Dynamic Maximum Depth and Klee's Measure

Jie Xue
New York University Shanghai

joint work with

Sujoy Bhore (IIT Bombay)
Subhash Suri (UCSB)
Xiongxin Yang (UCSB)
Jiumu Zhu (NYU Shanghai)

Background

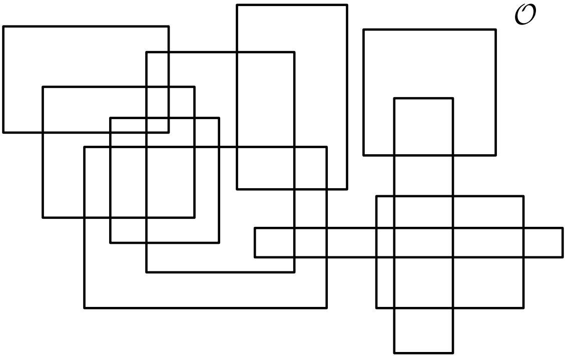
- **Maximum depth**

  $\mathcal{O}$ = a set of geometric objects in $\mathbb{R}^d$

Background

- **Maximum depth**

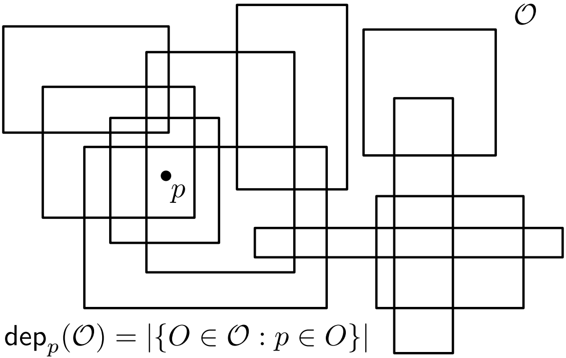
  $\mathcal{O}$ = a set of geometric objects in $\mathbb{R}^d$



Background

- **Maximum depth**

  $\mathcal{O}$ = a set of geometric objects in $\mathbb{R}^d$

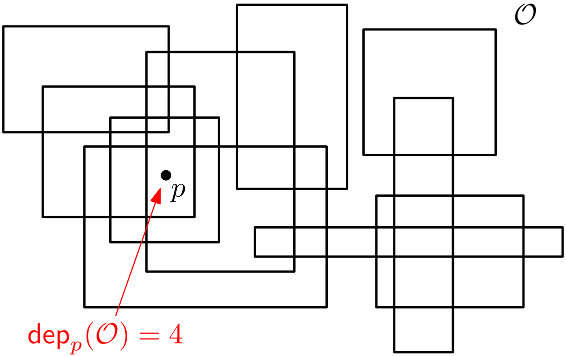


$$\mathrm{dep}_p(\mathcal{O}) = |\{O \in \mathcal{O} : p \in O\}|$$

Background

- **Maximum depth**

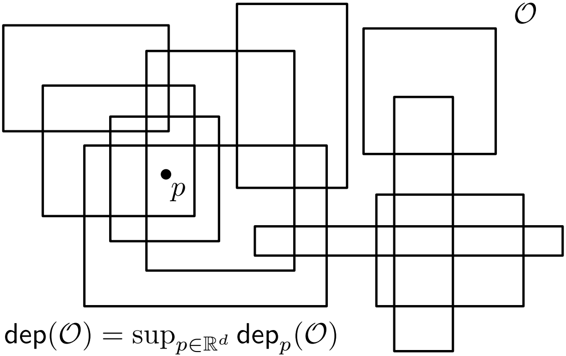
  $\mathcal{O}$ = a set of geometric objects in $\mathbb{R}^d$



Background

- **Maximum depth**

  $\mathcal{O}$ = a set of geometric objects in $\mathbb{R}^d$



$\text{dep}(\mathcal{O}) = \sup_{p \in \mathbb{R}^d} \text{dep}_p(\mathcal{O})$

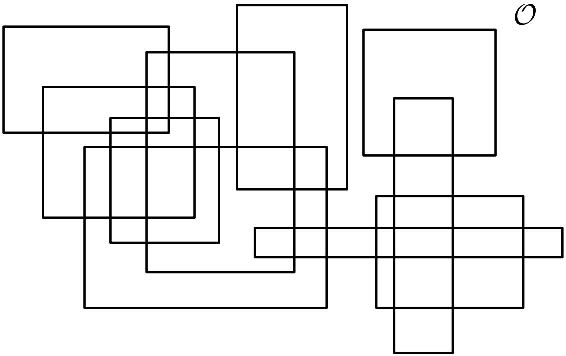
Background

- **Klee's measure**

  $\mathcal{O}$ = a set of geometric objects in $\mathbb{R}^d$

Background

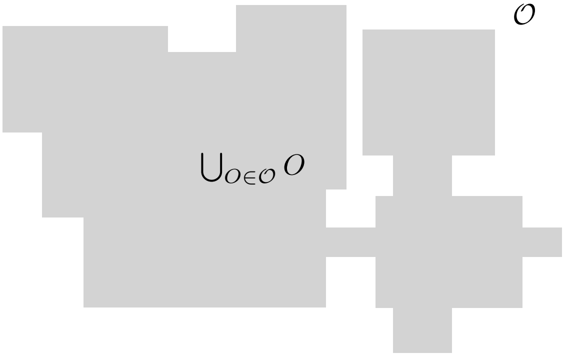
- **Klee's measure**
  $\mathcal{O}$ = a set of geometric objects in $\mathbb{R}^d$



