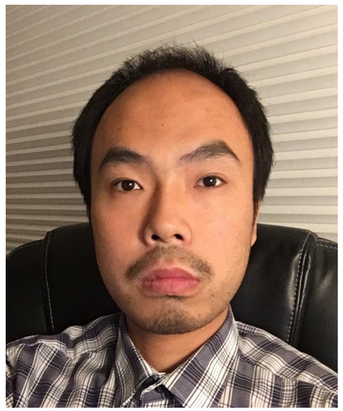
Dynamic Maximum Depth of Geometric Objects

Xiongxin Yang

NYU Shanghai ⟶ HKU

Based on joint work with

Subhash Suri
UCSB



Jie Xue
NYU Shanghai

Jiumu Zhu
NYU Shanghai

The 41st International Symposium on Computational Geometry (SoCG 2025)
Kanazawa, Japan
25 June, 2025

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.

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

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

Depth Problem

#objects that contain a point

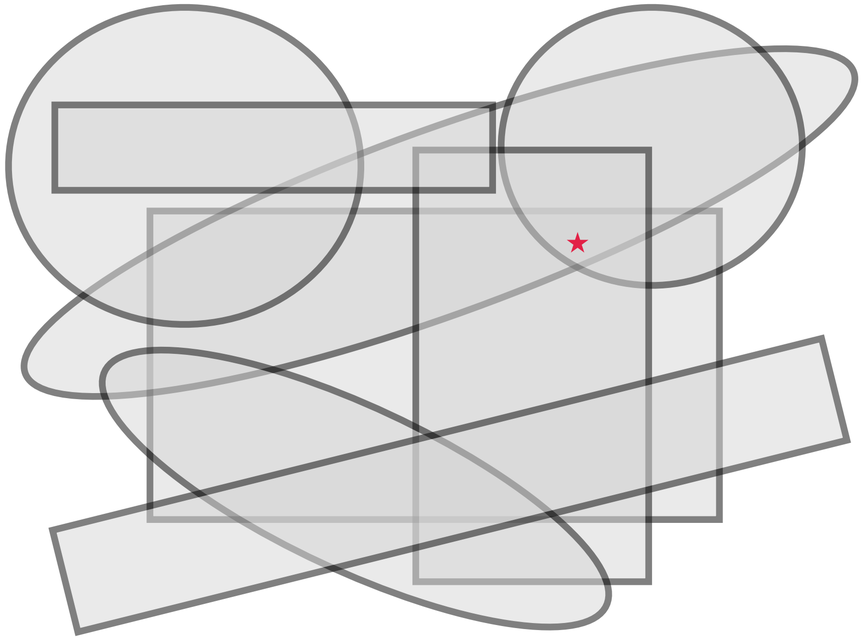
MaxDepth

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

Output: a point p^* s.t.

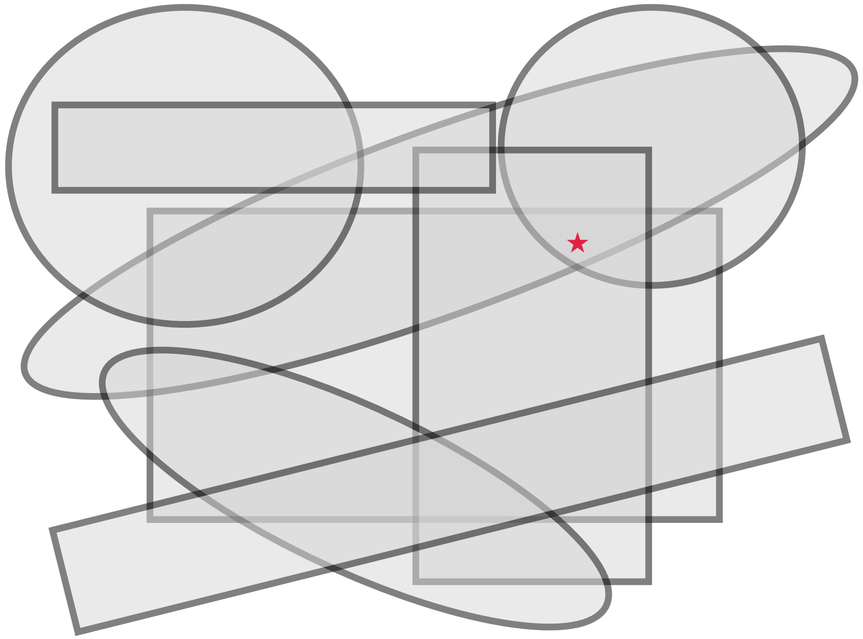
$$\mathrm{dep}(p^*, \mathcal{O}) = \mathrm{dep}(\mathcal{O}) = \max_{p \in \mathbb{R}^d} \mathrm{dep}(p, \mathcal{O})$$

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



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.

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

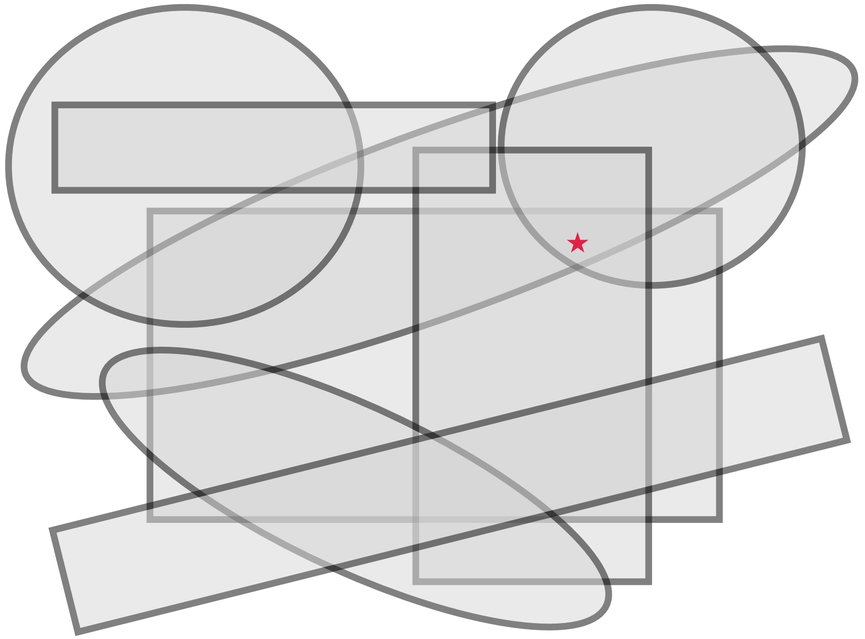
and the value of $\mathrm{dep}(p^*, \mathcal{O})$

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.

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

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

Canonical shape

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

Dynamic version

Insert or delete an object is allowed
Goal: 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, bichomatic 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, bichomatic 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, bichomatic closest pair, union of intervals, convex hull, width, geometric connectivity …

Dynamic data stru…etric optimization problem

On geometric inters…

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

minimum set cover […2020; Chan and He 2021; Chan, He, Suri and Xue 2022]

minimum hitting set […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, bichomatic closest pair, union of intervals, convex hull, width, geometric connectivity …

Dynamic data stru…etric optimization problem

On geometric inters…

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

minimum set cover […2020; Chan and He 2021; Chan, He, Suri and Xue 2022]

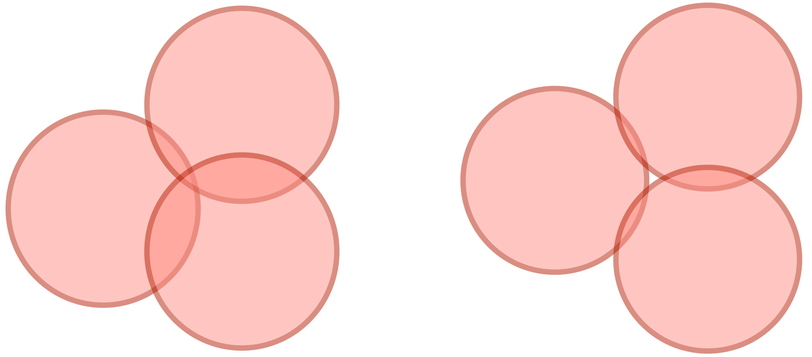
minimum hitting set […2005; Agarwal, Chang, Suri, Xiao and Xue 2020]

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

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

Dynamic 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 and

Exact depth 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 and

Exact depth 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: $O(\log n)$ update time [Imai and

This Work

Unlikely have sublinear update time for exact depth

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

For (axis-paralle rox. with $\tilde{O}(n^{1/(k+1)})$ amortized update time

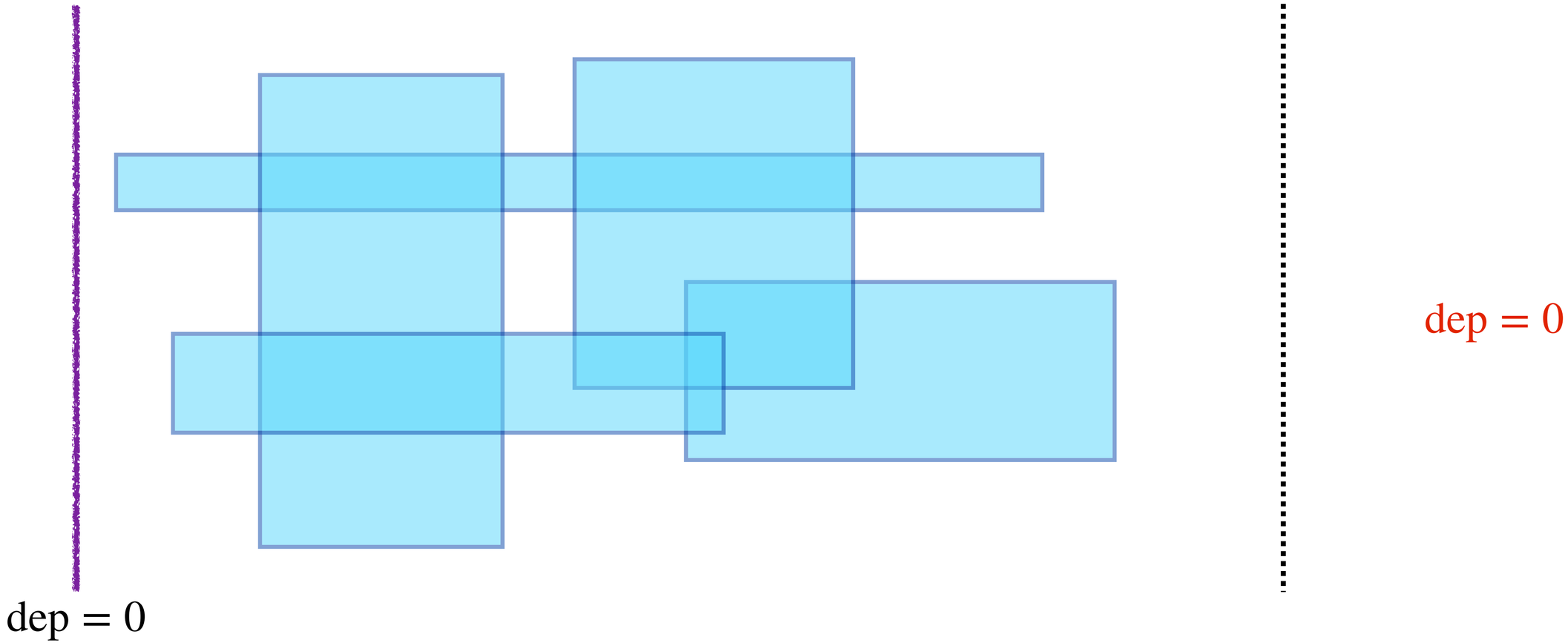
For disks: $(1/2-\varepsilon)$-approx. with $\tilde{O}(n^{2/3})$ update time

Warm Up: Sweep-Line Algorithm for Rectangles

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

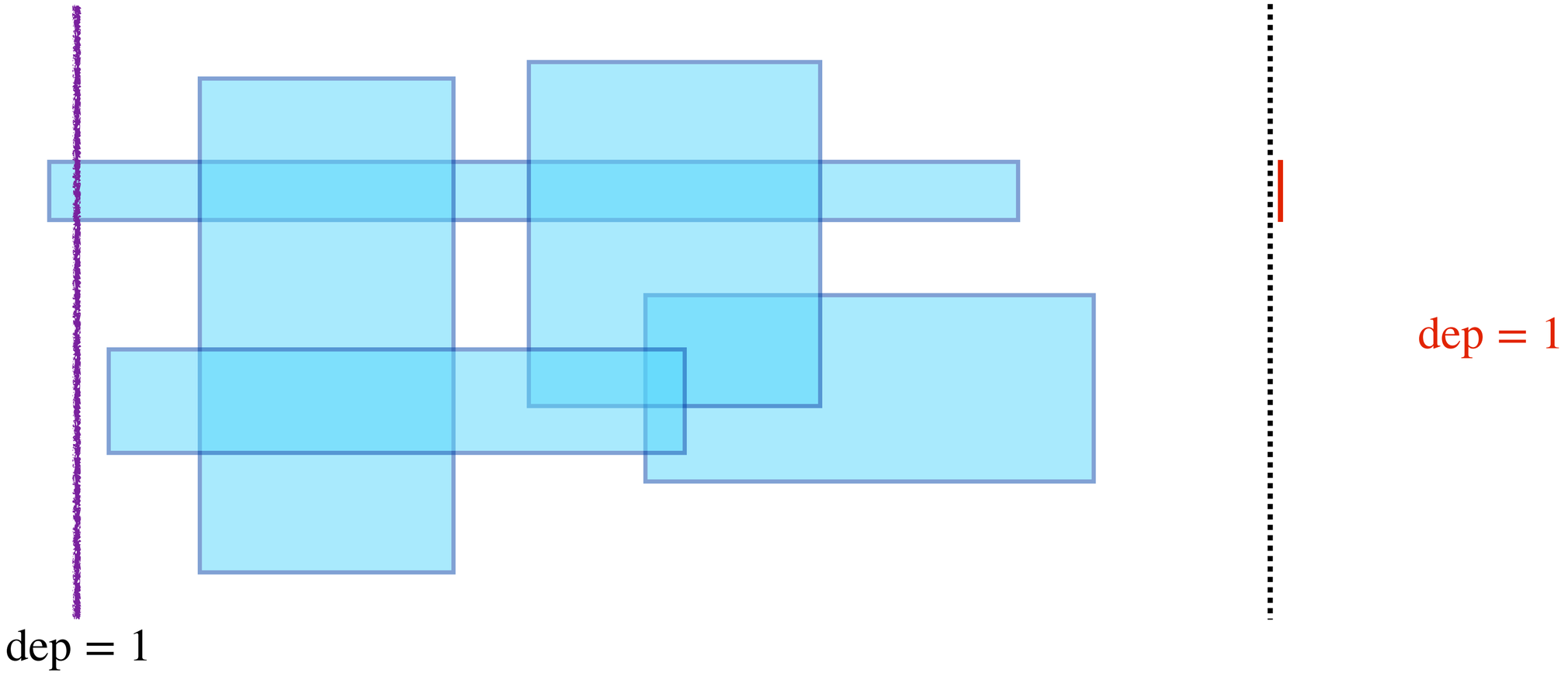
Warm Up: Sweep-Line Algorithm for Rectangles

