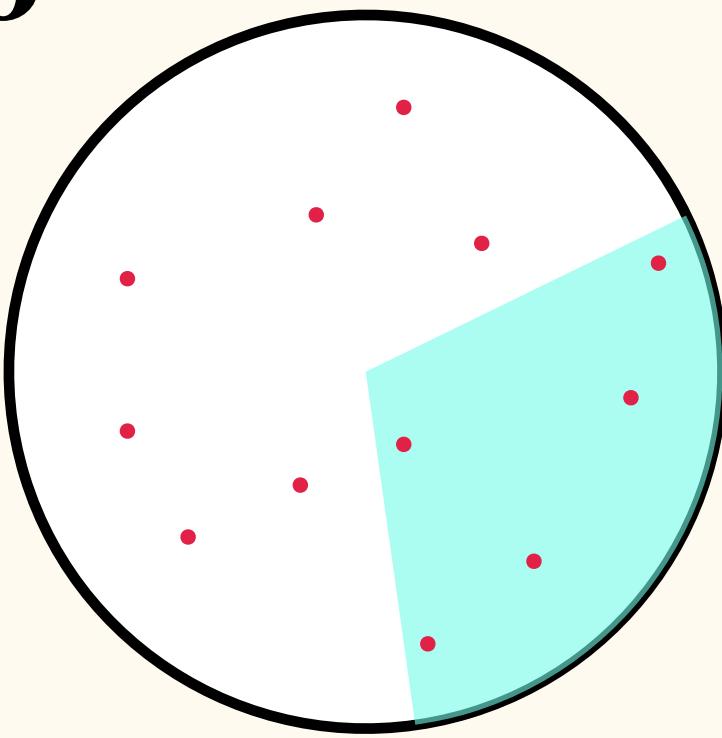
$$\forall \varepsilon > 0, \exists \Gamma, \text{s.t. } \text{dep}(\Gamma, \mathcal{O}) \geq \left( \frac{1}{2} - \varepsilon \right) \text{dep}(\mathcal{O})$$

## The source of ratio

Given a disk  $D$  and a set  $S$  of points in  $D$ ,

$\exists$  circular sector  $X$  with angle  $(1 - 2\varepsilon)\pi$ ,

$$\text{s.t. } |S \cap X| \geq \left( \frac{1}{2} - \varepsilon \right) |S|$$



# Approximation for Disks and Fat Objects

## Approx. for disks

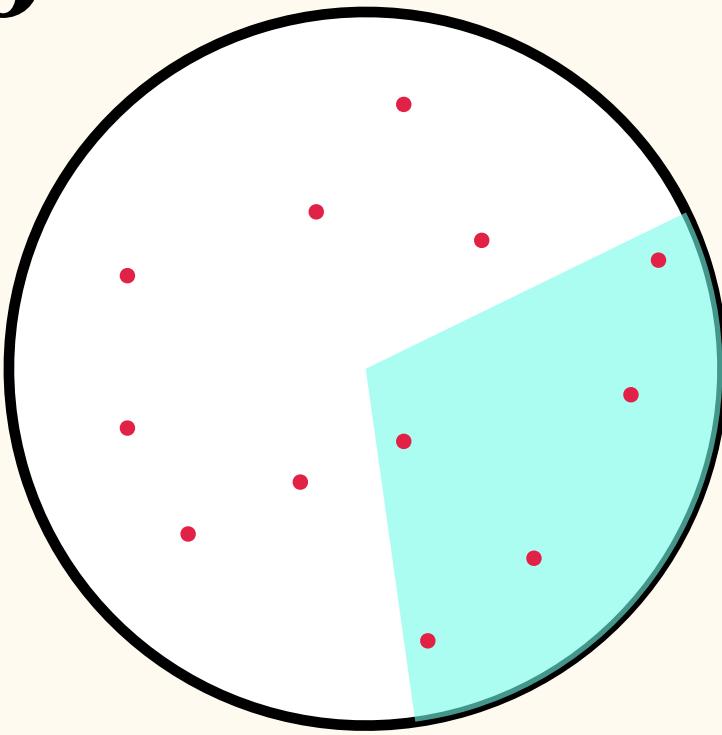
$$\forall \varepsilon > 0, \exists \Gamma, \text{s.t. } \text{dep}(\Gamma, \mathcal{O}) \geq \left( \frac{1}{2} - \varepsilon \right) \text{dep}(\mathcal{O})$$