Background

- **Klee's measure**

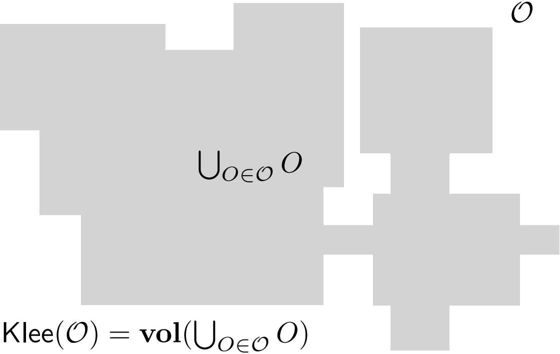
  $\mathcal{O}$ = a set of geometric objects in $\mathbb{R}^d$



Background

- **Klee's measure**

  $\mathcal{O}$ = a set of geometric objects in $\mathbb{R}^d$

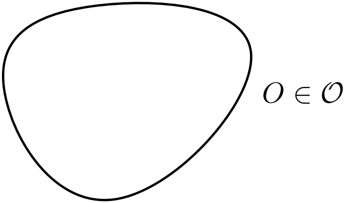


$\mathsf{Klee}(\mathcal{O}) = \mathbf{vol}(\bigcup_{O \in \mathcal{O}} O)$

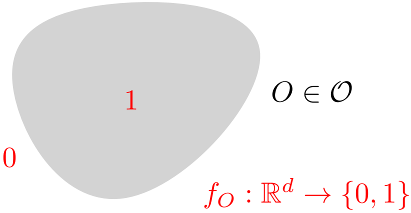
Background

- **Similarity between maximum depth and Klee's measure**

- **Similarity between maximum depth and Klee's measure**

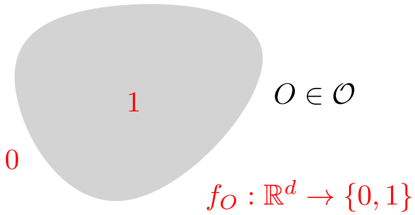


- **Similarity between maximum depth and Klee's measure**



Background

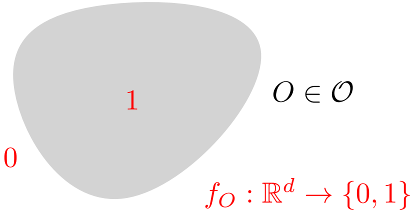
- **Similarity between maximum depth and Klee's measure**



$$\mathsf{dep}(\mathcal{O}) = \sup_{p \in \mathbb{R}^2} \sum_{O \in \mathcal{O}} f_O(p)$$

Background

- **Similarity between maximum depth and Klee's measure**



$$\mathsf{dep}(\mathcal{O}) = \sup_{p \in \mathbb{R}^2} \sum_{O \in \mathcal{O}} f_O(p)$$

$$\mathsf{Klee}(\mathcal{O}) = \int_{p \in \mathbb{R}^2} \max_{O \in \mathcal{O}} f_O(p)$$

Background

- In this talk, we focus on (axis-parallel) boxes in $\mathbb{R}^d$ for a fixed d.

Background

- In this talk, we focus on (axis-parallel) boxes in $\mathbb{R}^d$ for a fixed d.

Theorem (Overmars and Yap FOCS'88; Chan FOCS'13)

The maximum depth of n boxes in $\mathbb{R}^d$ can be computed in $\widetilde{O}(n^{d/2})$ time.

Theorem (Overmars and Yap FOCS'88; Chan FOCS'13)

The Klee's measure of n boxes in $\mathbb{R}^d$ can be computed in $\widetilde{O}(n^{d/2})$ time.

Background

- In this talk, we focus on (axis-parallel) boxes in $\mathbb{R}^d$ for a fixed d.

Theorem (Overmars and Yap FOCS'88; Chan FOCS'13)

The maximum depth of n boxes in $\mathbb{R}^d$ can be computed in $\widetilde{O}(n^{d/2})$ time.

Theorem (Overmars and Yap FOCS'88; Chan FOCS'13)

The Klee's measure of n boxes in $\mathbb{R}^d$ can be computed in $\widetilde{O}(n^{d/2})$ time.

- Here $\widetilde{O}(\cdot)$ hides factors $\log^{O(1)} n$ or $(\log^{O(1)} n)^{-1}$.

Background

- In this talk, we focus on (axis-parallel) boxes in $\mathbb{R}^d$ for a fixed d.

Theorem (Overmars and Yap FOCS'88; Chan FOCS'13)

The maximum depth of n boxes in $\mathbb{R}^d$ can be computed in $\widetilde{O}(n^{d/2})$ time.

Theorem (Overmars and Yap FOCS'88; Chan FOCS'13)

The Klee's measure of n boxes in $\mathbb{R}^d$ can be computed in $\widetilde{O}(n^{d/2})$ time.

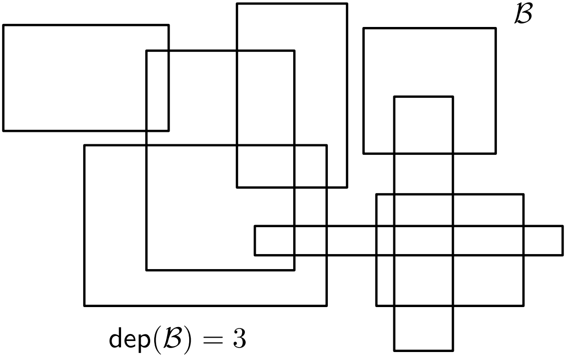
- Here $\widetilde{O}(\cdot)$ hides factors $\log^{O(1)} n$ or $(\log^{O(1)} n)^{-1}$.
- The bound $\widetilde{O}(n^{d/2})$ is believed to be tight up to logarithmic factors.