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



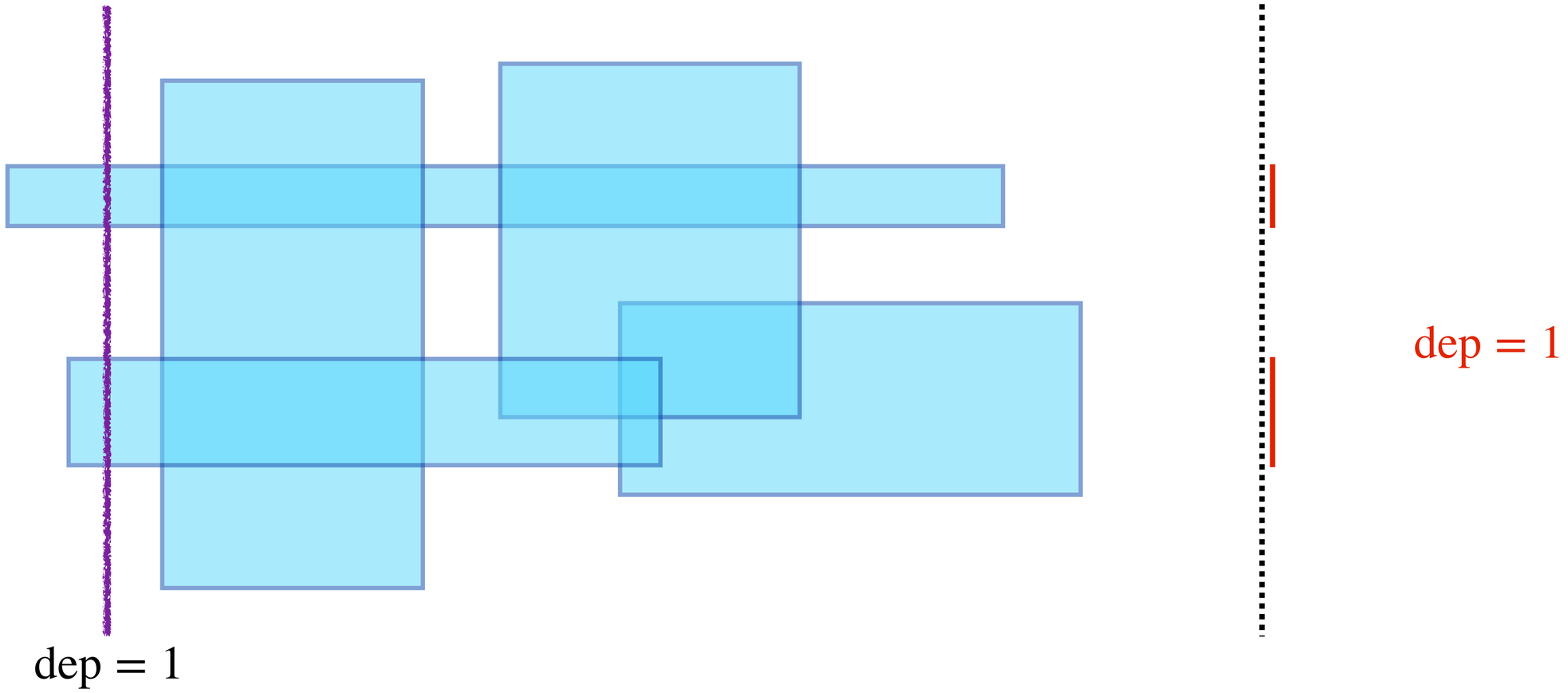
Warm Up: Sweep-Line Algorithm for Rectangles

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



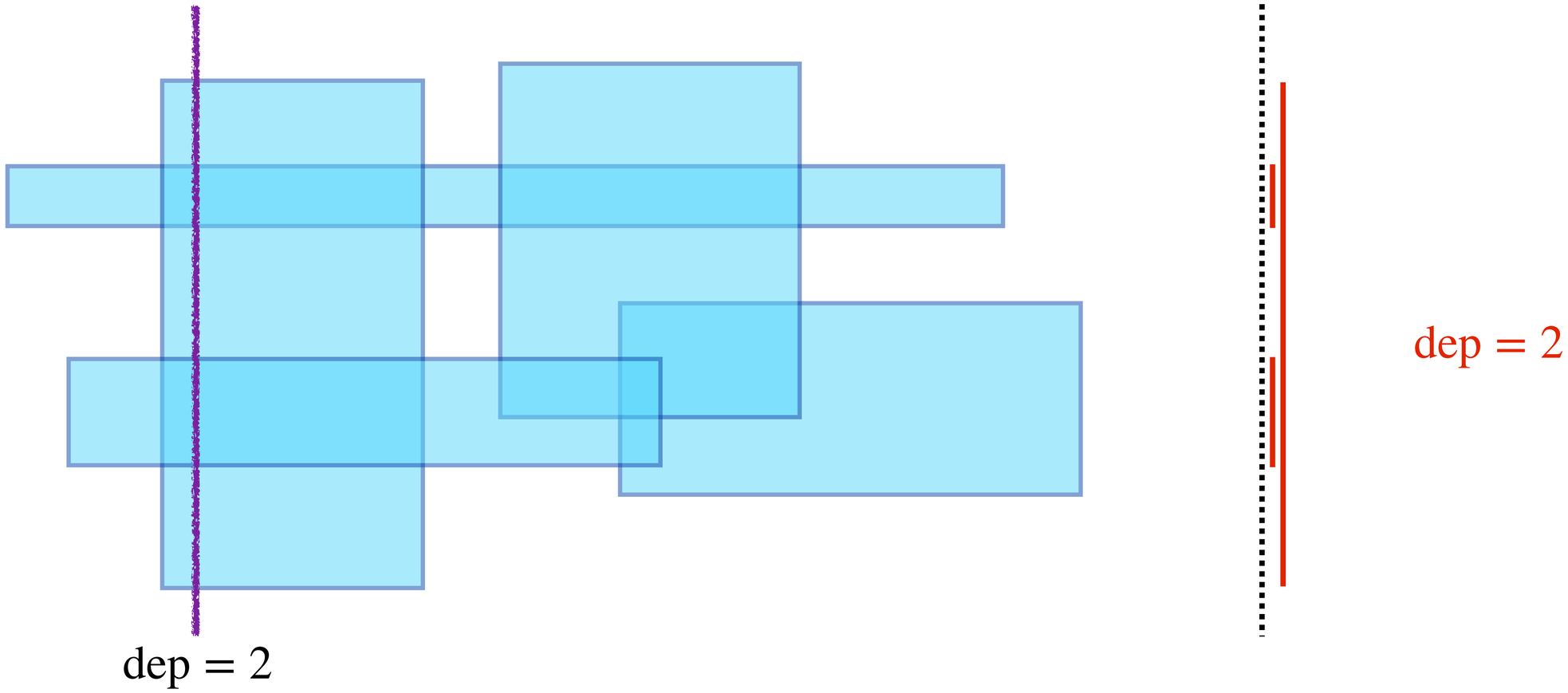
Warm Up: Sweep-Line Algorithm for Rectangles

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



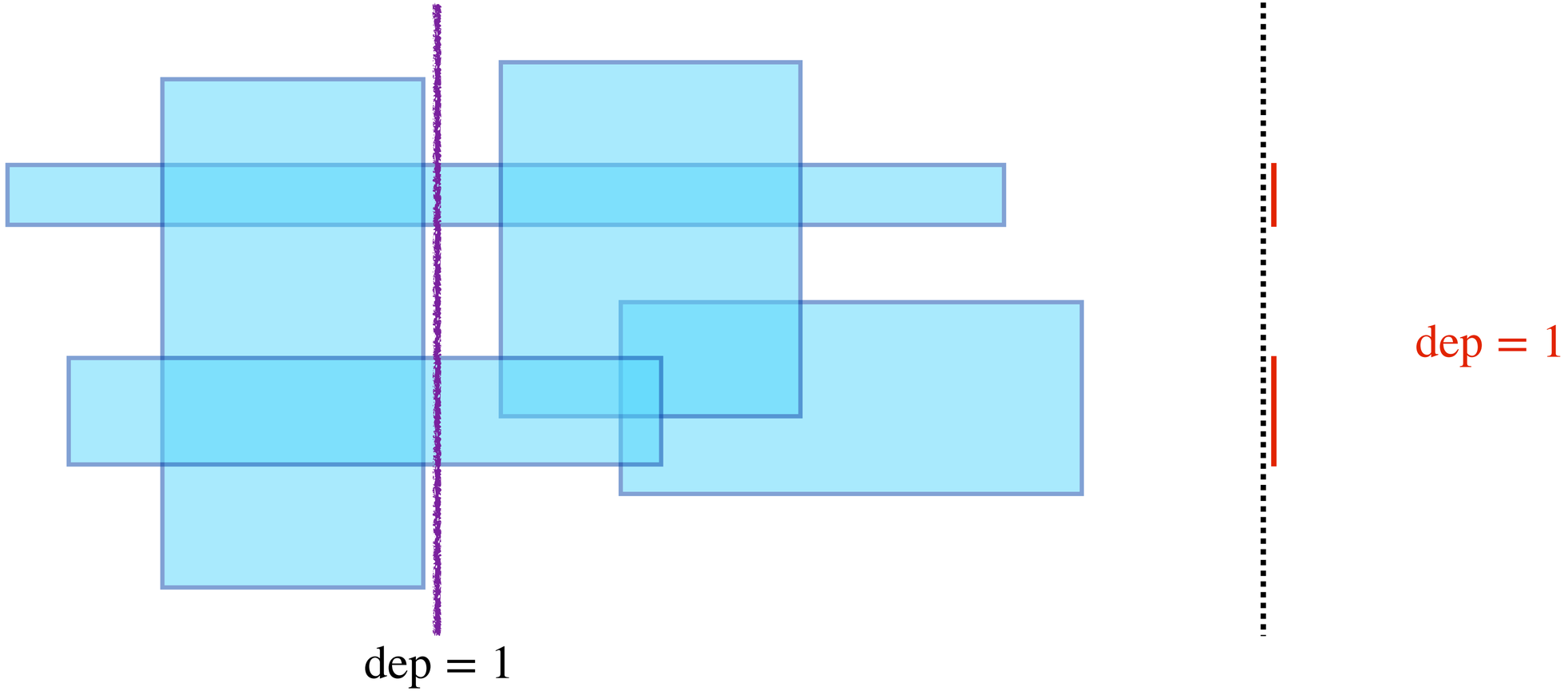
Warm Up: Sweep-Line Algorithm for Rectangles

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



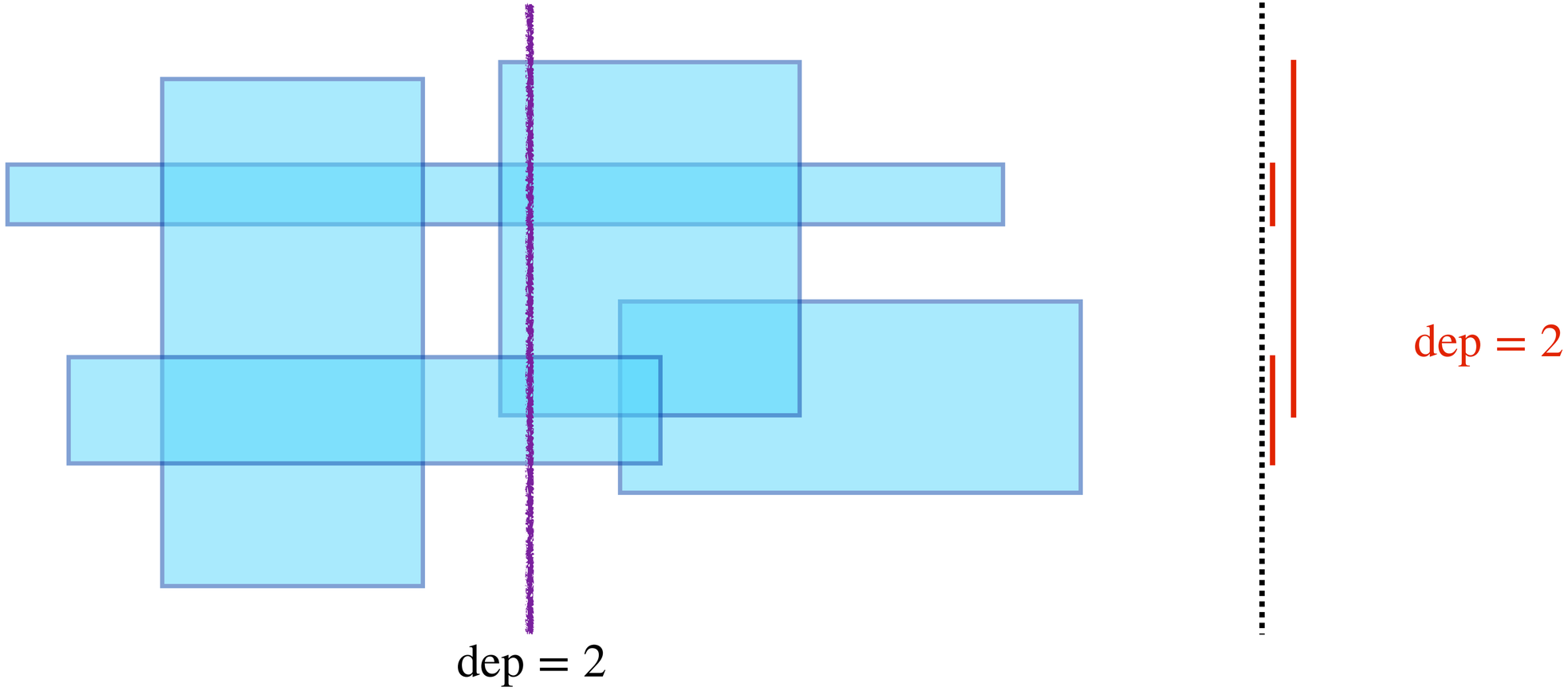
Warm Up: Sweep-Line Algorithm for Rectangles

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



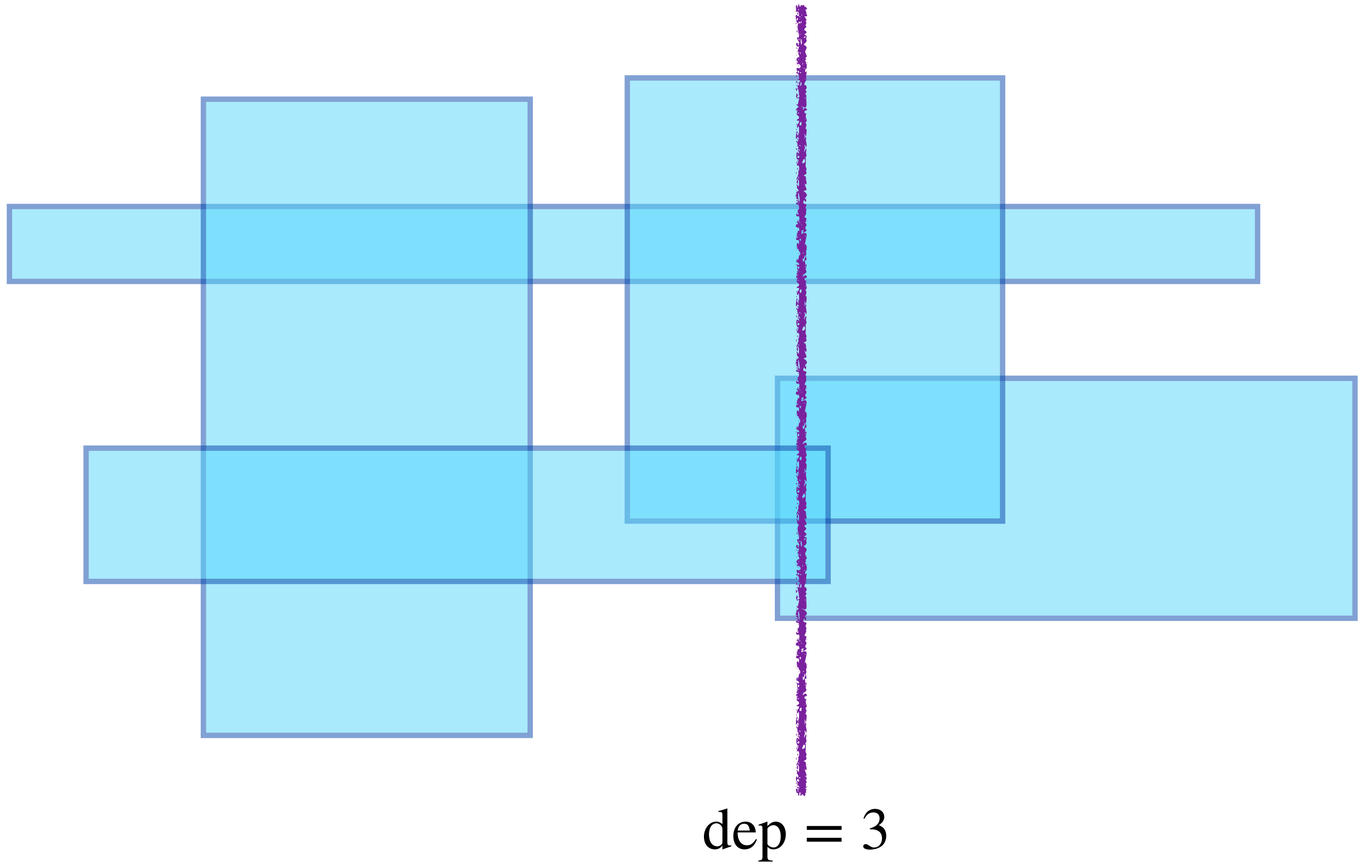
Warm Up: Sweep-Line Algorithm for Rectangles