## The source of ratio

Given a disk  $D$  and a set  $S$  of points in  $D$ ,

$\exists$  circular sector  $X$  with angle  $(1 - 2\varepsilon)\pi$ ,

$$\text{s.t. } |S \cap X| \geq \left( \frac{1}{2} - \varepsilon \right) |S|$$



## Data structure for disks

$\tilde{O}(n^{2/3})$  update time

via dynamic partition tree [Matoušek 1992]

# Approximation for Disks and Fat Objects

## Approx. for disks

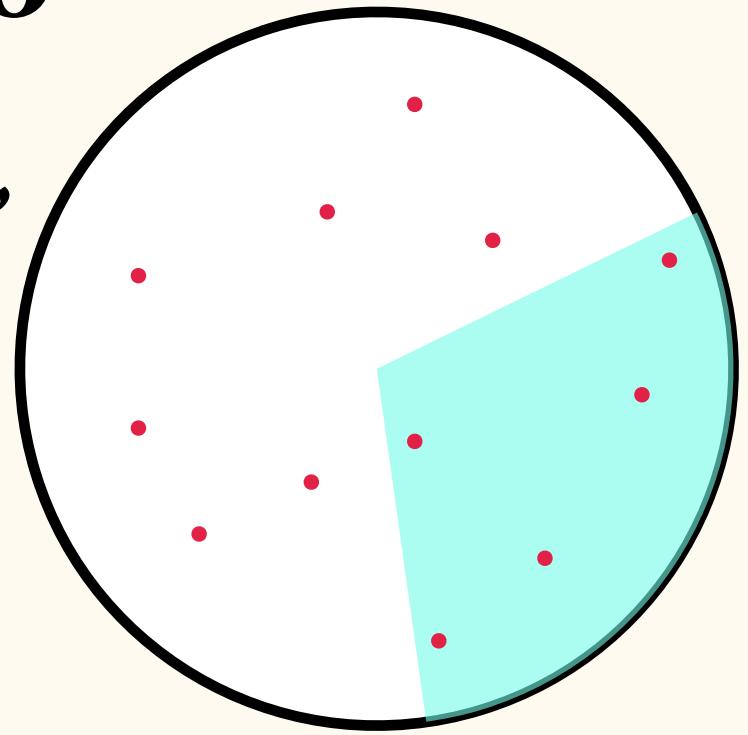
$$\forall \varepsilon > 0, \exists \Gamma, \text{s.t. } \text{dep}(\Gamma, \mathcal{O}) \geq \left( \frac{1}{2} - \varepsilon \right) \text{dep}(\mathcal{O})$$