Background

- We study these problems in the fully dynamic setting, where we are allowed to insert/delete boxes.

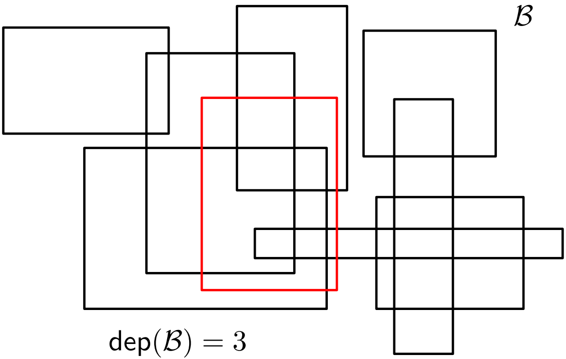
Background

- We study these problems in the fully dynamic setting, where we are allowed to insert/delete boxes.



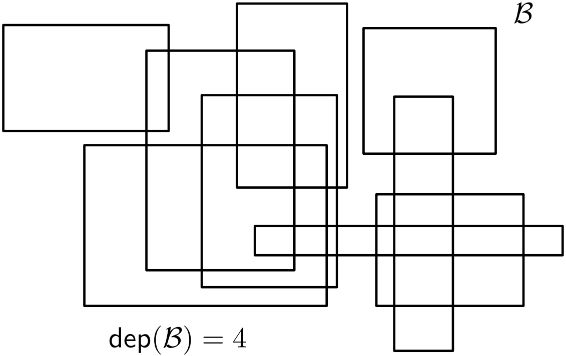
Background

- We study these problems in the fully dynamic setting, where we are allowed to insert/delete boxes.

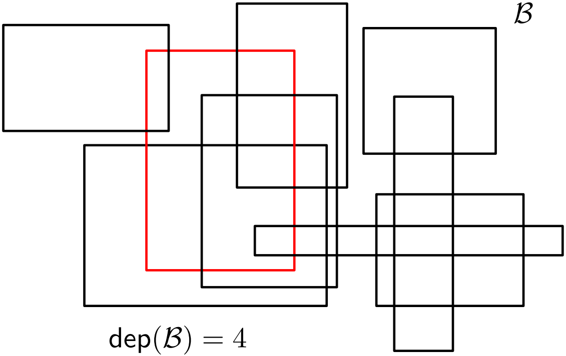


Background

- We study these problems in the fully dynamic setting, where we are allowed to insert/delete boxes.

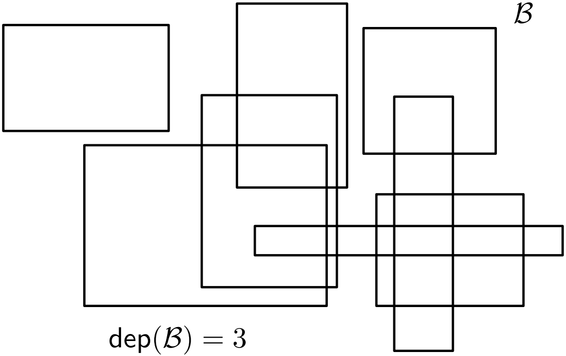


- We study these problems in the fully dynamic setting, where we are allowed to insert/delete boxes.



Background

- We study these problems in the fully dynamic setting, where we are allowed to insert/delete boxes.



Background

- What is the best update time one can hope?

Background

- What is the best update time one can hope?
- **Trivial lower bound:** $\widetilde{\Omega}(n^{d/2-1})$

Background

- What is the best update time one can hope?
- **Trivial lower bound:** $\widetilde{\Omega}(n^{d/2-1})$

$$O(n^{d/2-1-\varepsilon}) \text{ update time} \implies O(n^{d/2-\varepsilon}) \text{ time (static)}$$

Background

- What is the best update time one can hope?
- **Trivial lower bound:** $\widetilde{\Omega}(n^{d/2-1})$

$$O(n^{d/2-1-\varepsilon}) \text{ update time } \implies O(n^{d/2-\varepsilon}) \text{ time (static)}$$

- **Better lower bound:** $\widetilde{\Omega}(n^{(d-1)/2})$

Background

- What is the best update time one can hope?
- **Trivial lower bound:** $\widetilde{\Omega}(n^{d/2-1})$

$$O(n^{d/2-1-\varepsilon}) \text{ update time } \implies O(n^{d/2-\varepsilon}) \text{ time (static)}$$

- **Better lower bound:** $\widetilde{\Omega}(n^{(d-1)/2})$

$$O(n^{(d-1)/2-\varepsilon}) \text{ update time in } \mathbb{R}^d \implies O(n^{(d+1)/2-\varepsilon}) \text{ time in } \mathbb{R}^{d+1}$$

Background

- What is the best update time one can hope?
- **Trivial lower bound:** $\widetilde{\Omega}(n^{d/2-1})$

$$O(n^{d/2-1-\varepsilon}) \text{ update time } \implies O(n^{d/2-\varepsilon}) \text{ time (static)}$$

- **Better lower bound:** $\widetilde{\Omega}(n^{(d-1)/2})$

$$O(n^{(d-1)/2-\varepsilon}) \text{ update time in } \mathbb{R}^d \implies O(n^{(d+1)/2-\varepsilon}) \text{ time in } \mathbb{R}^{d+1}$$

Question

Do there exist dynamic data structures with update time $\widetilde{O}(n^{(d-1)/2})$?