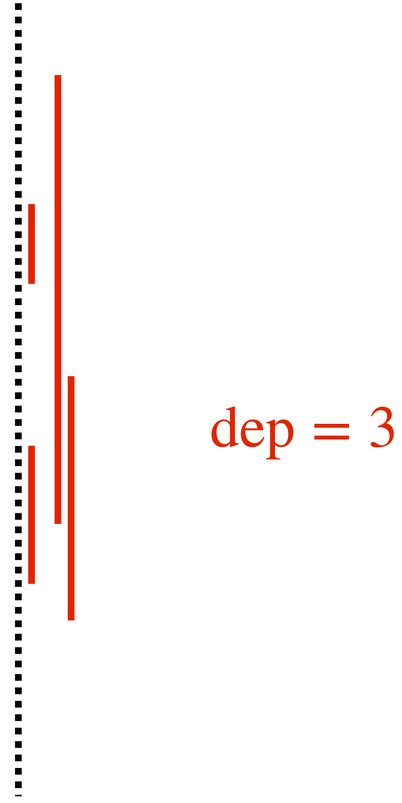
For rectangles: $O(n \log n)$-time algorithm via sweep line [Imai and Asano 1977]



Warm Up: Sweep-Line Algorithm for Rectangles

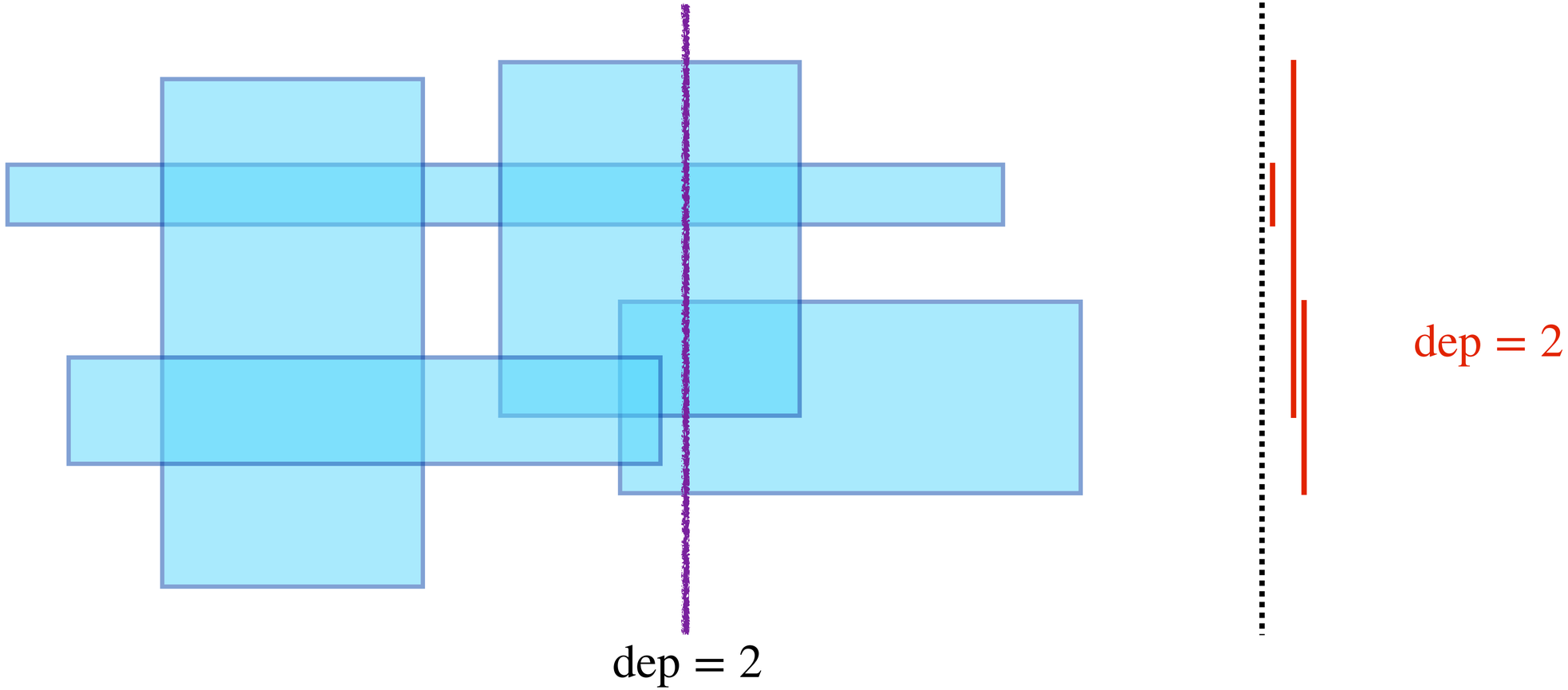
For rectangles: $O(n \log n)$-time algorithm via sweep line [Imai and Asano 1977]





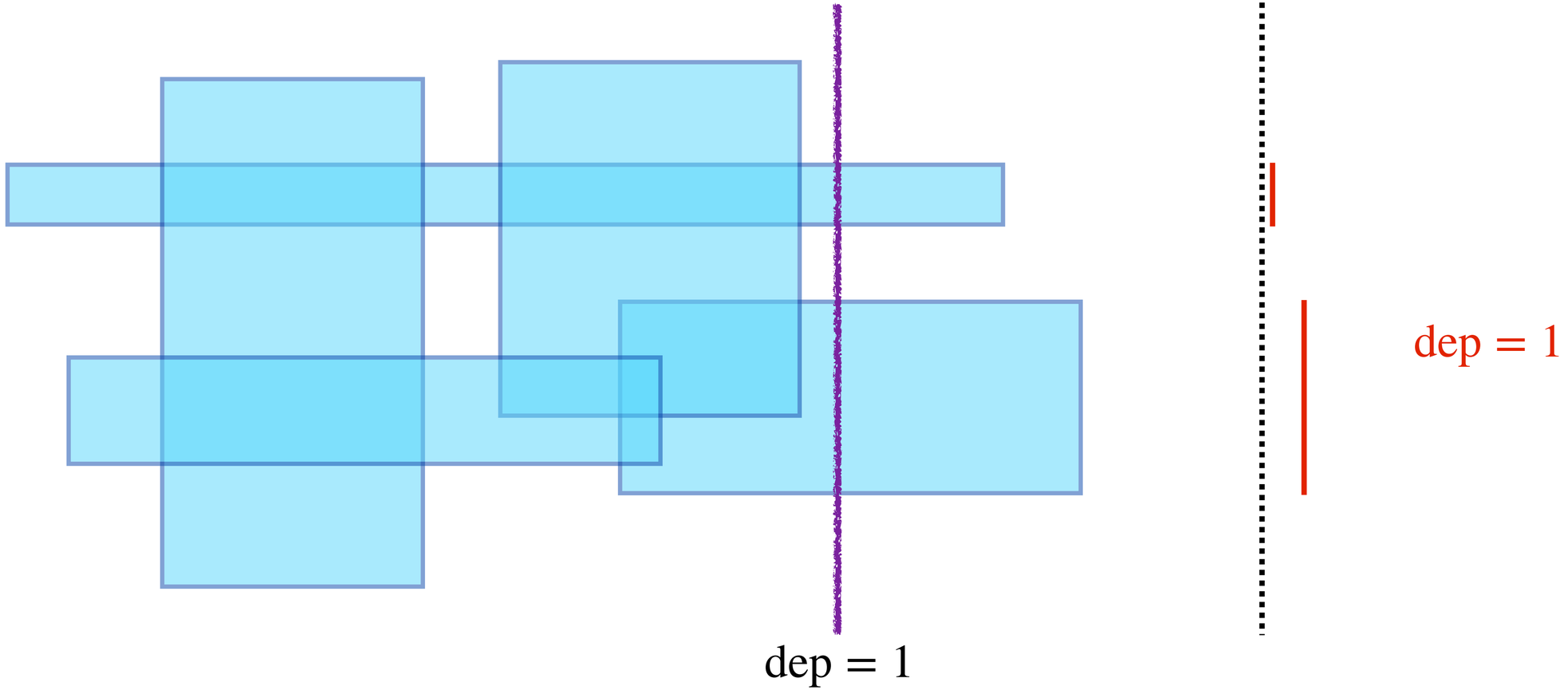
Warm Up: Sweep-Line Algorithm for Rectangles

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



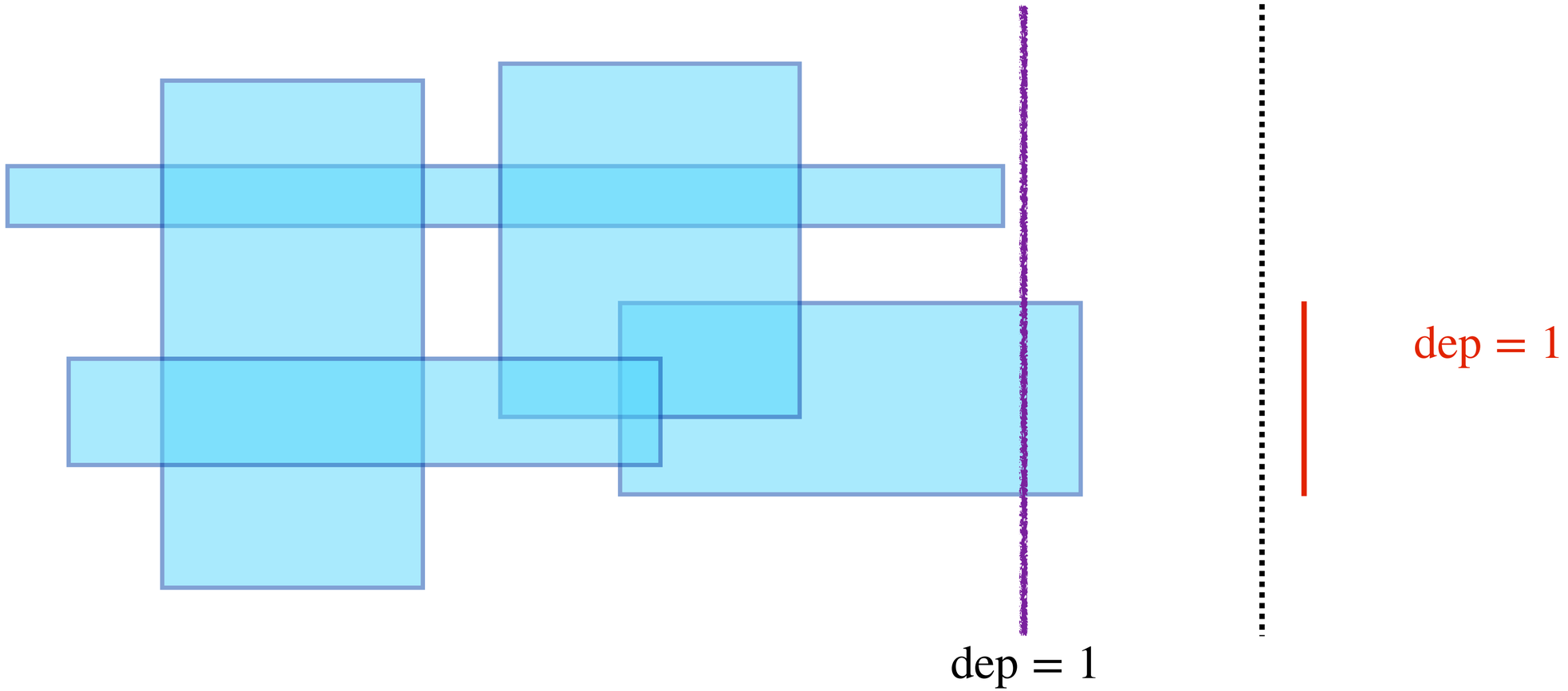
Warm Up: Sweep-Line Algorithm for Rectangles

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



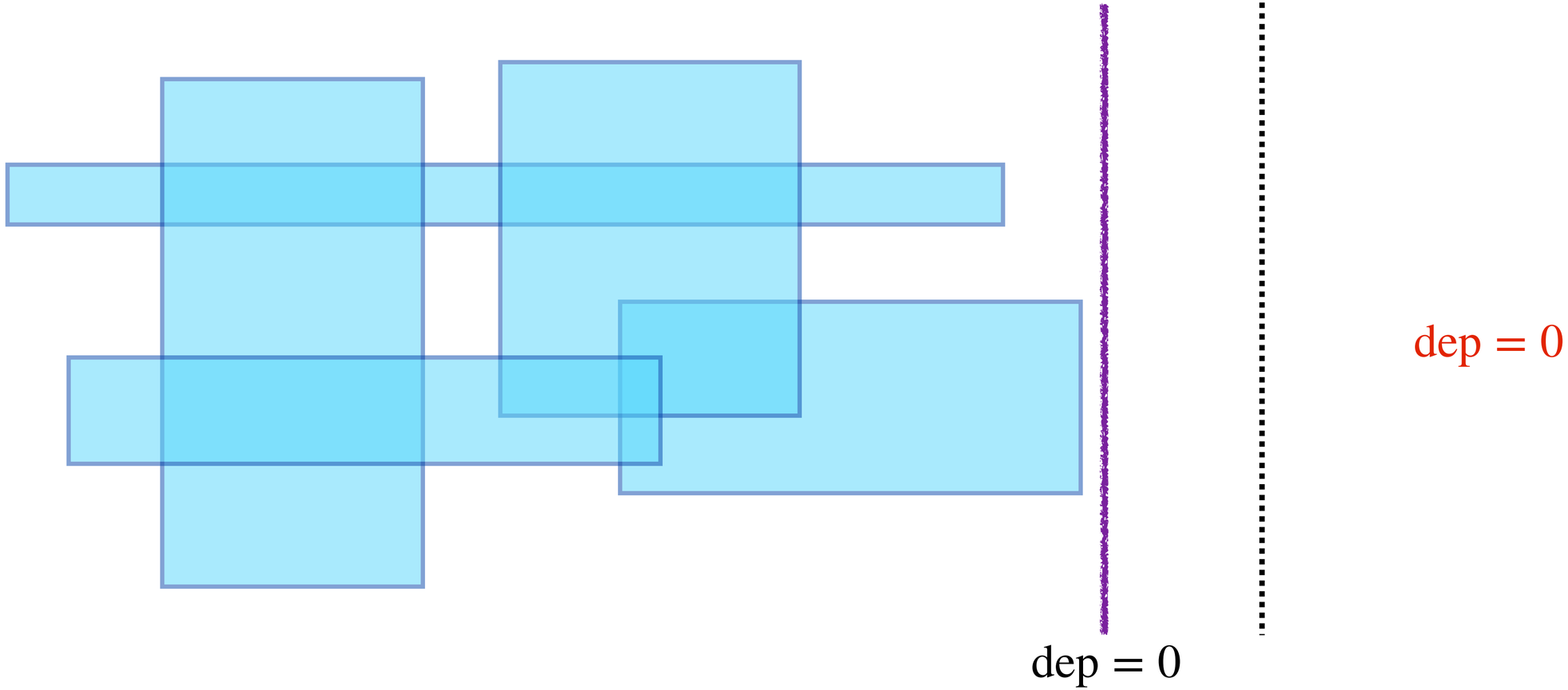
Warm Up: Sweep-Line Algorithm for Rectangles

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



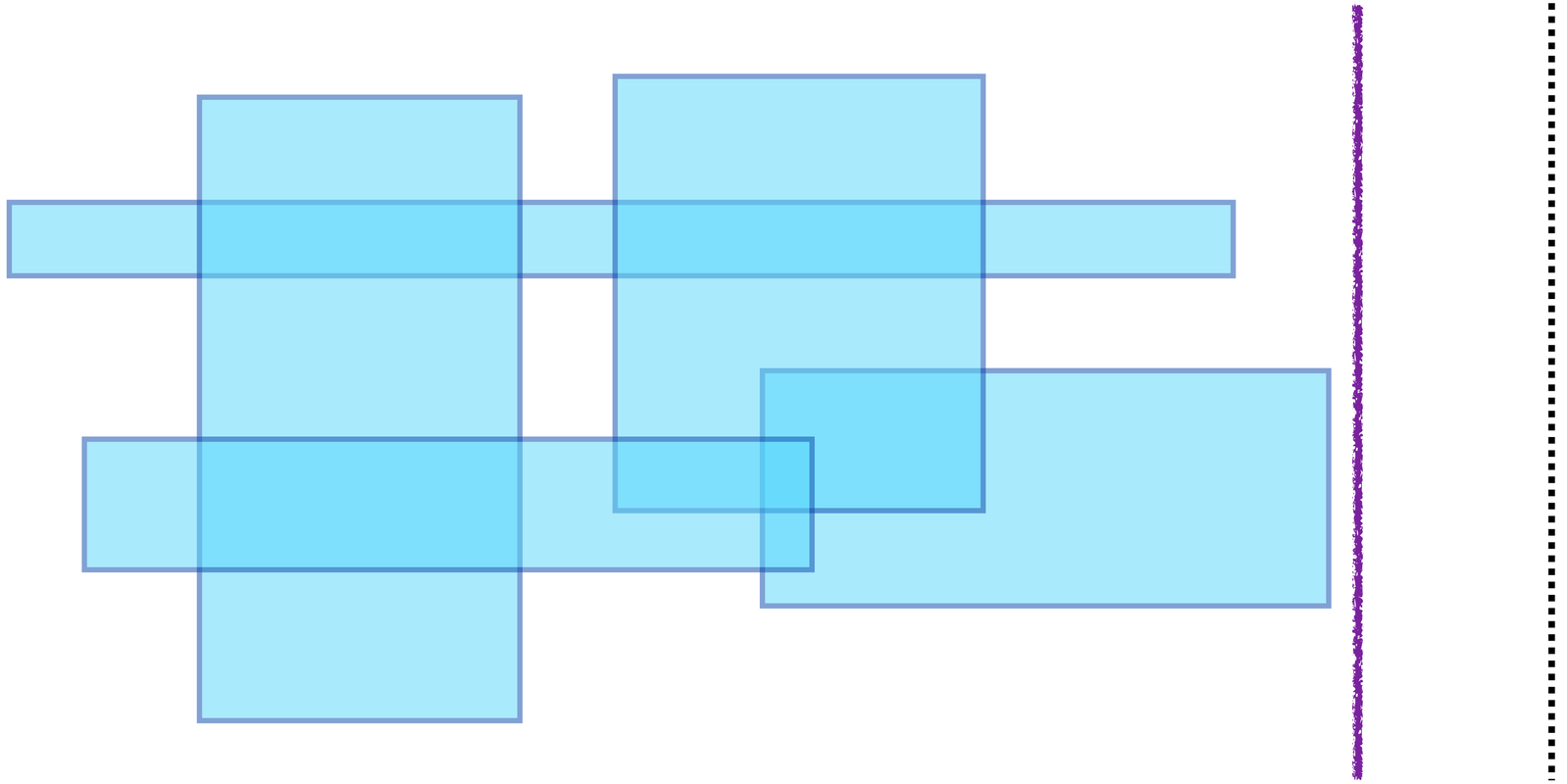
Warm Up: Sweep-Line Algorithm for Rectangles