## The source of ratio

Given a disk  $D$  and a set  $S$  of points in  $D$ ,

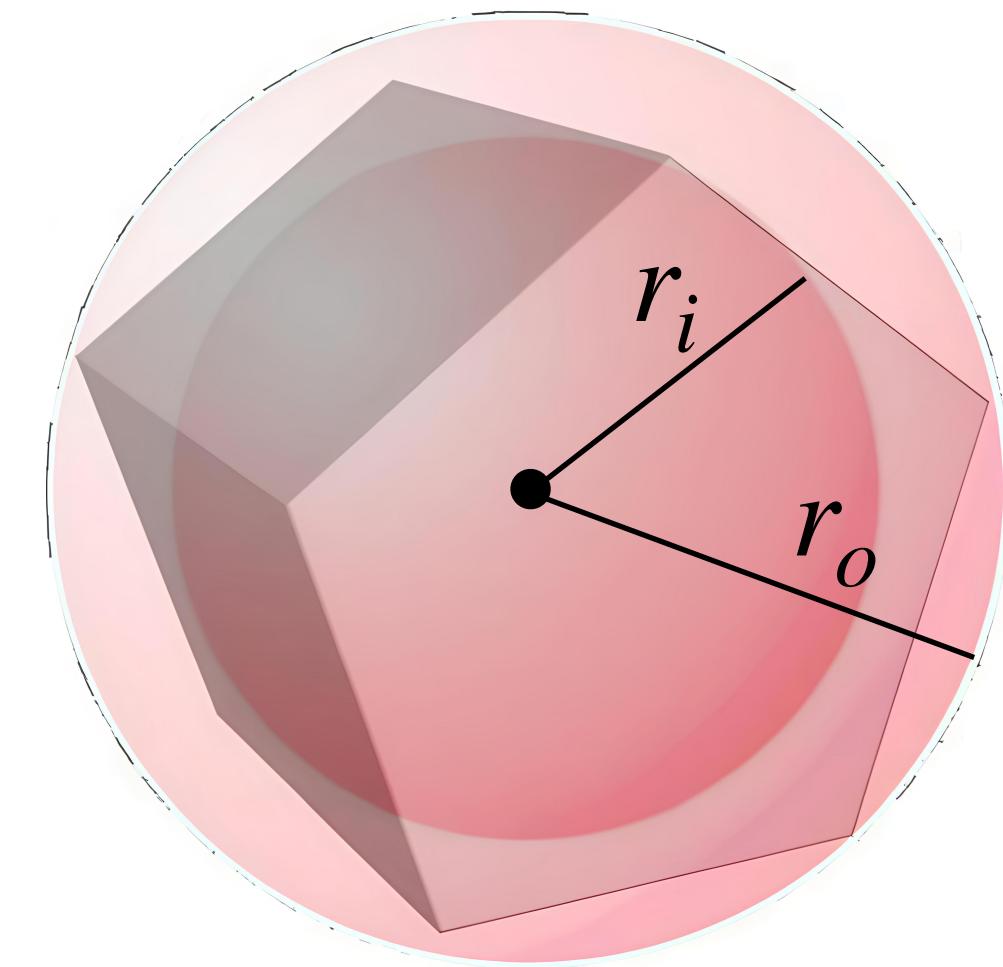
$\exists$  circular sector  $X$  with angle  $(1 - 2\varepsilon)\pi$ ,

$$\text{s.t. } |S \cap X| \geq \left( \frac{1}{2} - \varepsilon \right) |S|$$



$\alpha$ -fat object:

$$\frac{r_i}{r_o} \leq \alpha$$



## Data structure for disks

$\tilde{O}(n^{2/3})$  update time

via dynamic partition tree [Matoušek 1992]

# Approximation for Disks and Fat Objects

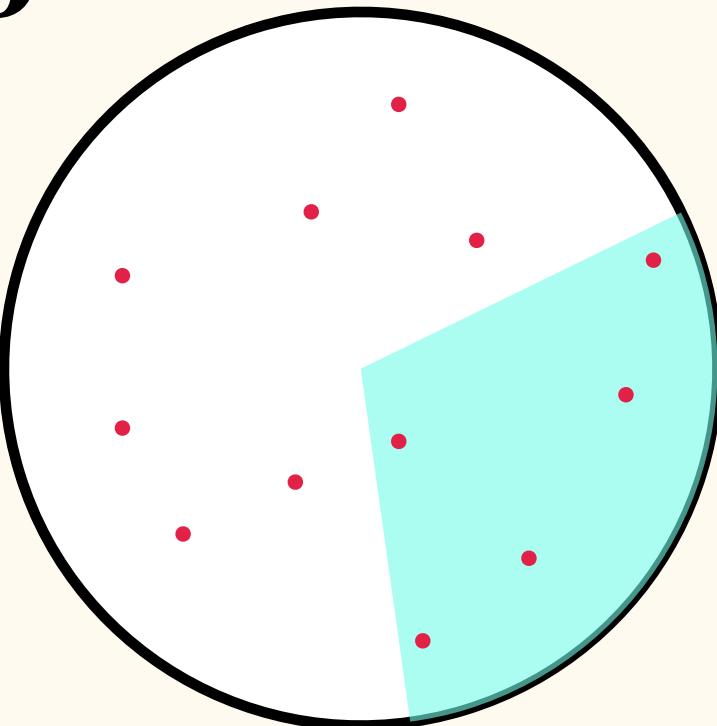
## Approx. for disks

$$\forall \varepsilon > 0, \exists \Gamma, \text{s.t. } \text{dep}(\Gamma, \mathcal{O}) \geq \left( \frac{1}{2} - \varepsilon \right) \text{dep}(\mathcal{O})$$