Background

- What is the best update time one can hope?
- **Trivial lower bound:** $\widetilde{\Omega}(n^{d/2-1})$

$$O(n^{d/2-1-\varepsilon}) \text{ update time } \implies O(n^{d/2-\varepsilon}) \text{ time (static)}$$

- **Better lower bound:** $\widetilde{\Omega}(n^{(d-1)/2})$

$$O(n^{(d-1)/2-\varepsilon}) \text{ update time in } \mathbb{R}^d \implies O(n^{(d+1)/2-\varepsilon}) \text{ time in } \mathbb{R}^{d+1}$$

Question

Do there exist dynamic data structures with update time $\widetilde{O}(n^{(d-1)/2})$?

- For $d = 1$, one can achieve $O(\log n)$ update time using interval trees.

Our results

- We answer this question affirmatively for both problems for all fixed d.

Our results

- We answer this question affirmatively for both problems for all fixed d.

Theorem

There exists a dynamic data structure for maximum depth of boxes in $\mathbb{R}^d$ with $\widetilde{O}(n^{(d-1)/2})$ amortized update time.

Our results

- We answer this question affirmatively for both problems for all fixed d.

Theorem

There exists a dynamic data structure for maximum depth of boxes in $\mathbb{R}^d$ with $\widetilde{O}(n^{(d-1)/2})$ amortized update time.

Theorem

There exists a dynamic data structure for Klee's measure of boxes in $\mathbb{R}^d$ with $\widetilde{O}(n^{(d-1)/2})$ amortized update time.

Warmup: Rectangles in $\mathbb{R}^2$

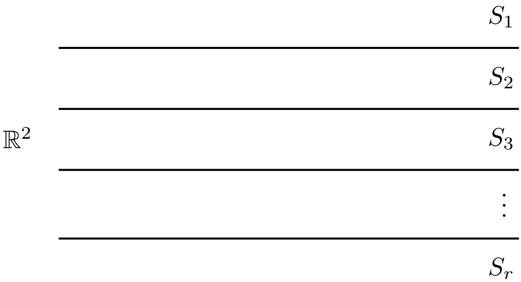
- $\mathcal{B} =$ a dynamic set of rectangles in $\mathbb{R}^2$

Warmup: Rectangles in $\mathbb{R}^2$

- $\mathcal{B}$ = a dynamic set of rectangles in $\mathbb{R}^2$
- Let's consider how to maintain $\text{dep}(\mathcal{B})$ in $\widetilde{O}(\sqrt{n})$ time.

Warmup: Rectangles in $\mathbb{R}^2$

- $\mathcal{B}$ = a dynamic set of rectangles in $\mathbb{R}^2$
- Let's consider how to maintain $\mathrm{dep}(\mathcal{B})$ in $\widetilde{O}(\sqrt{n})$ time.
- **Space partitioning**



Warmup: Rectangles in $\mathbb{R}^2$

- $\mathcal{B}$ = a dynamic set of rectangles in $\mathbb{R}^2$
- Let's consider how to maintain dep($\mathcal{B}$) in $\widetilde{O}(\sqrt{n})$ time.
- **Space partitioning**

$O(n/r)$ rectangle corners S_1

$O(n/r)$ rectangle corners S_2

$\mathbb{R}^2$ $O(n/r)$ rectangle corners S_3

$\vdots$

$O(n/r)$ rectangle corners S_r

Warmup: Rectangles in $\mathbb{R}^2$

- For each strip S_i, maintain $\mathrm{dep}_{S_i}(\mathcal{B})$ individually.

Warmup: Rectangles in $\mathbb{R}^2$

- For each strip S_i, maintain $\text{dep}_{S_i}(\mathcal{B})$ individually.
- Note that S_i can intersect $\Omega(n)$ rectangles in $\mathcal{B}$.

Warmup: Rectangles in $\mathbb{R}^2$

- For each strip S_i, maintain $\text{dep}_{S_i}(\mathcal{B})$ individually.
- Note that S_i can intersect $\Omega(n)$ rectangles in $\mathcal{B}$.
- **Intuition:** all but $O(n/r)$ rectangles intersecting S_i are good

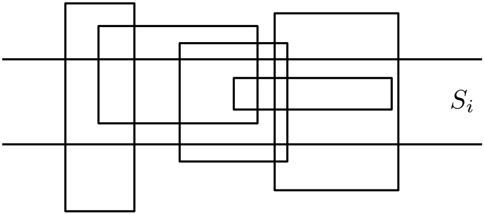
Warmup: Rectangles in $\mathbb{R}^2$

- For each strip S_i, maintain $\text{dep}_{S_i}(\mathcal{B})$ individually.
- Note that S_i can intersect $\Omega(n)$ rectangles in $\mathcal{B}$.
- **Intuition:** all but $O(n/r)$ rectangles intersecting S_i are good

  (No corners in S_i)