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



Warm Up: Sweep-Line Algorithm for Rectangles

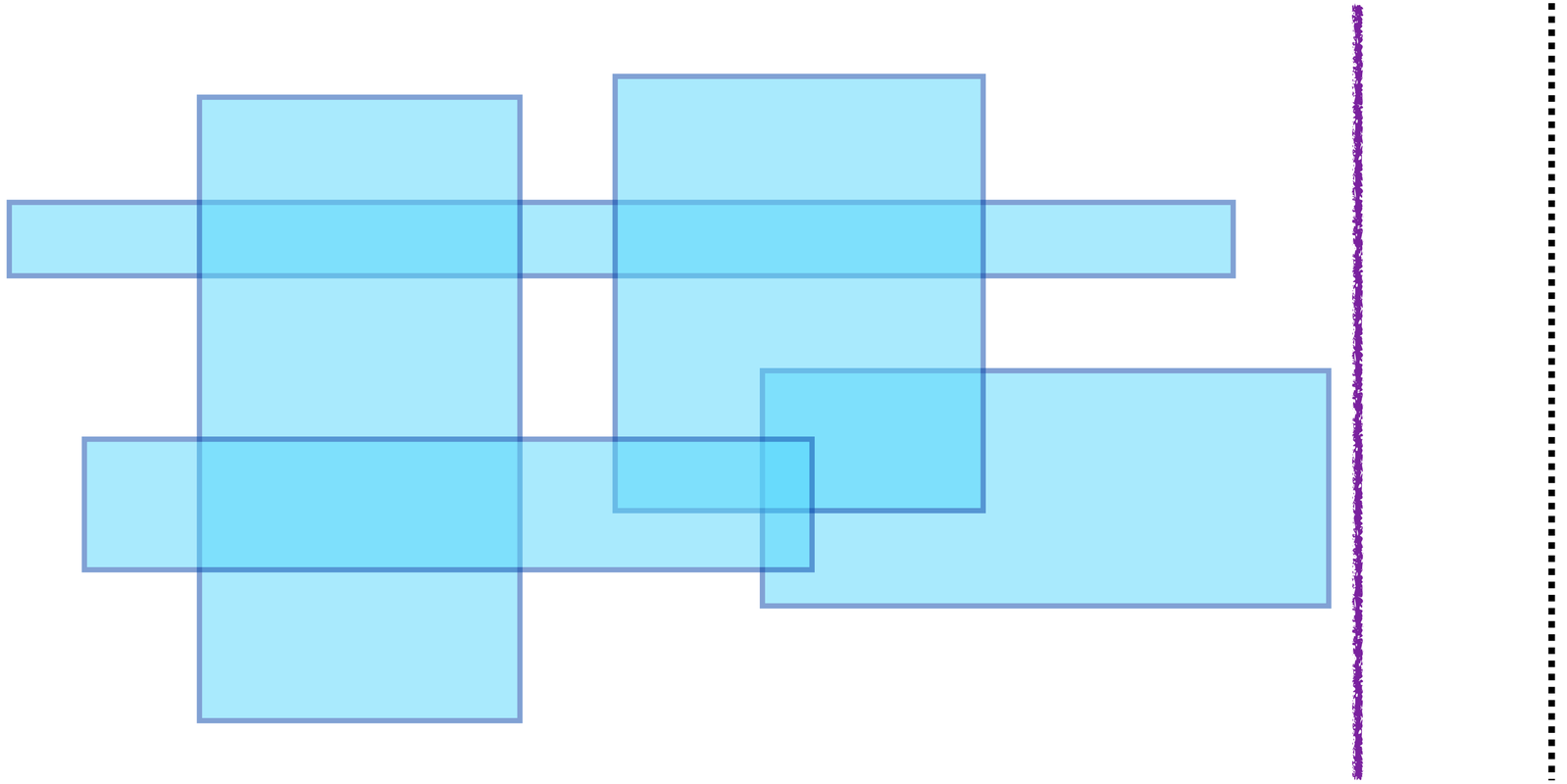
For rectangles: $O(n \log n)$-time algorithm via sweep line [Imai and Asano 1977]



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]

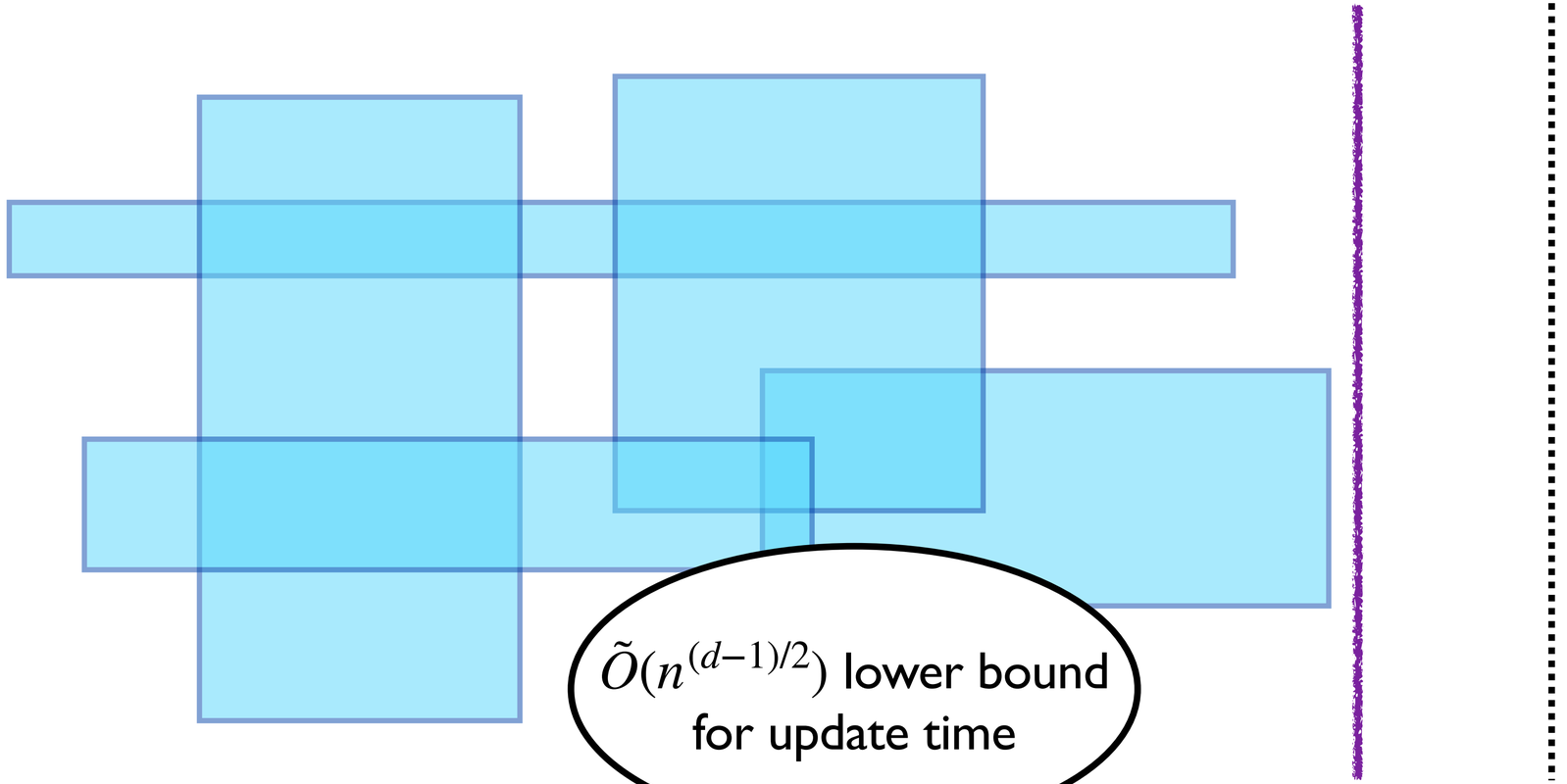


Key ingredient: **dynamic** data structure for intervals with $O(\log n)$ update time

$O(n^c)$ update time in $\mathbb{R}^d \xrightarrow{\text{sweep line}} O(n^{c+1})$ static algorithm in $\mathbb{R}^{d+1}$

Warm Up: Sweep-Line Algorithm for Rectangles

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

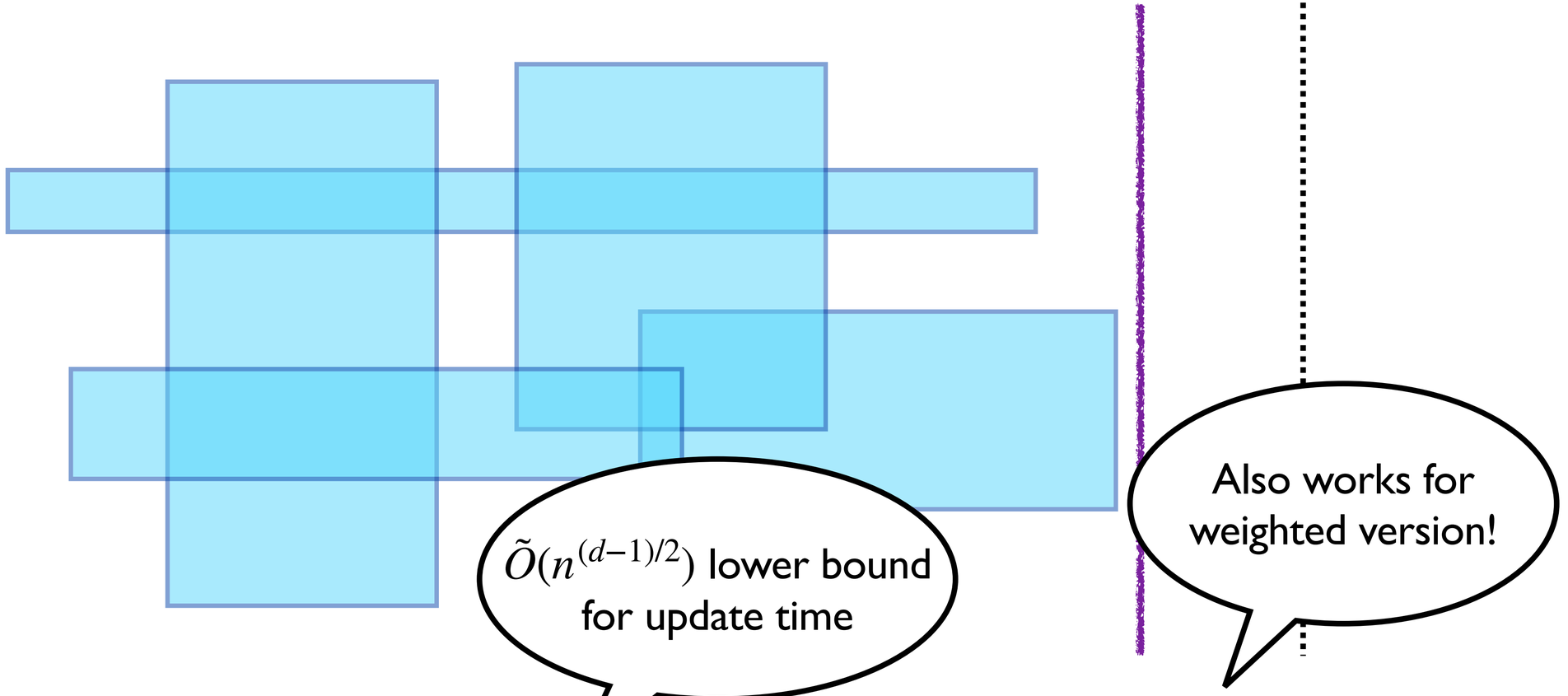


Key ingredient: **dynamic** data structure for intervals with $O(\log n)$ update time

$O(n^c)$ update time in $\mathbb{R}^d \xrightarrow{\text{sweep line}} O(n^{c+1})$ static algorithm in $\mathbb{R}^{d+1}$

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

$O(n^c)$ update time in $\mathbb{R}^d$ $\xrightarrow{\text{sweep line}}$ $O(n^{c+1})$ static algorithm in $\mathbb{R}^{d+1}$

Rectangle: Optimal Update via Partition

Rectangle: Optimal Update via Partition

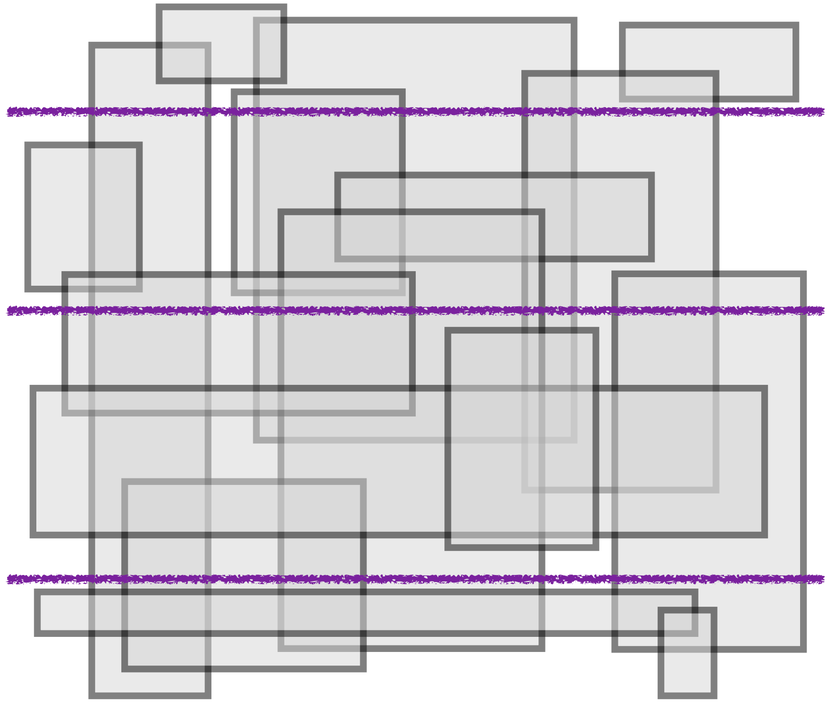
Partition the plane

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

Rectangle: Optimal Update via Partition

Partition the plane

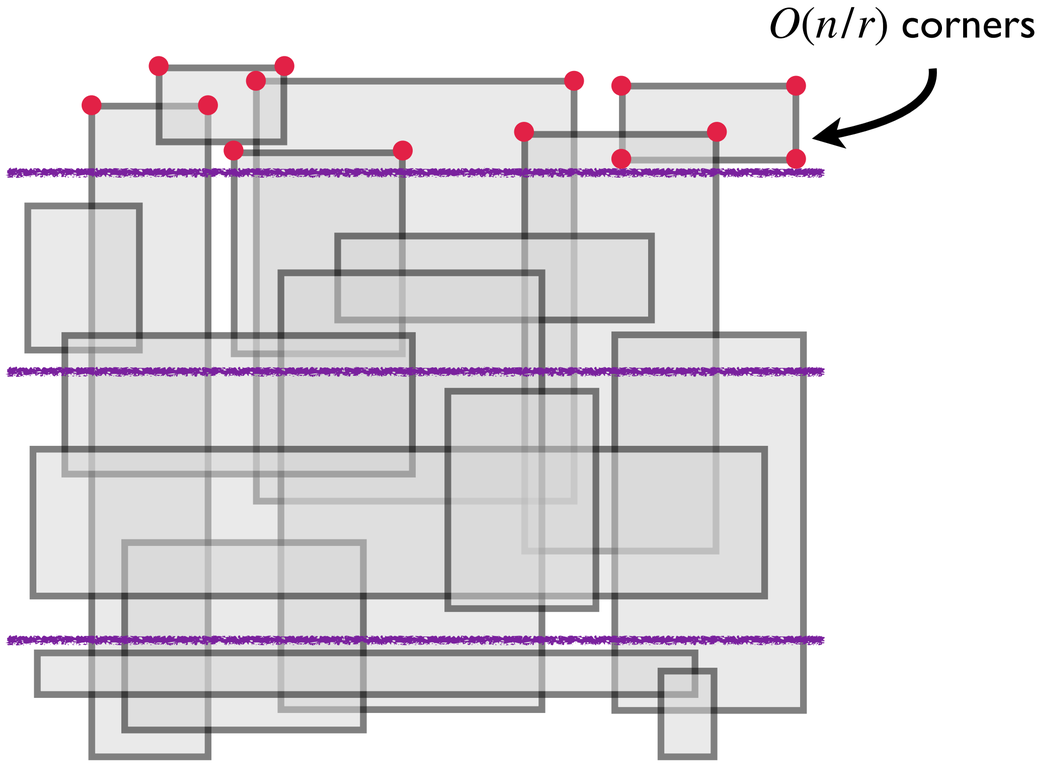
- Partition the plane into r slabs
- Each slab have $O(n/r)$ corners of rectangles