## The source of ratio

Given a disk  $D$  and a set  $S$  of points in  $D$ ,  
 $\exists$  circular sector  $X$  with angle  $(1 - 2\varepsilon)\pi$ ,

$$\text{s.t. } |S \cap X| \geq \left( \frac{1}{2} - \varepsilon \right) |S|$$



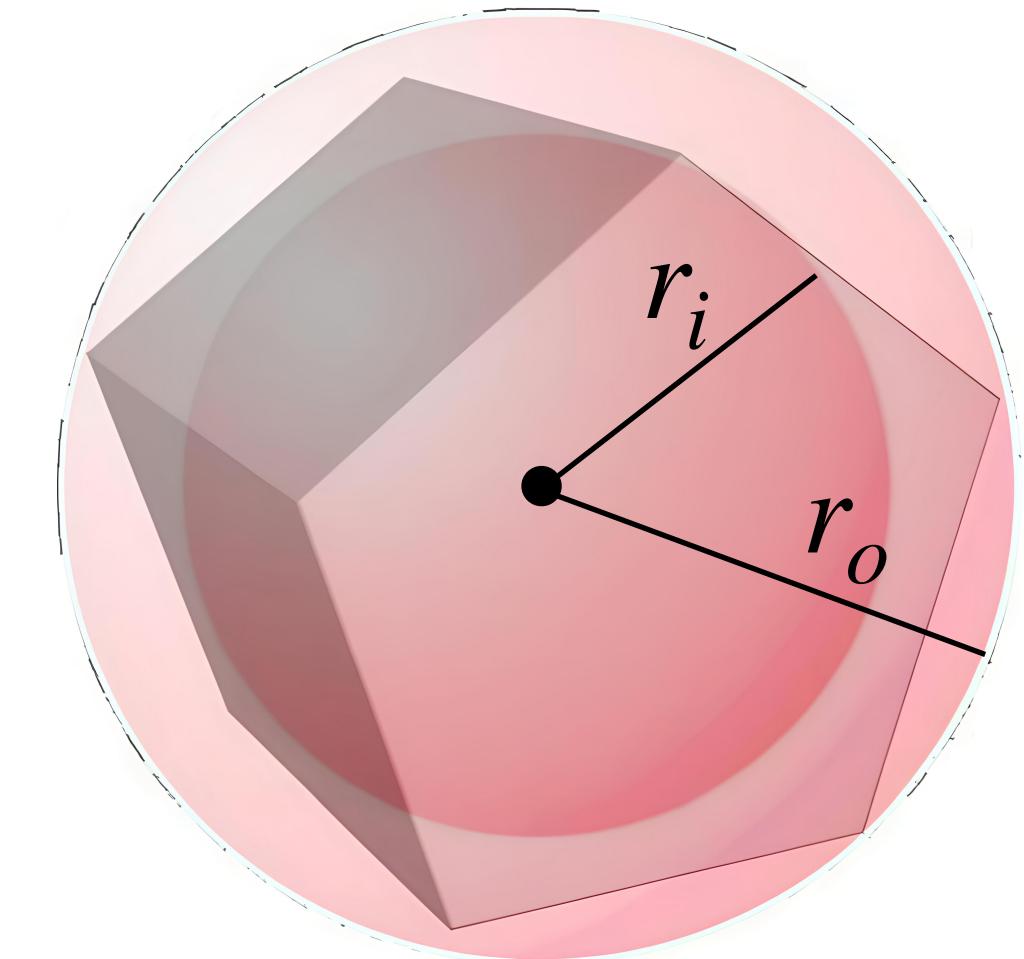
## Data structure for disks

$\tilde{O}(n^{2/3})$  update time

via dynamic partition tree [Matoušek 1992]

$\alpha$ -fat object:

$$\frac{r_i}{r_o} \leq \alpha$$



## Strategy for fat objects (in $\mathbb{R}^d$ )

Reduce to “balls cover ball” problem:

If  $k$  unit balls cover a ball with radius  $2\alpha$ ,  
then  $\text{dep}(\Gamma, \mathcal{O}) \geq \frac{1}{k} \text{dep}(\mathcal{O})$