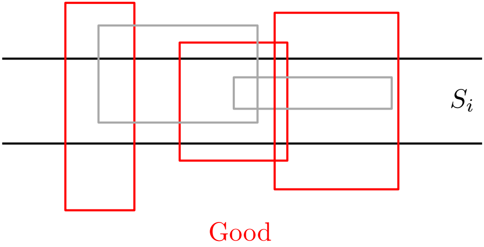
Warmup: Rectangles in $\mathbb{R}^2$

- For each strip S_i, maintain $\mathrm{dep}_{S_i}(\mathcal{B})$ individually.
- Note that S_i can intersect $\Omega(n)$ rectangles in $\mathcal{B}$.
- **Intuition:** all but $O(n/r)$ rectangles intersecting S_i are good (No corners in S_i)



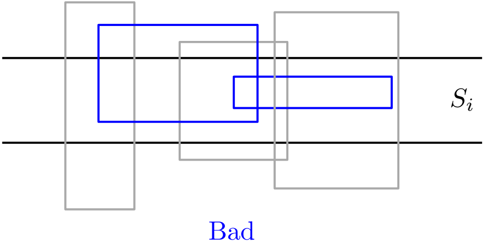
Warmup: Rectangles in $\mathbb{R}^2$

- For each strip S_i, maintain $\mathrm{dep}_{S_i}(\mathcal{B})$ individually.
- Note that S_i can intersect $\Omega(n)$ rectangles in $\mathcal{B}$.
- **Intuition:** all but $O(n/r)$ rectangles intersecting S_i are good (No corners in S_i)



Warmup: Rectangles in $\mathbb{R}^2$

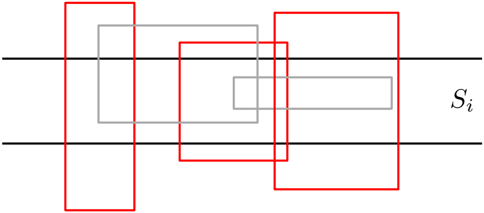
- For each strip S_i, maintain $\text{dep}_{S_i}(\mathcal{B})$ individually.
- Note that S_i can intersect $\Omega(n)$ rectangles in $\mathcal{B}$.
- **Intuition:** all but $O(n/r)$ rectangles intersecting S_i are good (No corners in S_i)



Warmup: Rectangles in $\mathbb{R}^2$

- For each strip S_i, maintain $\mathrm{dep}_{S_i}(\mathcal{B})$ individually.
- Note that S_i can intersect $\Omega(n)$ rectangles in $\mathcal{B}$.
- **Intuition:** all but $O(n/r)$ rectangles intersecting S_i are good

  (No corners in S_i)



Good rectangles can be viewed as intervals

Warmup: Rectangles in $\mathbb{R}^2$

- $\mathcal{B}_i \subseteq \mathcal{B}$ consists of the rectangles intersecting S_i.

Warmup: Rectangles in $\mathbb{R}^2$

- $\mathcal{B}_i \subseteq \mathcal{B}$ consists of the rectangles intersecting S_i.
- **Main idea:** map the rectangles in $\mathcal{B}_i$ to a set $\mathcal{I}_i$ of $O(n)$ (weighted) intervals in $\mathbb{R}^1$ such that $\mathrm{dep}_{S_i}(\mathcal{B}_i) = \mathrm{dep}(\mathcal{I}_i)$

Warmup: Rectangles in $\mathbb{R}^2$

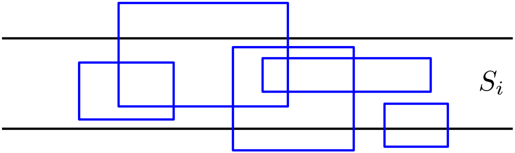
- $\mathcal{B}_i \subseteq \mathcal{B}$ consists of the rectangles intersecting S_i.
- **Main idea:** map the rectangles in $\mathcal{B}_i$ to a set $\mathcal{I}_i$ of $O(n)$ (weighted) intervals in $\mathbb{R}^1$ such that $\mathrm{dep}_{S_i}(\mathcal{B}_i) = \mathrm{dep}(\mathcal{I}_i)$
- $B \in \mathcal{B}_i^{\mathrm{good}} \mapsto \mathbf{proj}_x(B) \in \mathcal{I}_i$ with weight 1

Warmup: Rectangles in $\mathbb{R}^2$

- $\mathcal{B}_i \subseteq \mathcal{B}$ consists of the rectangles intersecting S_i.
- **Main idea:** map the rectangles in $\mathcal{B}_i$ to a set $\mathcal{I}_i$ of $O(n)$ (weighted) intervals in $\mathbb{R}^1$ such that $\mathrm{dep}_{S_i}(\mathcal{B}_i) = \mathrm{dep}(\mathcal{I}_i)$
- $B \in \mathcal{B}_i^{\text{good}} \mapsto \mathbf{proj}_x(B) \in \mathcal{I}_i$ with weight 1
- How do we handle the bad rectangles?

Warmup: Rectangles in $\mathbb{R}^2$

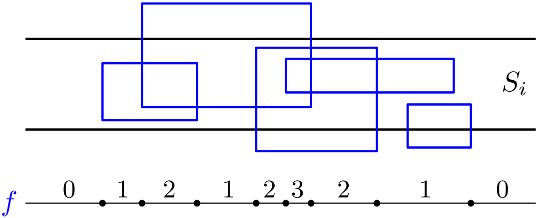
- $\mathcal{B}_i \subseteq \mathcal{B}$ consists of the rectangles intersecting S_i.
- **Main idea:** map the rectangles in $\mathcal{B}_i$ to a set $\mathcal{I}_i$ of $O(n)$ (weighted) intervals in $\mathbb{R}^1$ such that $\mathsf{dep}_{S_i}(\mathcal{B}_i) = \mathsf{dep}(\mathcal{I}_i)$
- $B \in \mathcal{B}_i^{\mathsf{good}} \mapsto \mathbf{proj}_x(B) \in \mathcal{I}_i$ with weight 1
- How do we handle the bad rectangles?