Rectangle: Optimal Update via Partition

Partition the plane

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



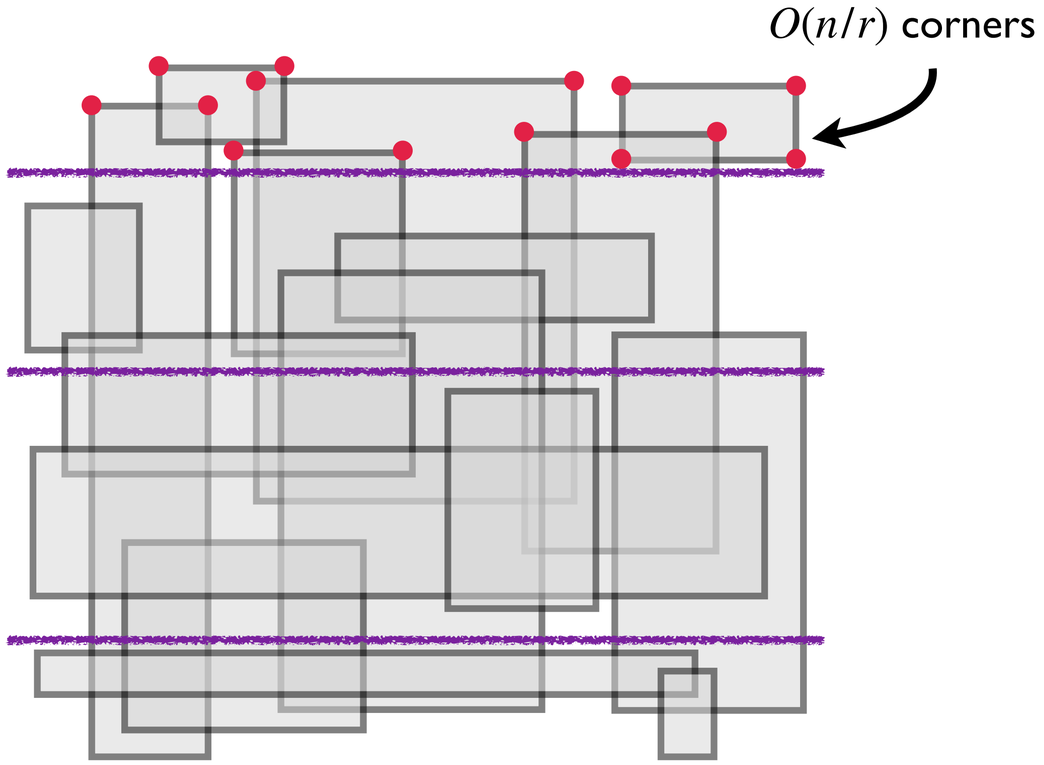
Rectangle: Optimal Update via Partition

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
Pick the largest one



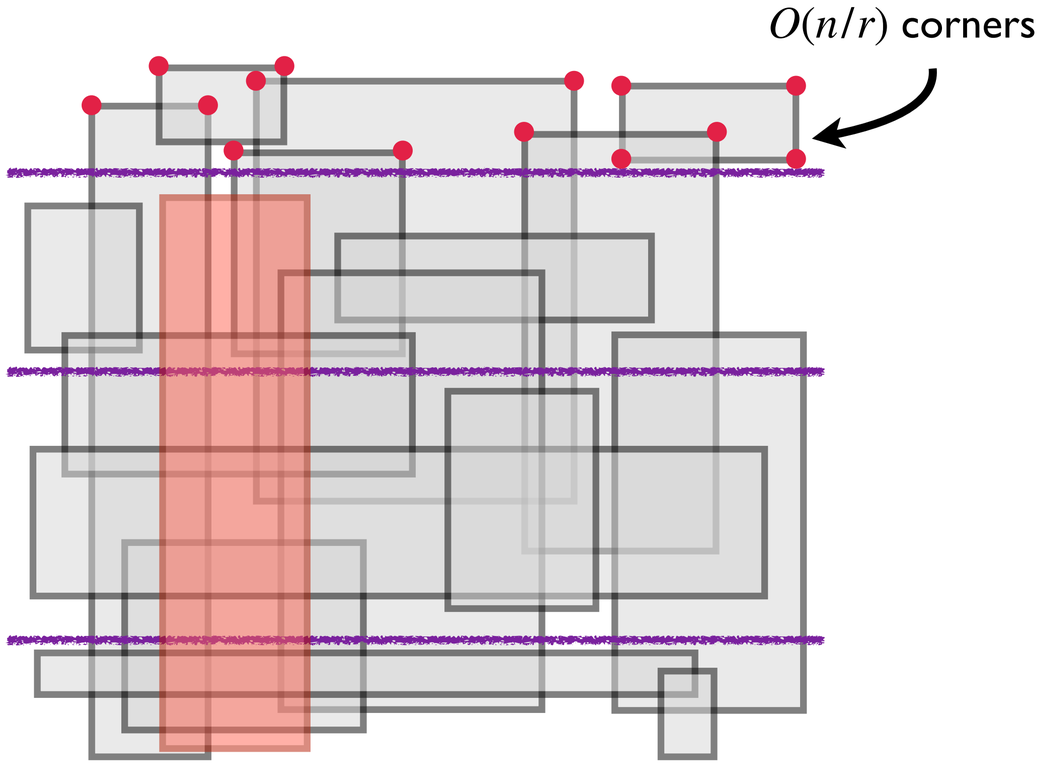
Rectangle: Optimal Update via Partition

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
Pick the largest one



Rectangle: Optimal Update via Partition

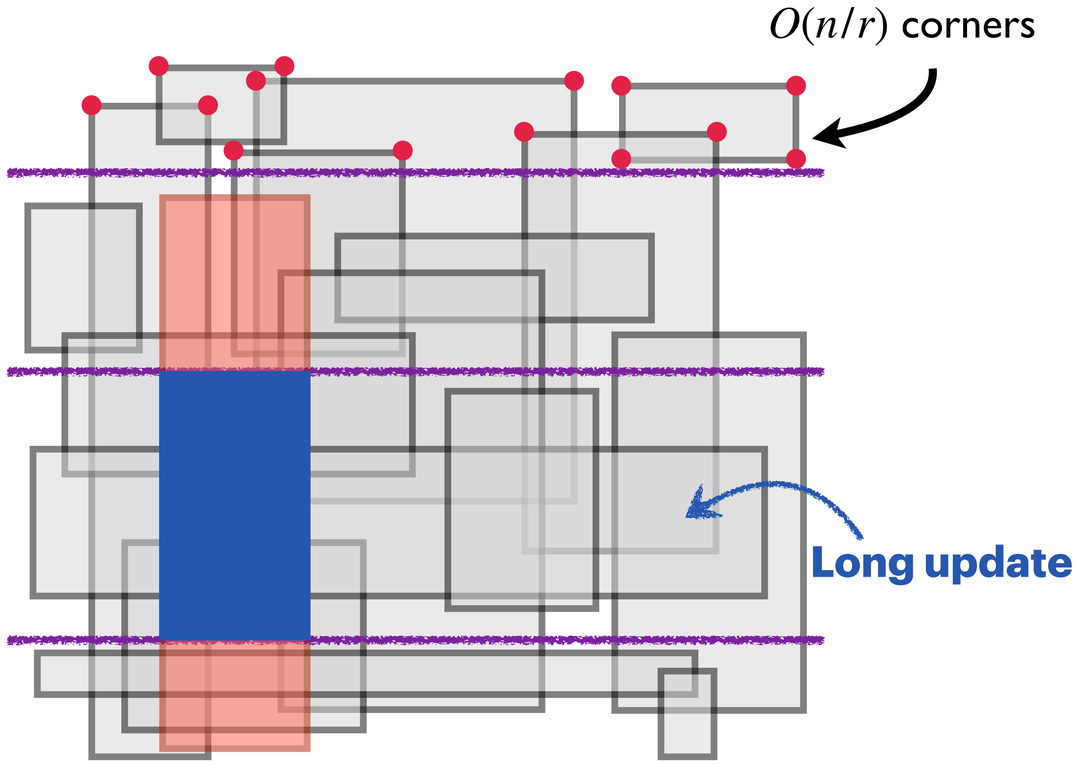
Partition the plane

- 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
Pick the largest one

"Long" update



Rectangle: Optimal Update via Partition

Partition the plane

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

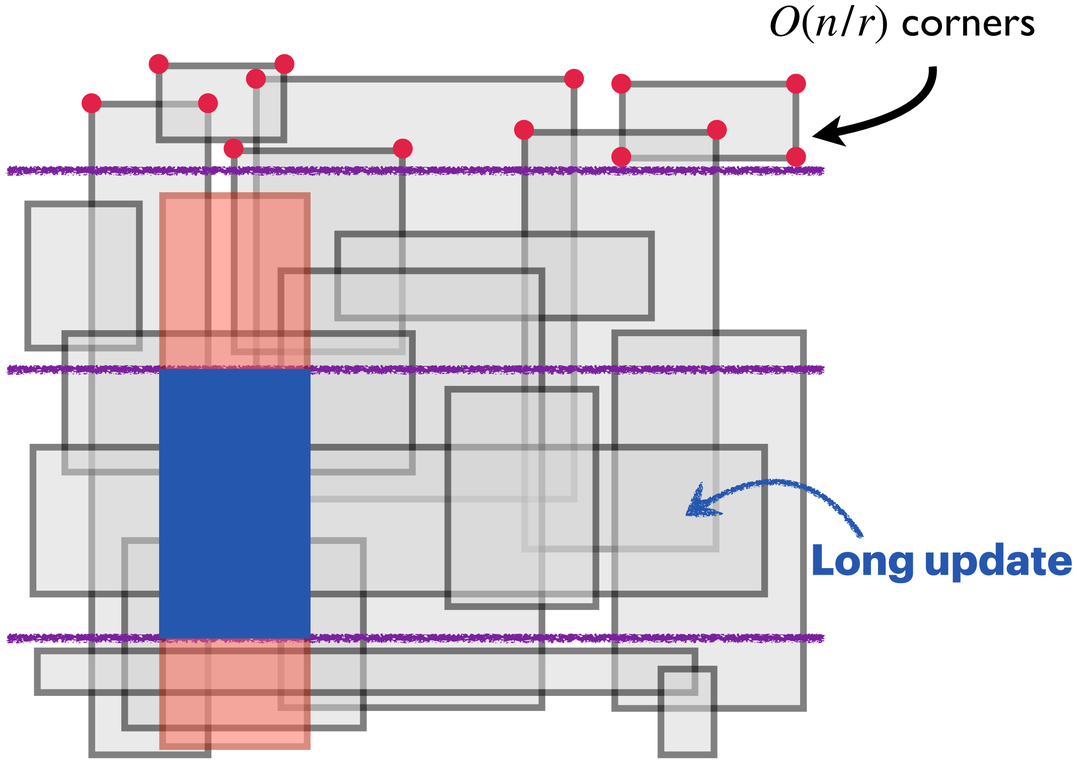
Update in each slab

Maintain the max depth in each slab
Pick the largest one

"Long" update

😇 easy: $O(\log n)$

😈 many: $O(r)$



Rectangle: Optimal 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

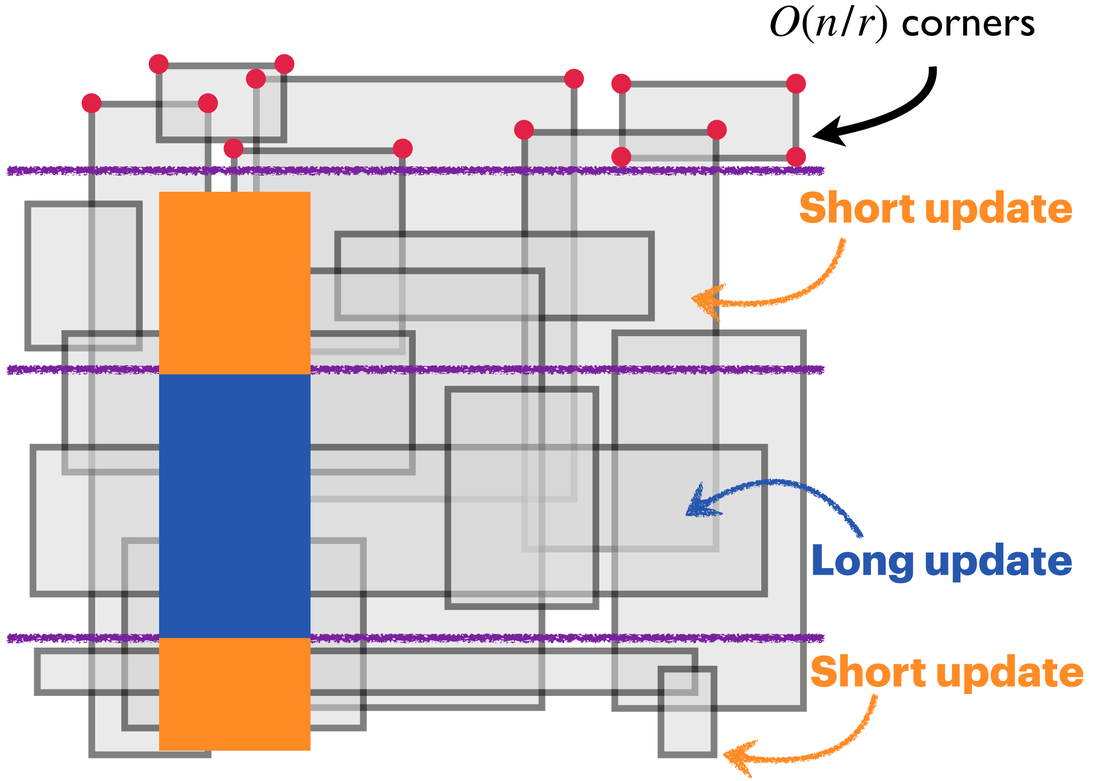
Maintain the max depth in each slab
Pick the largest one