# Approximation for Disks and Fat Objects

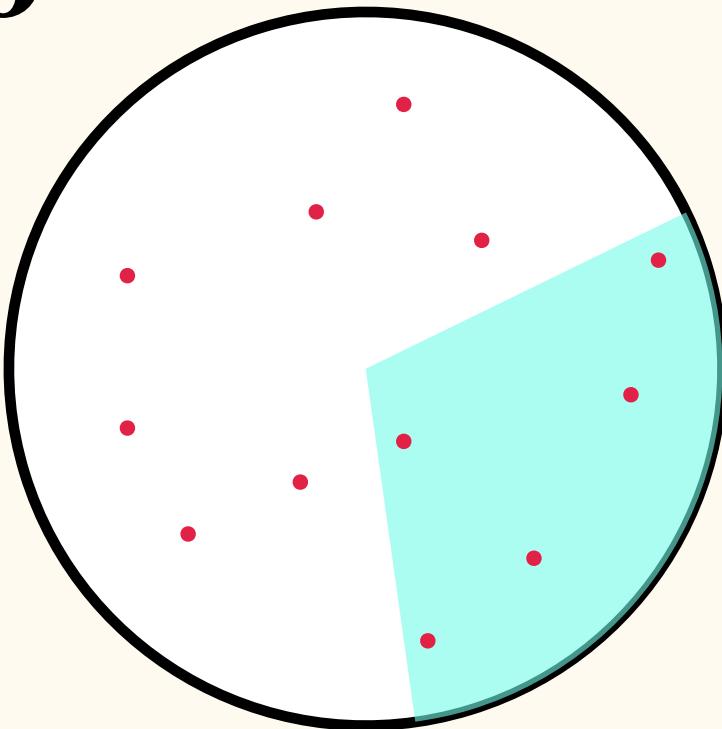
## Approx. for disks

$$\forall \varepsilon > 0, \exists \Gamma, \text{s.t. } \text{dep}(\Gamma, \mathcal{O}) \geq \left( \frac{1}{2} - \varepsilon \right) \text{dep}(\mathcal{O})$$

## The source of ratio

Given a disk  $D$  and a set  $S$  of points in  $D$ ,  
 $\exists$  circular sector  $X$  with angle  $(1 - 2\varepsilon)\pi$ ,