$f : \mathbb{R} \to \mathbb{N}_0 \qquad f(x) = \mathsf{dep}_{\{x\}\times\mathbb{R}}(\mathcal{B}_i^{\mathrm{bad}})$

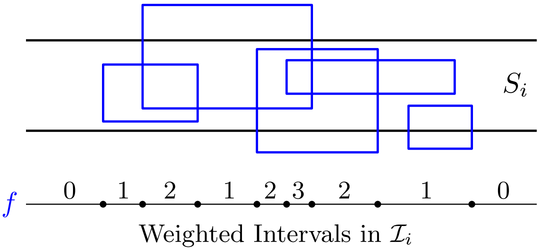
Warmup: Rectangles in $\mathbb{R}^2$

- $\mathcal{B}_i \subseteq \mathcal{B}$ consists of the rectangles intersecting S_i.
- **Main idea:** map the rectangles in $\mathcal{B}_i$ to a set $\mathcal{I}_i$ of $O(n)$ (weighted) intervals in $\mathbb{R}^1$ such that $\mathrm{dep}_{S_i}(\mathcal{B}_i) = \mathrm{dep}(\mathcal{I}_i)$
- $B \in \mathcal{B}_i^{\text{good}} \mapsto \mathbf{proj}_x(B) \in \mathcal{I}_i$ with weight 1
- How do we handle the bad rectangles?



Warmup: Rectangles in $\mathbb{R}^2$

- $\mathcal{B}_i \subseteq \mathcal{B}$ consists of the rectangles intersecting S_i.
- **Main idea:** map the rectangles in $\mathcal{B}_i$ to a set $\mathcal{I}_i$ of $O(n)$ (weighted) intervals in $\mathbb{R}^1$ such that $\text{dep}_{S_i}(\mathcal{B}_i) = \text{dep}(\mathcal{I}_i)$
- $B \in \mathcal{B}_i^{\text{good}} \mapsto \mathbf{proj}_x(B) \in \mathcal{I}_i$ with weight 1
- How do we handle the bad rectangles?



Warmup: Rectangles in $\mathbb{R}^2$

- We maintain $\mathcal{I}_i$ in a dynamic maximum depth data structure $\mathcal{D}_i$ for intervals with $O(\log n)$ update time.

Warmup: Rectangles in $\mathbb{R}^2$

- We maintain $\mathcal{I}_i$ in a dynamic maximum depth data structure $\mathcal{D}_i$ for intervals with $O(\log n)$ update time.
- When inserting or deleting a rectangle B...

Warmup: Rectangles in $\mathbb{R}^2$

- We maintain $\mathcal{I}_i$ in a dynamic maximum depth data structure $\mathcal{D}_i$ for intervals with $O(\log n)$ update time.
- When inserting or deleting a rectangle B...

Is B good w.r.t. S_i?

Warmup: Rectangles in $\mathbb{R}^2$

- We maintain $\mathcal{I}_i$ in a dynamic maximum depth data structure $\mathcal{D}_i$ for intervals with $O(\log n)$ update time.
- When inserting or deleting a rectangle B...

Is B good w.r.t. S_i? $\xrightarrow{\text{Yes}}$ Insert/delete $\mathbf{proj}_x(B)$ to $\mathcal{I}_i$

$O(\log n)$

Warmup: Rectangles in $\mathbb{R}^2$

- We maintain $\mathcal{I}_i$ in a dynamic maximum depth data structure $\mathcal{D}_i$ for intervals with $O(\log n)$ update time.
- When inserting or deleting a rectangle B...

Is B good w.r.t. S_i?

Yes → Insert/delete $\mathbf{proj}_x(B)$ to $\mathcal{I}_i$

$O(\log n)$

No →

Warmup: Rectangles in $\mathbb{R}^2$

- We maintain $\mathcal{I}_i$ in a dynamic maximum depth data structure $\mathcal{D}_i$ for intervals with $O(\log n)$ update time.
- When inserting or deleting a rectangle B...

Is B good w.r.t. S_i?

Yes → Insert/delete $\mathbf{proj}_x(B)$ to $\mathcal{I}_i$

$O(\log n)$

No → Recompute the dep function
Delete intervals of the old func.
Insert intervals of the new func.

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

Warmup: Rectangles in $\mathbb{R}^2$

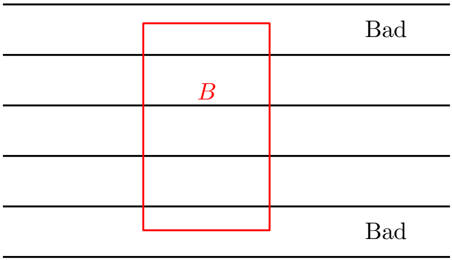
- **Global update time**

Warmup: Rectangles in $\mathbb{R}^2$

- **Global update time**
- B is bad for at most two S_i's and good for $O(r)$ S_i's.