"Long" update

😇 easy: $O(\log n)$

😈 many: $O(r)$

"Short" update



Rectangle: Optimal Update via Partition

Partition the plane

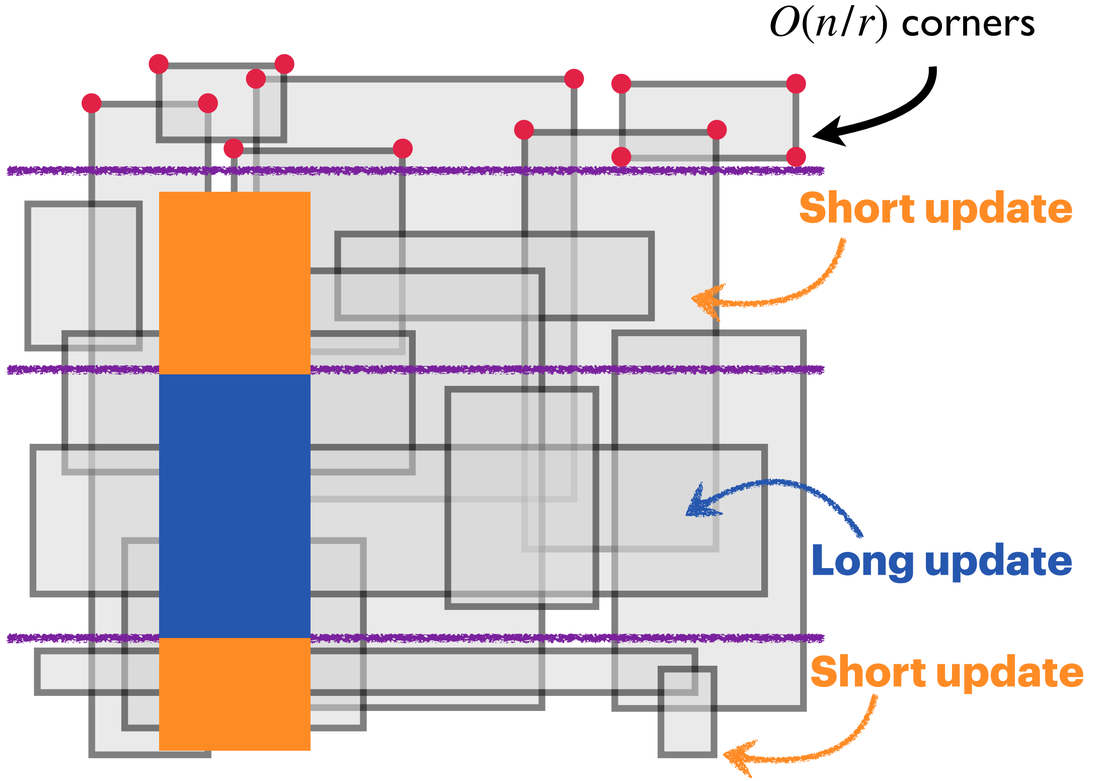
- Partition the plane into r slabs
- Each slab have $O(n/r)$ corners of rectangles

Update in each slab

Maintain the max depth in each slab
Pick the largest one

"Long" update
- 😇 easy: $O(\log n)$
- 😈 many: $O(r)$

"Short" update
- 😈 hard: $O((n/r)\log n)$
- 😇 rare: ≤ 2



Rectangle: Optimal Update via Partition

Partition the plane

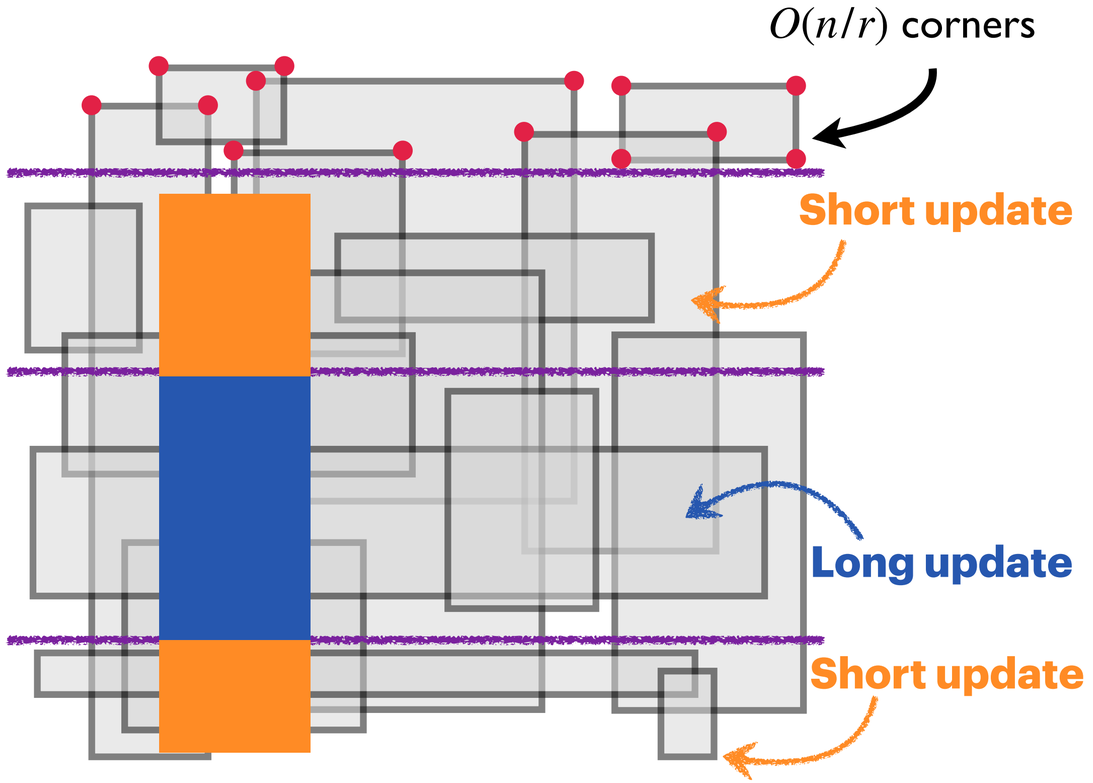
- Partition the plane into r slabs
- Each slab have $O(n/r)$ corners of rectangles

Update in each slab

Maintain the max depth in each slab
Pick the largest one

"Long" update
- 😇 easy: $O(\log n)$
- 😈 many: $O(r)$

"Short" update
- 😈 hard: $O((n/r)\log n)$
- 😇 rare: ≤ 2

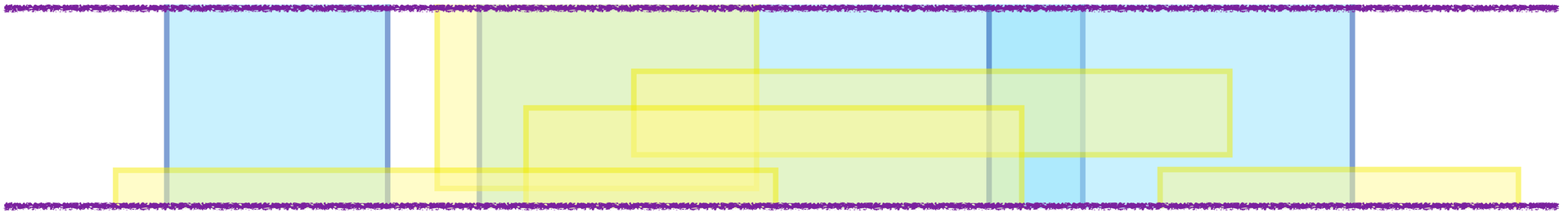


Total update time

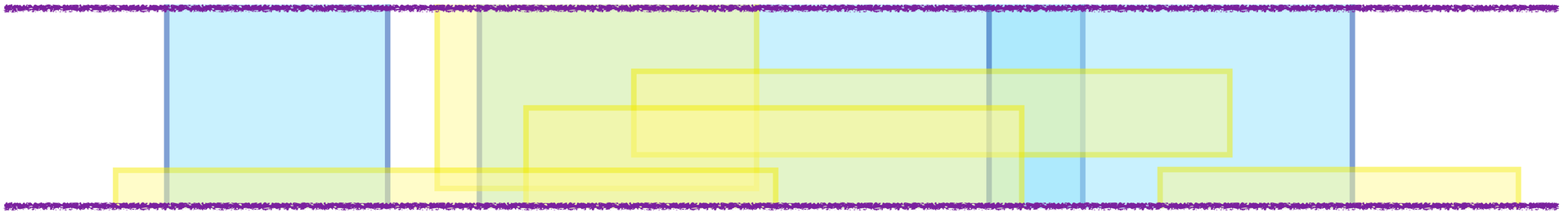
$$O\left(2 \cdot \frac{n}{r} \log n + r \cdot \log n\right) = \tilde{O}\left(\sqrt{n}\right)$$