$$\text{s.t. } |S \cap X| \geq \left( \frac{1}{2} - \varepsilon \right) |S|$$

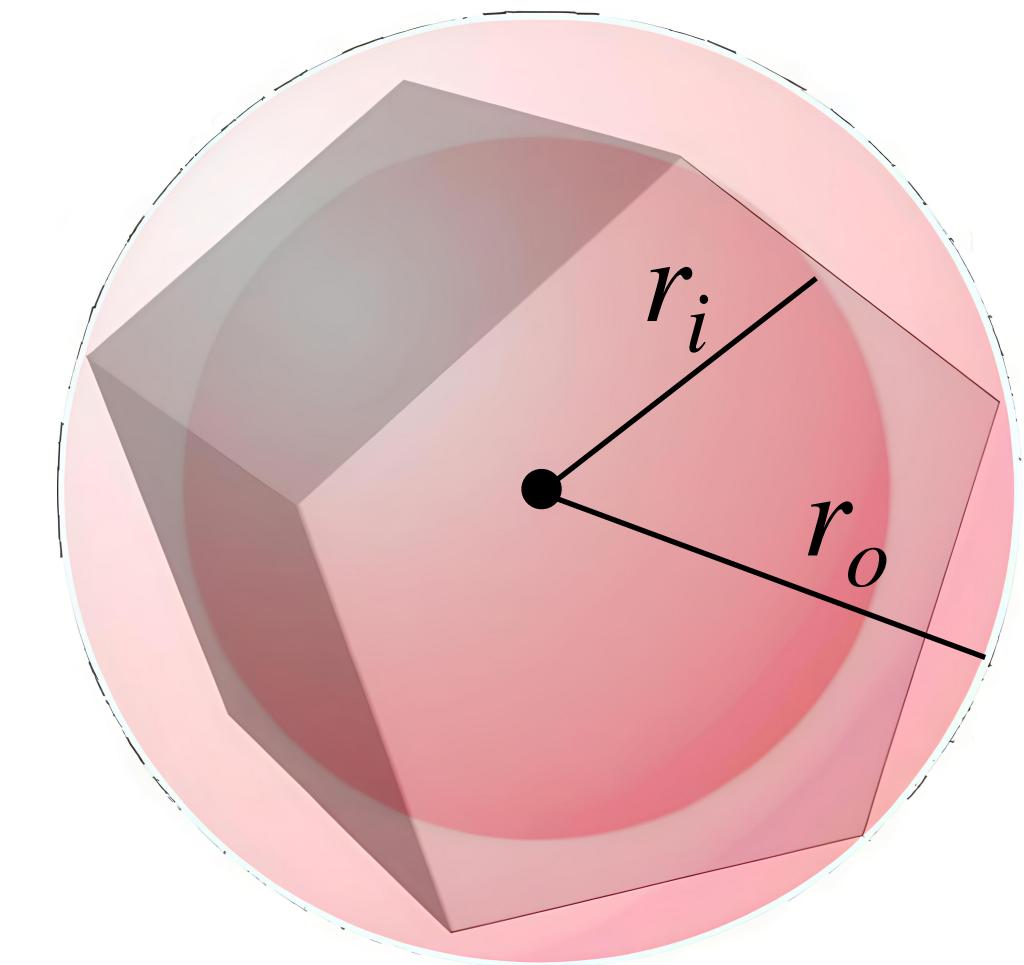


## Data structure for disks

$\tilde{O}(n^{2/3})$  update time  
via dynamic partition tree [Matoušek 1992]

$\alpha$ -fat object:

$$\frac{r_i}{r_o} \leq \alpha$$



## Strategy for fat objects (in $\mathbb{R}^d$ )

Reduce to “balls cover ball” problem:

If  $k$  unit balls cover a ball with radius  $2\alpha$ ,

$$\text{then } \text{dep}(\Gamma, \mathcal{O}) \geq \frac{1}{k} \text{dep}(\mathcal{O})$$

$$\text{Trivial bound: } k \leq \left( 2 \lceil \sqrt{2\alpha} \rceil \right)^d$$

# Approximation for Disks and Fat Objects

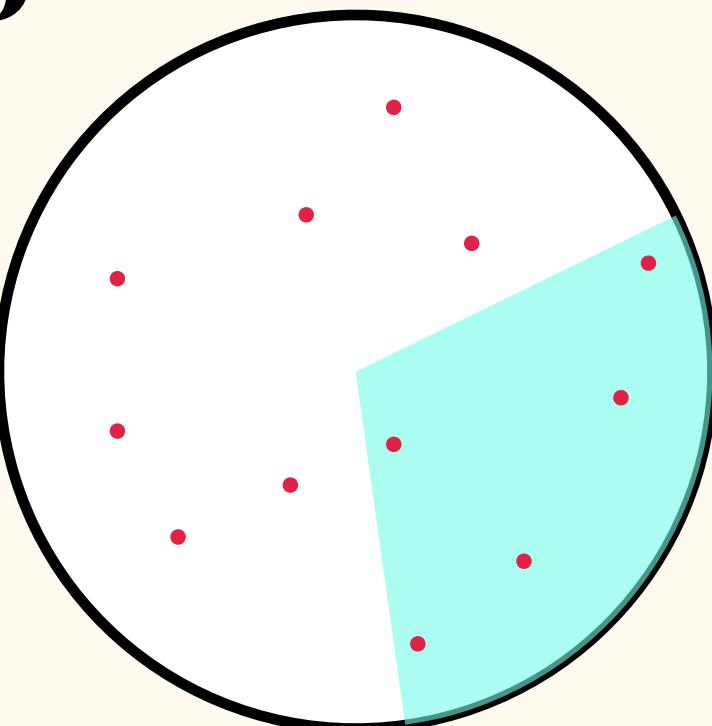
## Approx. for disks

$$\forall \varepsilon > 0, \exists \Gamma, \text{s.t. } \text{dep}(\Gamma, \mathcal{O}) \geq \left( \frac{1}{2} - \varepsilon \right) \text{dep}(\mathcal{O})$$

## The source of ratio

Given a disk  $D$  and a set  $S$  of points in  $D$ ,  
 $\exists$  circular sector  $X$  with angle  $(1 - 2\varepsilon)\pi$ ,

$$\text{s.t. } |S \cap X| \geq \left( \frac{1}{2} - \varepsilon \right) |S|$$



## Data structure for disks

$\tilde{O}(n^{2/3})$  update time  
via dynamic partition tree [Matoušek 1992]