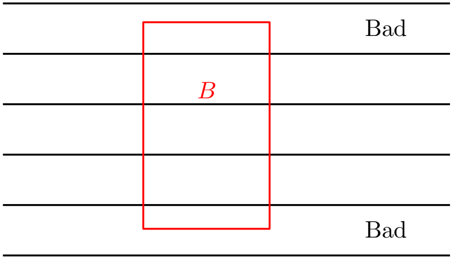
Warmup: Rectangles in $\mathbb{R}^2$

- **Global update time**
- B is bad for at most two S_i's and good for $O(r)$ S_i's.



Warmup: Rectangles in $\mathbb{R}^2$

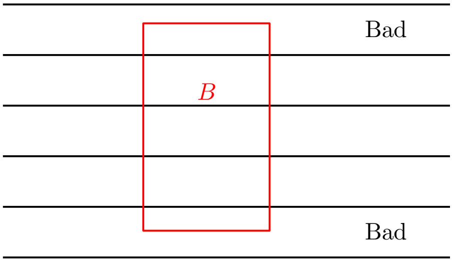
- **Global update time**
- B is bad for at most two S_i's and good for $O(r)$ S_i's.



- $T(n, r) = O((n/r)\log n) + O(r \log n)$

Warmup: Rectangles in $\mathbb{R}^2$

- **Global update time**
- B is bad for at most two S_i's and good for $O(r)$ S_i's.



- $T(n,r) = O((n/r)\log n) + O(r \log n)$
- Setting $r = \sqrt{n}$ gives us $O(\sqrt{n}\log n)$ update time.

General case

- Unfortunately, the result doesn't directly generalize to $d \geq 3$.

General case

- Unfortunately, the result doesn't directly generalize to $d \geq 3$.
- The data structure easily generalizes (but with worse update time).

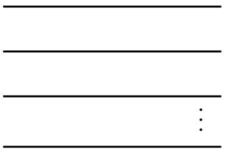
General case

- Unfortunately, the result doesn't directly generalize to $d \geq 3$.
- The data structure easily generalizes (but with worse update time).
- **Some natural attempts**

General case

- Unfortunately, the result doesn't directly generalize to $d \geq 3$.
- The data structure easily generalizes (but with worse update time).
- **Some natural attempts**

Attempt 1

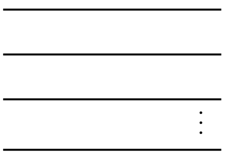


Partition in the first dim

General case

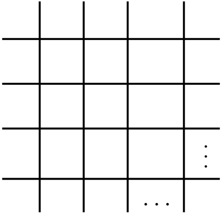
- Unfortunately, the result doesn't directly generalize to $d \geq 3$.
- The data structure easily generalizes (but with worse update time).
- **Some natural attempts**

Attempt 1



Partition in the first dim

Attempt 2



Partition in the first $d-1$ dim

General case

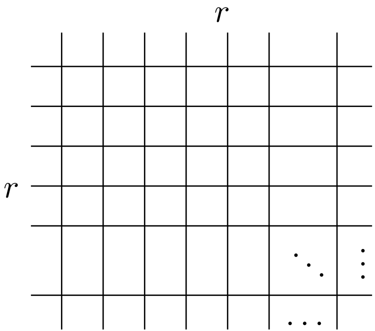
- Attempt 1 seems intrinsically unfeasible.

General case

- Attempt 1 seems intrinsically unfeasible.
- Our result for $d \geq 3$ works in the direction of Attempt 2.

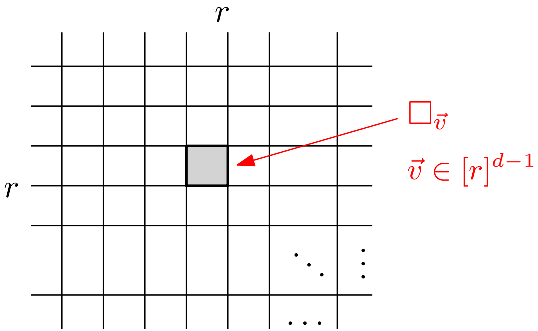
General case

- Attempt 1 seems intrinsically unfeasible.
- Our result for $d \geq 3$ works in the direction of Attempt 2.



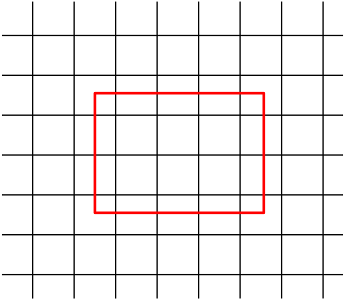
General case

- Attempt 1 seems intrinsically unfeasible.
- Our result for $d \geq 3$ works in the direction of Attempt 2.



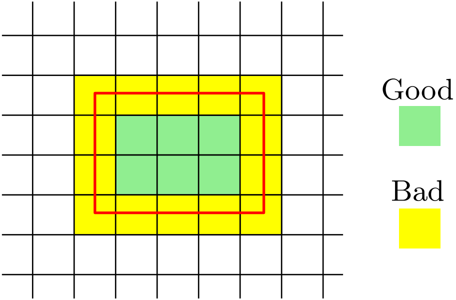
General case

- Attempt 1 seems intrinsically unfeasible.
- Our result for $d \geq 3$ works in the direction of Attempt 2.



General case

- Attempt 1 seems intrinsically unfeasible.
- Our result for $d \geq 3$ works in the direction of Attempt 2.



General case

- When inserting/deleting a box B...

General case

- When inserting/deleting a box B...
- There are $O(r^{d-1})$ cells $\square_{\vec{v}}$ for which B is good.