when $r = \sqrt{n}$

Details of Slab Data Structure

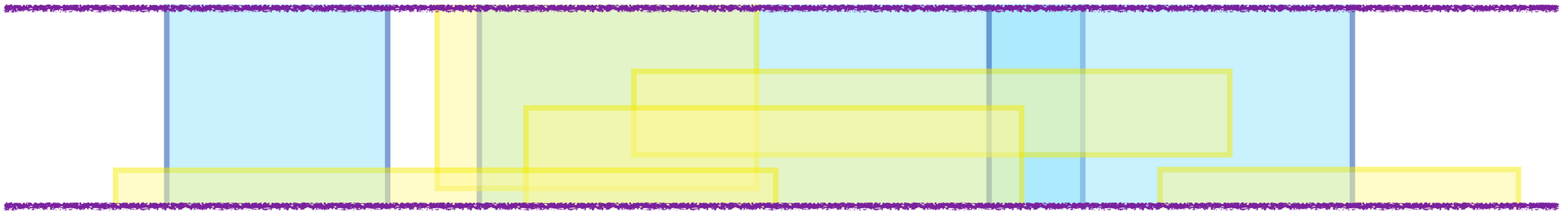


Details of Slab Data Structure



Reduce to intervals

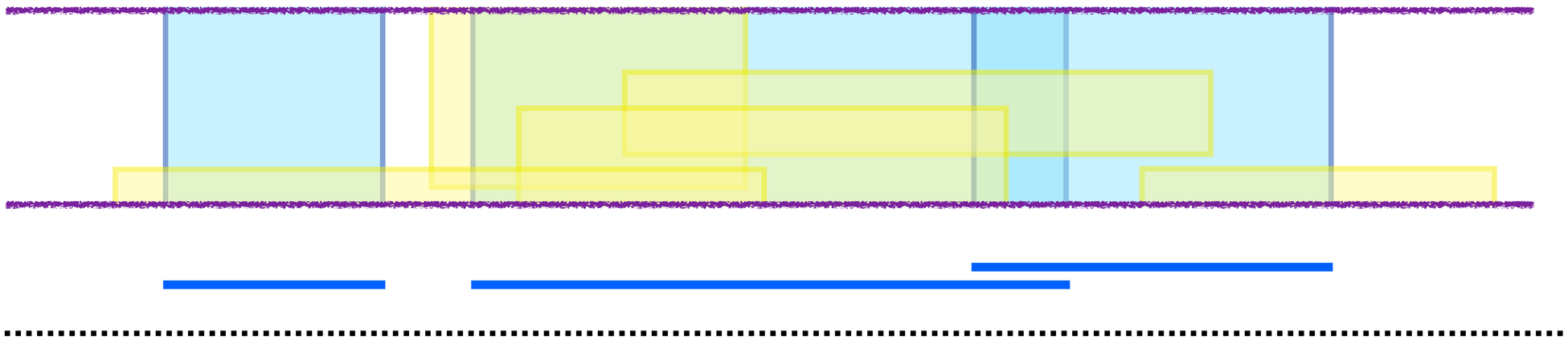
Details of Slab Data Structure



Reduce to intervals

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

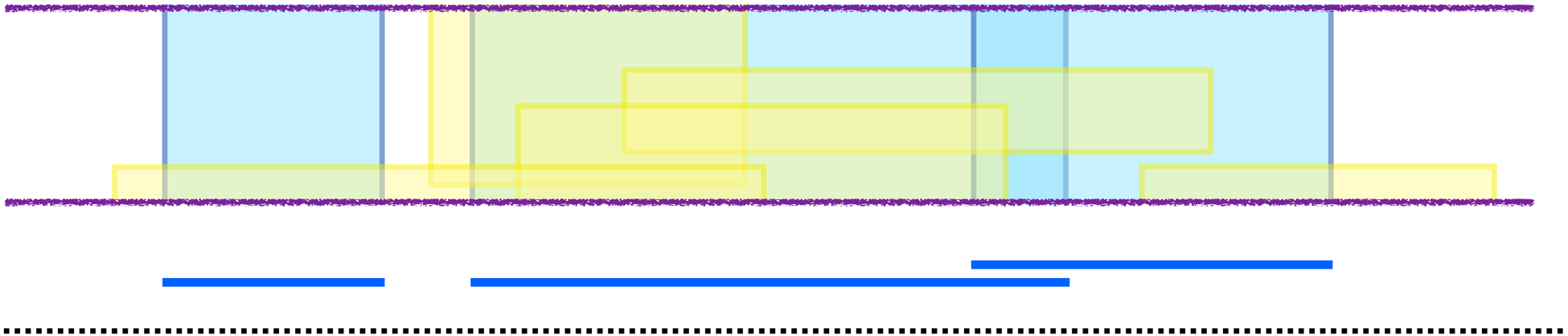
Details of Slab Data Structure



Reduce to intervals

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

Details of Slab Data Structure



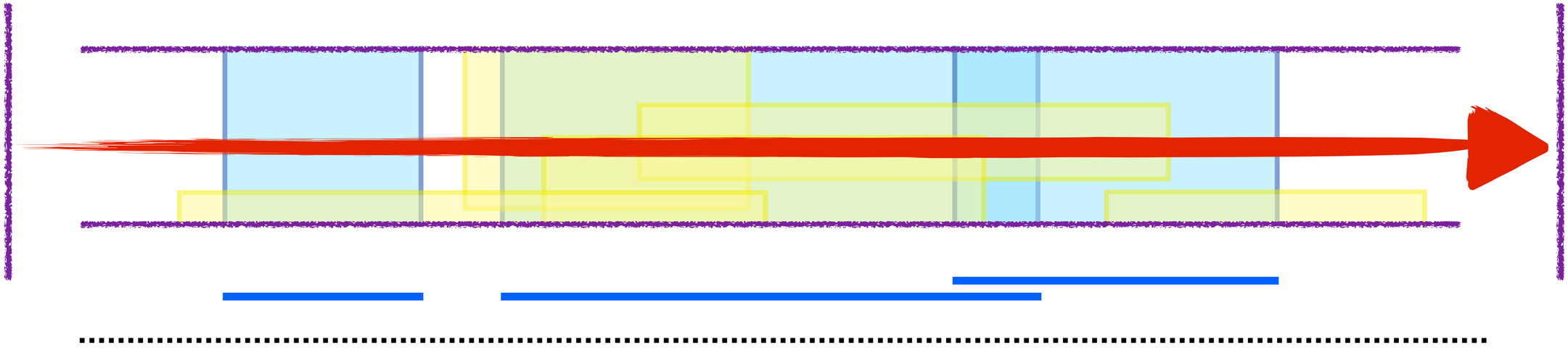
Reduce to intervals

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

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

interval: two adjacent boundaries
weight: max dep w.r.t. short rectangles

Details of Slab Data Structure



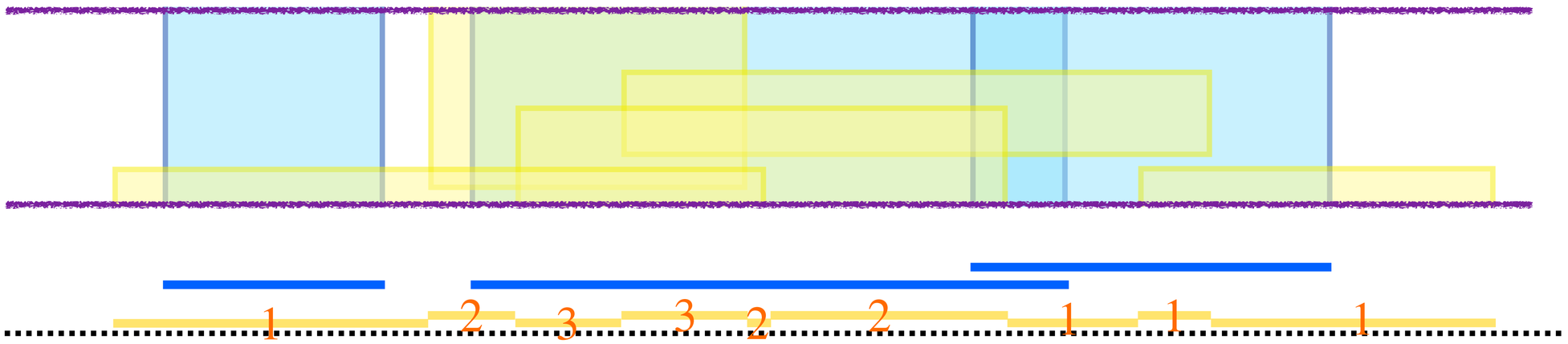
Reduce to intervals

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

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

interval: two adjacent boundaries
weight: max dep w.r.t. short rectangles

Details of Slab Data Structure



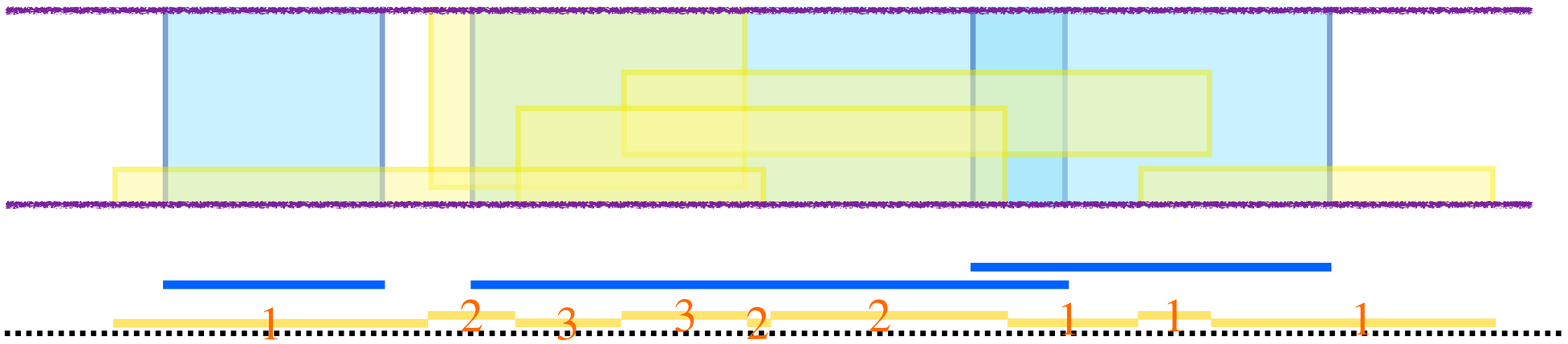
Reduce to intervals

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

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

interval: two adjacent boundaries
weight: max dep w.r.t. short rectangles

Details of Slab Data Structure



Reduce to intervals

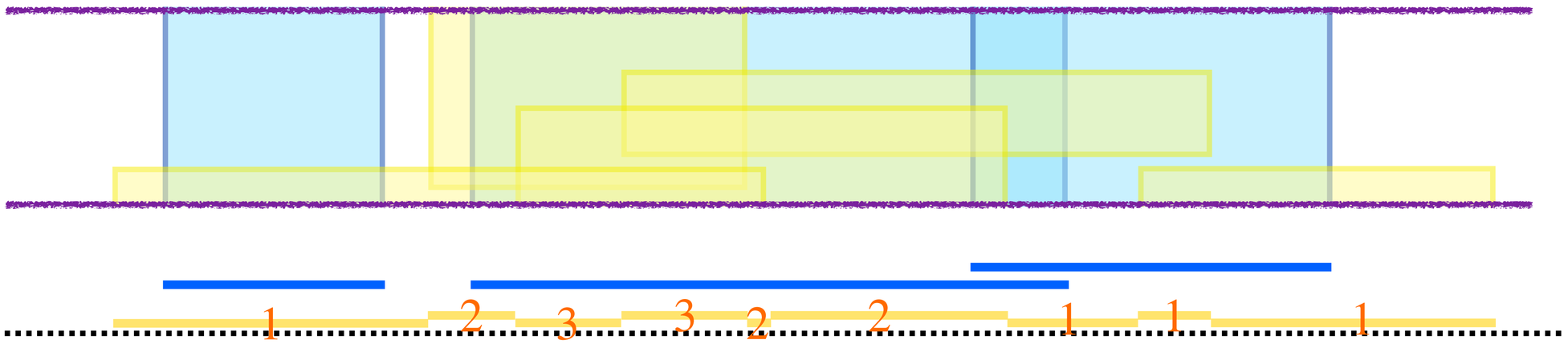
$O(n)$ long rectangles $\xrightarrow[O(n) \text{ time}]{\text{directly}}$ $O(n)$ unweight intervals

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

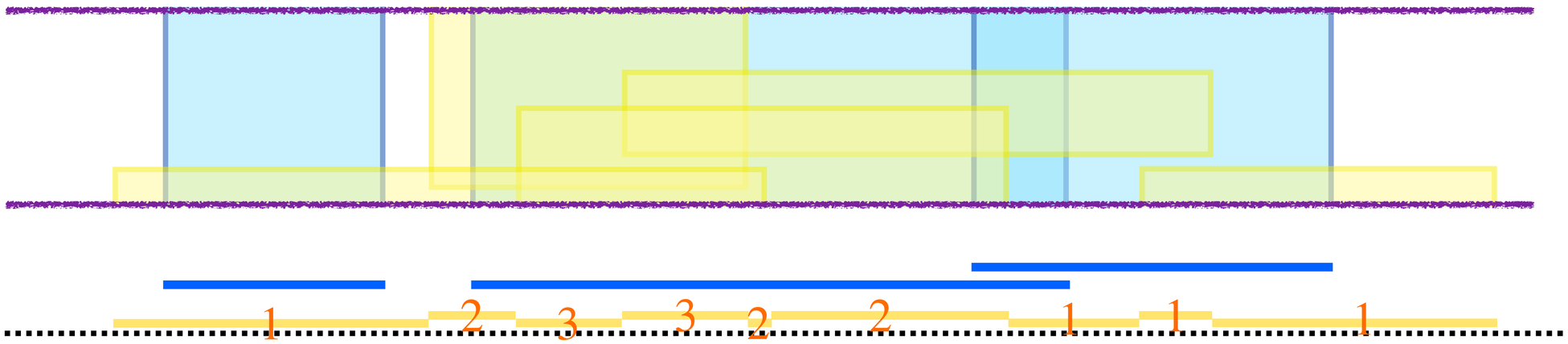
$O\left(\frac{n}{r}\right)$ short rectangles $\xrightarrow[o\left(\frac{n}{r}\log\frac{n}{r}\right) \text{ time}]{\text{sweep line}}$ $O(n/r)$ weighted intervals

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

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

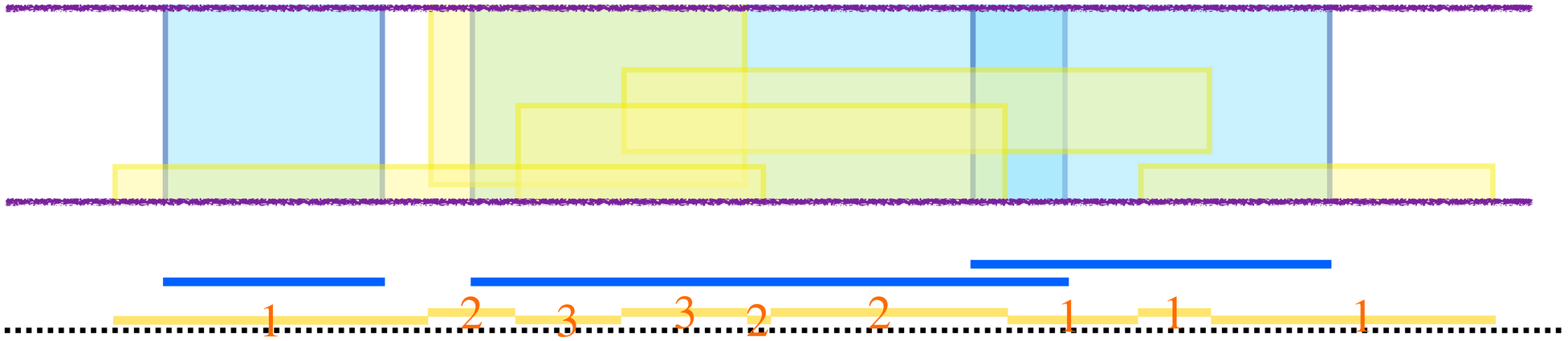
interval: two adjacent boundaries
weight: max dep w.r.t. short rectangles

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

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

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

Approx. Ratio vs Update Time

🤔 Exact data structure: balancing long and short rectangles

Approx. Ratio vs Update Time

🤔 Exact data structure: balancing long and short rectangles

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

Approx. Ratio vs Update Time

$\mathbf{D}_1$: exact data structure
$\mathbf{I}$: interval data structure
$\mathscr{L}$: set of long rect.
$\mathscr{S}$: set of short rect.

🤔 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 $\mathscr{S}$, $\mathbf{I}$ on $\mathscr{L}$

Approx. Ratio vs Update Time

$\mathbf{D}_1$: exact data structure
$\mathbf{I}$: interval data structure
$\mathscr{L}$: set of long rect.
$\mathscr{S}$: set of short rect.

🤔 Exact data structure: balancing long and short rectangles

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

	Construction	Approx. ratio
$\mathbf{D}_2$:	Build $\mathbf{D}_1$ on $\mathscr{S}$, $\mathbf{I}$ on $\mathscr{L}$	$\max\{\text{dep}(\mathscr{S}), \text{dep}(\mathscr{L})\} \geq \frac{1}{2}\text{dep}(\mathscr{O})$

Approx. Ratio vs Update Time

$\mathbf{D}_1$: exact data structure
$\mathbf{I}$: interval data structure
$\mathscr{L}$: set of long rect.
$\mathscr{S}$: set of short rect.

🤔 Exact data structure: balancing long and short rectangles

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

	Construction	Approx. ratio	Update time
$\mathbf{D}_2$:	Build $\mathbf{D}_1$ on $\mathscr{S}$, $\mathbf{I}$ on $\mathscr{L}$	$\max\{\mathrm{dep}(\mathscr{S}), \mathrm{dep}(\mathscr{L})\} \geq \frac{1}{2}\mathrm{dep}(\mathscr{O})$	$\tilde{O}\left(\underset{r = n^{1/3}}{2 \cdot (n/r)^{\frac{1}{2}} + r}\right) = \tilde{O}\left(n^{\frac{1}{3}}\right)$

Approx. Ratio vs Update Time

$\mathbf{D}_1$: exact data structure
$\mathbf{I}$: interval data structure
$\mathscr{L}$: set of long rect.
$\mathscr{S}$: set of short rect.

🤔 Exact data structure: balancing long and short rectangles

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

	Construction	Approx. ratio	Update time
$\mathbf{D}_2$:	Build $\mathbf{D}_1$ on $\mathscr{S}$, $\mathbf{I}$ on $\mathscr{L}$	$\max\left\{\mathrm{dep}(\mathscr{S}), \mathrm{dep}(\mathscr{L})\right\} \geq \frac{1}{2}\mathrm{dep}(\mathscr{O})$	$\tilde{O}\left(2 \cdot (n/r)^{\frac{1}{2}} + r\right) = \tilde{O}\left(n^{\frac{1}{3}}\right)$ $r = n^{1/3}$
$\mathbf{D}_3$:	Build $\mathbf{D}_2$ on $\mathscr{S}$, $\mathbf{I}$ on $\mathscr{L}$		

Approx. Ratio vs Update Time

$\mathbf{D}_1$: exact data structure
$\mathbf{I}$: interval data structure
$\mathscr{L}$: set of long rect.
$\mathcal{S}$: set of short rect.

🤔 Exact data structure: balancing long and short rectangles

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

	Construction	Approx. ratio	Update time
$\mathbf{D}_2$:	Build $\mathbf{D}_1$ on $\mathcal{S}$, $\mathbf{I}$ on $\mathscr{L}$	$\max\{\mathrm{dep}(\mathcal{S}), \mathrm{dep}(\mathscr{L})\} \geq \frac{1}{2}\mathrm{dep}(\mathcal{O})$	$\tilde{O}\left(2 \cdot (n/r)^{\frac{1}{2}} + r\right) = \tilde{O}\left(n^{\frac{1}{3}}\right)$ $r = n^{1/3}$
$\mathbf{D}_3$:	Build $\mathbf{D}_2$ on $\mathcal{S}$, $\mathbf{I}$ on $\mathscr{L}$	$\max\left\{\frac{1}{2}\mathrm{dep}(\mathcal{S}), \mathrm{dep}(\mathscr{L})\right\} \geq \frac{1}{3}\mathrm{dep}(\mathcal{O})$	

Approx. Ratio vs Update Time

$\mathbf{D}_1$: exact data structure
$\mathbf{I}$: interval data structure
$\mathscr{L}$: set of long rect.
$\mathscr{S}$: set of short rect.

🤔 Exact data structure: balancing long and short rectangles

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

	Construction	Approx. ratio	Update time
$\mathbf{D}_2$:	Build $\mathbf{D}_1$ on $\mathscr{S}$, $\mathbf{I}$ on $\mathscr{L}$	$\max\{\mathrm{dep}(\mathscr{S}), \mathrm{dep}(\mathscr{L})\} \geq \frac{1}{2}\mathrm{dep}(\mathscr{O})$	$\tilde{O}\left(\underset{r = n^{1/3}}{2 \cdot (n/r)^{\frac{1}{2}} + r}\right) = \tilde{O}\left(n^{\frac{1}{3}}\right)$
$\mathbf{D}_3$:	Build $\mathbf{D}_2$ on $\mathscr{S}$, $\mathbf{I}$ on $\mathscr{L}$	$\max\left\{\frac{1}{2}\mathrm{dep}(\mathscr{S}), \mathrm{dep}(\mathscr{L})\right\} \geq \frac{1}{3}\mathrm{dep}(\mathscr{O})$	$\tilde{O}\left(\underset{r = n^{1/4}}{2 \cdot (n/r)^{\frac{1}{3}} + r}\right) = \tilde{O}\left(n^{\frac{1}{4}}\right)$

Approx. Ratio vs Update Time

$\mathbf{D}_1$: exact data structure
$\mathbf{I}$: interval data structure
$\mathscr{L}$: set of long rect.
$\mathscr{S}$: set of short rect.

🤔 Exact data structure: balancing long and short rectangles

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

	Construction	Approx. ratio	Update time
$\mathbf{D}_2$:	Build $\mathbf{D}_1$ on $\mathscr{S}$, $\mathbf{I}$ on $\mathscr{L}$	$\max\{\mathrm{dep}(\mathscr{S}), \mathrm{dep}(\mathscr{L})\} \geq \frac{1}{2}\mathrm{dep}(\mathscr{O})$	$\tilde{O}\left(2 \cdot (n/r)^{\frac{1}{2}} + r\right) = \tilde{O}\left(n^{\frac{1}{3}}\right)$ $r = n^{1/3}$
$\mathbf{D}_3$:	Build $\mathbf{D}_2$ on $\mathscr{S}$, $\mathbf{I}$ on $\mathscr{L}$	$\max\left\{\frac{1}{2}\mathrm{dep}(\mathscr{S}), \mathrm{dep}(\mathscr{L})\right\} \geq \frac{1}{3}\mathrm{dep}(\mathscr{O})$	$\tilde{O}\left(2 \cdot (n/r)^{\frac{1}{3}} + r\right) = \tilde{O}\left(n^{\frac{1}{4}}\right)$ $r = n^{1/4}$
• • •	• • •	• • •	

Approx. Ratio vs Update Time

$\mathbf{D}_1$: exact data structure
$\mathbf{I}$: interval data structure
$\mathscr{L}$: set of long rect.
$\mathscr{S}$: set of short rect.

🤔 Exact data structure: balancing long and short rectangles

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

	Construction	Approx. ratio	Update time
$\mathbf{D}_2$:	Build $\mathbf{D}_1$ on $\mathscr{S}$, $\mathbf{I}$ on $\mathscr{L}$	$\max\{\mathrm{dep}(\mathscr{S}), \mathrm{dep}(\mathscr{L})\} \geq \frac{1}{2}\mathrm{dep}(\mathscr{O})$	$\tilde{O}\left(2 \cdot (n/r)^{\frac{1}{2}} + r\right) = \tilde{O}\left(n^{\frac{1}{3}}\right)$ $r = n^{1/3}$
$\mathbf{D}_3$:	Build $\mathbf{D}_2$ on $\mathscr{S}$, $\mathbf{I}$ on $\mathscr{L}$	$\max\left\{\frac{1}{2}\mathrm{dep}(\mathscr{S}), \mathrm{dep}(\mathscr{L})\right\} \geq \frac{1}{3}\mathrm{dep}(\mathscr{O})$	$\tilde{O}\left(2 \cdot (n/r)^{\frac{1}{3}} + r\right) = \tilde{O}\left(n^{\frac{1}{4}}\right)$ $r = n^{1/4}$
• • •		• • •	• • •
$\mathbf{D}_k$:	Build $\mathbf{D}_{k-1}$ on $\mathscr{S}$, $\mathbf{I}$ on $\mathscr{L}$	$\max\left\{\frac{1}{k-1}\mathrm{dep}(\mathscr{S}), \mathrm{dep}(\mathscr{L})\right\} \geq \frac{1}{k}\mathrm{dep}(\mathscr{O})$	$\tilde{O}\left(2 \cdot (n/r)^{\frac{1}{k}} + r\right) = \tilde{O}\left(n^{\frac{1}{k+1}}\right)$ $r = n^{1/(k+1)}$

Approx. Ratio vs Update Time

$\mathbf{D}_1$: exact data structure
$\mathbf{I}$: interval data structure
$\mathscr{L}$: set of long rect.
$\mathscr{S}$: set of short rect.