$\alpha$ -fat object:

$$\frac{r_i}{r_o} \leq \alpha$$

No data structure yet 😱

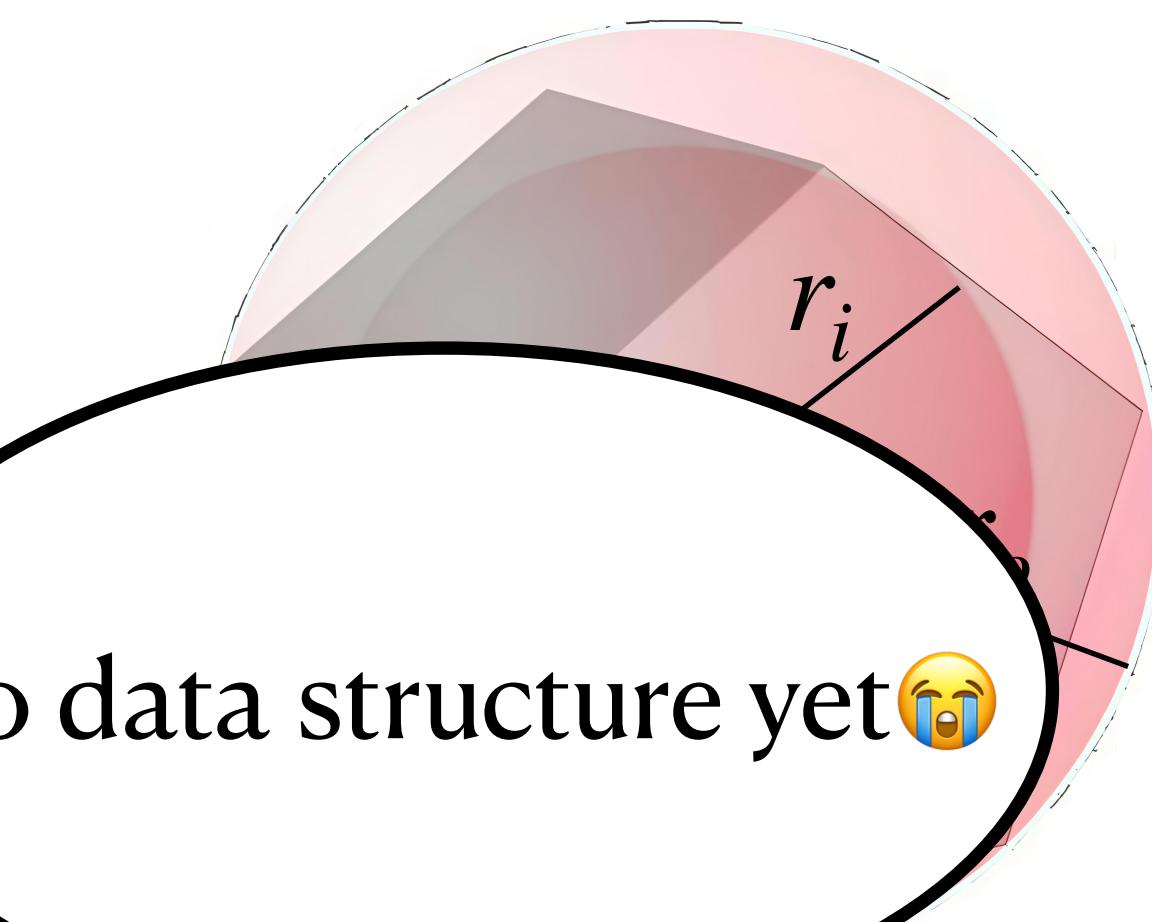
## Strategy for fat objects (in $\mathbb{R}^d$ )

Reduce to “balls cover ball” problem:

If  $k$  unit balls cover a ball with radius  $2\alpha$ ,

$$\text{then } \text{dep}(\Gamma, \mathcal{O}) \geq \frac{1}{k} \text{dep}(\mathcal{O})$$

Trivial bound:  $k \leq \left( 2 \lceil \sqrt{2\alpha} \rceil \right)^d$



# Thank you!

To appear in SoCG 2025

## Open Problems

- Data structure for  $d$ -dim. boxes with  $\tilde{O}(n^{(d-1)/2})$  update time
- Improve update time for special objects, e.g., squares
- Data structure for fat objects in discrete setting
- Data structure for other objects, e.g., halfspaces, quadrants