🤔 Exact data structure: balancing long and short rectangles

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

	Construction	Approx. ratio	Update time
$\mathbf{D}_2$:	Build $\mathbf{D}_1$ on $\mathscr{S}$, $\mathbf{I}$ on $\mathscr{L}$	$\max\{\text{dep}(\mathscr{S}), \text{dep}(\mathscr{L})\} \geq \frac{1}{2}\text{dep}(\mathscr{O})$	$\tilde{O}\left(2 \cdot (n/r)^{\frac{1}{2}} + r\right) = \tilde{O}\left(n^{\frac{1}{3}}\right)$ $r = n^{1/3}$
$\mathbf{D}_3$:	Build $\mathbf{D}_2$ on $\mathscr{S}$, $\mathbf{I}$ on $\mathscr{L}$	$\max\left\{\frac{1}{2}\text{dep}(\mathscr{S}), \text{dep}(\mathscr{L})\right\} \geq \frac{1}{3}\text{dep}(\mathscr{O})$	$\tilde{O}\left(2 \cdot (n/r)^{\frac{1}{3}} + r\right) = \tilde{O}\left(n^{\frac{1}{4}}\right)$ $r = n^{1/4}$
• • •		• • •	• • •
$\mathbf{D}_k$:	Build $\mathbf{D}_{k-1}$ on $\mathscr{S}$, $\mathbf{I}$ on $\mathscr{L}$	$\max\left\{\frac{1}{k-1}\text{dep}(\mathscr{S}), \text{dep}(\mathscr{L})\right\} \geq \frac{1}{k}\text{dep}(\mathscr{O})$	$\tilde{O}\left(2 \cdot (n/r)^{\frac{1}{k}} + r\right) = \tilde{O}\left(n^{\frac{1}{k+1}}\right)$ $r = n^{1/(k+1)}$

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.

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

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

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.
$\mathrm{dep}(p^*, \mathcal{O}) = \mathrm{dep}(\mathcal{P}, \mathcal{O}) = \max_{p \in \mathcal{P}} \mathrm{dep}(p, \mathcal{O})$
and the value of $\mathrm{dep}(p^*, \mathcal{O})$

Companion points

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

Approximate via Discretization

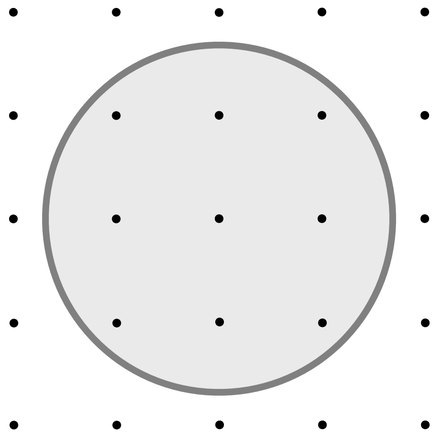
DisMaxDepth

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

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

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

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



Companion points

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

Approximate via Discretization

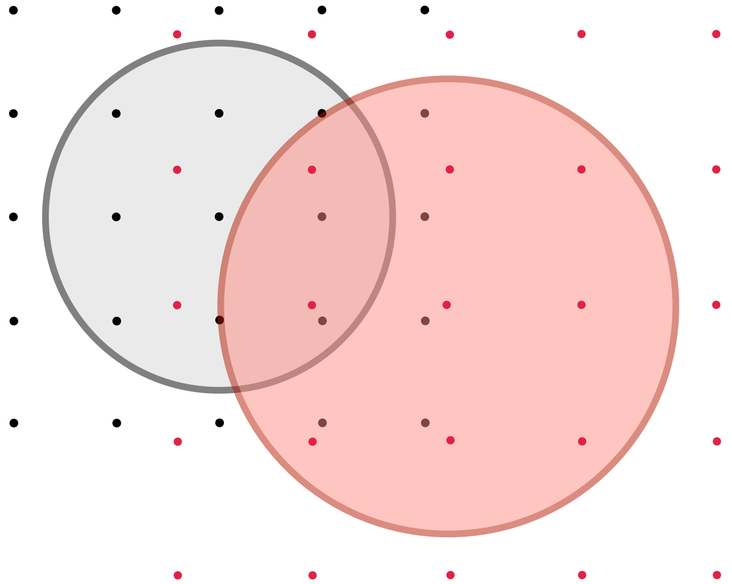
DisMaxDepth

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

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

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

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



Companion points

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

Approximate via Discretization

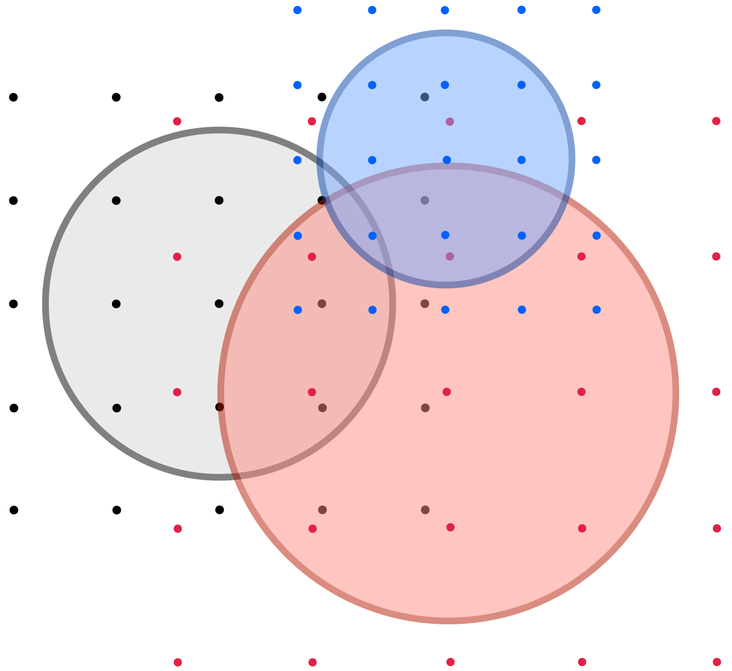
DisMaxDepth

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

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

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

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



Companion points

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

Approximate via Discretization

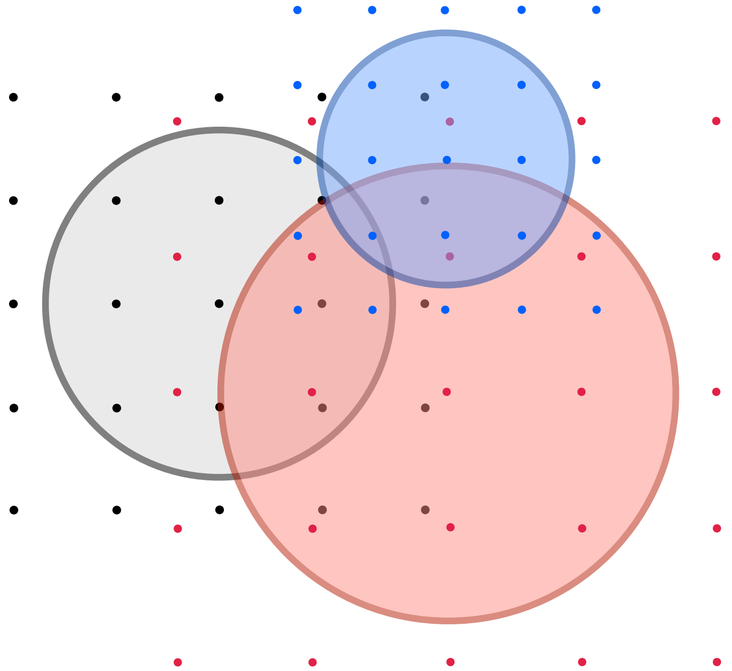
DisMaxDepth

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

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

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

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



Companion points

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

"Well-chosen"

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

Approximation for Disks and Fat Objects

Approximation for Disks and Fat Objects

Approx. for disks

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

Approximation for Disks and Fat Objects

Approx. for disks

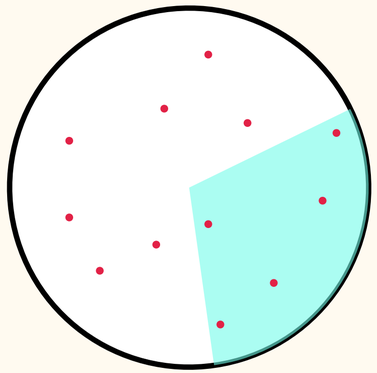
$$\forall \varepsilon > 0, \exists\, \Gamma, \text{s.t. } \mathrm{dep}(\Gamma, \mathcal{O}) \geq \left(\frac{1}{2} - \varepsilon\right) \mathrm{dep}(\mathcal{O})$$

Source of approx. ratio

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

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

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



Approximation for Disks and Fat Objects

Approx. for disks

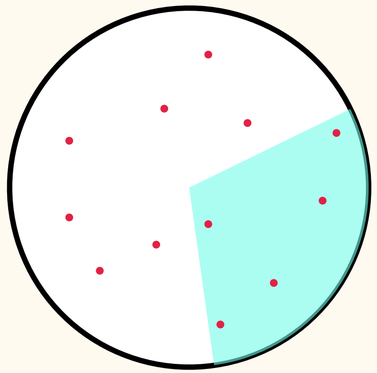
$$\forall \varepsilon > 0, \exists\, \Gamma, \text{s.t. } \mathrm{dep}(\Gamma, \mathcal{O}) \geq \left(\frac{1}{2} - \varepsilon\right) \mathrm{dep}(\mathcal{O})$$

Source of approx. ratio

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

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

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

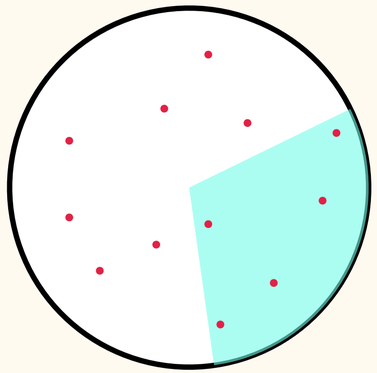
$$\forall \varepsilon > 0, \exists\, \Gamma, \text{s.t. } \mathrm{dep}(\Gamma, \mathcal{O}) \geq \left(\frac{1}{2} - \varepsilon\right) \mathrm{dep}(\mathcal{O})$$

Source of approx. ratio

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

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

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



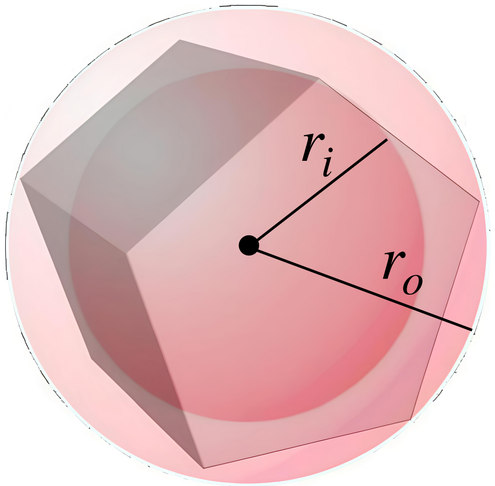
Data structure for disks

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

via dynamic partition tree [Matoušek 1992]

α-fat object:

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



Approximation for Disks and Fat Objects

Approx. for disks

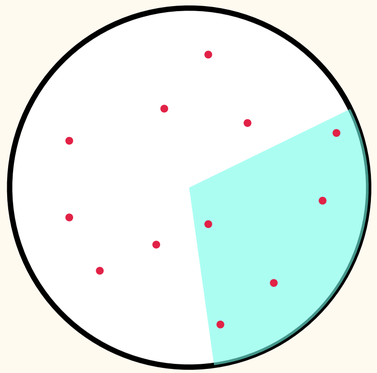
$$\forall \varepsilon > 0, \exists\, \Gamma, \text{s.t. } \mathrm{dep}(\Gamma, \mathcal{O}) \geq \left(\frac{1}{2} - \varepsilon\right) \mathrm{dep}(\mathcal{O})$$

Source of approx. ratio

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

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

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



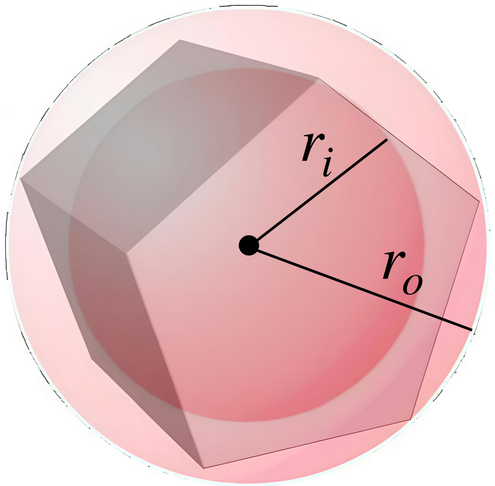
Data structure for disks

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

via dynamic partition tree [Matoušek 1992]

α-fat object:

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



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

Reduce to "balls cover a ball" problem:

If k unit balls cover a ball with radius 2α,

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

Approximation for Disks and Fat Objects

Approx. for disks

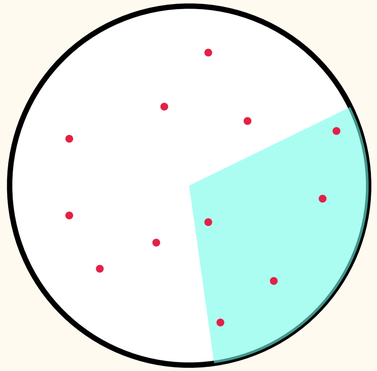
$$\forall \varepsilon > 0, \exists\, \Gamma, \text{s.t. } \mathrm{dep}(\Gamma, \mathcal{O}) \geq \left(\frac{1}{2} - \varepsilon\right) \mathrm{dep}(\mathcal{O})$$

Source of approx. ratio

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

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

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



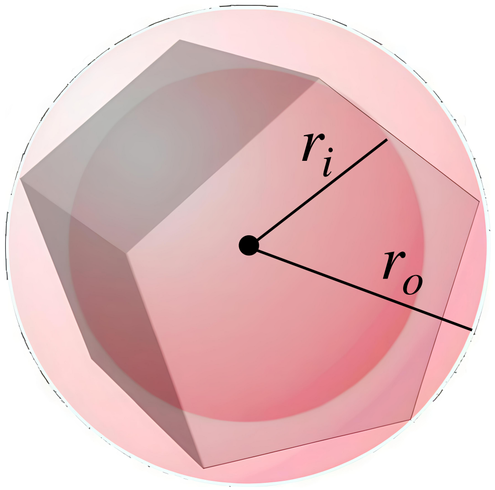
Data structure for disks

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

via dynamic partition tree [Matoušek 1992]

α-fat object:

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



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

Reduce to "balls cover a ball" problem:

If k unit balls cover a ball with radius 2α,

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

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

Approximation for Disks and Fat Objects

Approx. for disks

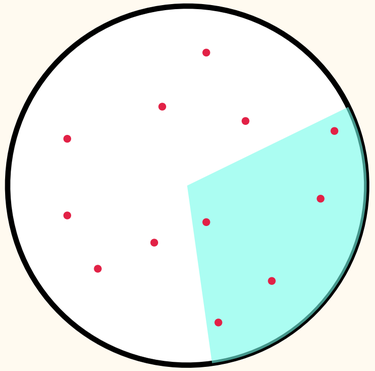
$$\forall \varepsilon > 0, \exists\, \Gamma, \text{s.t. } \mathrm{dep}(\Gamma, \mathcal{O}) \geq \left(\frac{1}{2} - \varepsilon\right) \mathrm{dep}(\mathcal{O})$$

Source of approx. ratio

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

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

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



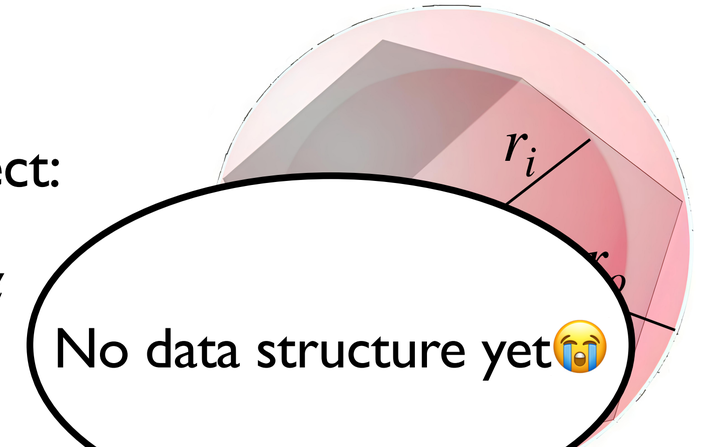
Data structure for disks

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

via dynamic partition tree [Matoušek 1992]

α-fat object:

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



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

Reduce to "balls cover a ball" problem:

If k unit balls cover a ball with radius 2α,

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

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

Thank you!

Any Questions?

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