General case

- When inserting/deleting a box B...
- There are $O(r^{d-1})$ cells $\square_{\vec{v}}$ for which B is good.
- Update for each such cell takes $O(\log n)$ time.

General case

- When inserting/deleting a box B...
- There are $O(r^{d-1})$ cells $\Box_{\vec{v}}$ for which B is good.
- Update for each such cell takes $O(\log n)$ time.
  $\Rightarrow \widetilde{O}(r^{d-1})$ time in total

General case

- When inserting/deleting a box B...
- There are $O(r^{d-1})$ cells $\square_{\vec{v}}$ for which B is good.
- Update for each such cell takes $O(\log n)$ time.
  $\Rightarrow \widetilde{O}(r^{d-1})$ time in total
- There are $O(r^{d-2})$ cells $\square_{\vec{v}}$ for which B is bad.

General case

- When inserting/deleting a box B...
- There are $O(r^{d-1})$ cells $\square_{\vec{v}}$ for which B is good.
- Update for each such cell takes $O(\log n)$ time.
  $\Rightarrow \widetilde{O}(r^{d-1})$ time in total
- There are $O(r^{d-2})$ cells $\square_{\vec{v}}$ for which B is bad.
- Update for each such cell takes $\widetilde{O}((n/r)^{d/2})$ time.
  - Recomputing the depth function $\rightarrow \widetilde{O}((n/r)^{d/2})$ time
  - Deleting/inserting the intervals $\rightarrow \widetilde{O}(n/r)$ time

General case

- When inserting/deleting a box B...
- There are $O(r^{d-1})$ cells $\square_{\vec{v}}$ for which B is good.
- Update for each such cell takes $O(\log n)$ time.
  $\Rightarrow \widetilde{O}(r^{d-1})$ time in total
- There are $O(r^{d-2})$ cells $\square_{\vec{v}}$ for which B is bad.
- Update for each such cell takes $\widetilde{O}((n/r)^{d/2})$ time.
  - Recomputing the depth function $\rightarrow \widetilde{O}((n/r)^{d/2})$ time
  - Deleting/inserting the intervals $\rightarrow \widetilde{O}(n/r)$ time

  $\Rightarrow \widetilde{O}(n^{d/2} r^{d/2-2})$ time in total (can't afford!)

General case

- When inserting/deleting a box B...
- There are $O(r^{d-1})$ cells $\square_{\vec{v}}$ for which B is good.
- Update for each such cell takes $O(\log n)$ time.
  $\Rightarrow \widetilde{O}(r^{d-1})$ time in total
- There are $O(r^{d-2})$ cells $\square_{\vec{v}}$ for which B is bad.
- Update for each such cell takes $\widetilde{O}((n/r)^{d/2})$ time.
  - Recomputing the depth function $\rightarrow \widetilde{O}((n/r)^{d/2})$ time $\leftarrow$ Bottleneck
  - Deleting/inserting the intervals $\rightarrow \widetilde{O}(n/r)$ time

  $\Rightarrow \widetilde{O}(n^{d/2} r^{d/2-2})$ time in total (can't afford!)

General case

- If we look at a single cell $\square_{\vec{v}}$, there can be $O(n/r)$ bad boxes w.r.t. $\square_{\vec{v}}$, which are totally unstructured.

General case

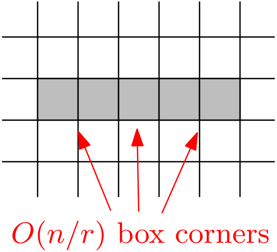
- If we look at a single cell $\square_{\vec{v}}$, there can be $O(n/r)$ bad boxes w.r.t. $\square_{\vec{v}}$, which are totally unstructured.
- We have to pay $\widetilde{O}((n/r)^{d/2})$ time to compute the depth function for a single cell $\square_{\vec{v}}$, in the worst case.

General case

- If we look at a single cell $\square_{\vec{v}}$, there can be $O(n/r)$ bad boxes w.r.t. $\square_{\vec{v}}$, which are totally unstructured.
- We have to pay $\widetilde{O}((n/r)^{d/2})$ time to compute the depth function for a single cell $\square_{\vec{v}}$, in the worst case.
- **Main insight:** globally consider all cells $\square_{\vec{v}}$

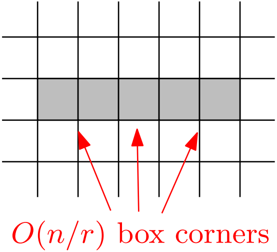
General case

- If we look at a single cell $\square_{\vec{v}}$, there can be $O(n/r)$ bad boxes w.r.t. $\square_{\vec{v}}$, which are totally unstructured.
- We have to pay $\widetilde{O}((n/r)^{d/2})$ time to compute the depth function for a single cell $\square_{\vec{v}}$, in the worst case.
- **Main insight:** globally consider all cells $\square_{\vec{v}}$



General case

- If we look at a single cell $\square_{\vec{v}}$, there can be $O(n/r)$ bad boxes w.r.t. $\square_{\vec{v}}$, which are totally unstructured.
- We have to pay $\widetilde{O}((n/r)^{d/2})$ time to compute the depth function for a single cell $\square_{\vec{v}}$, in the worst case.
- **Main insight:** globally consider all cells $\square_{\vec{v}}$



Most bad boxes are only bounded in one dimension

General case

- Computing the depth function in $\square_{\vec{v}}$ can be reduced to the offline dynamic problem in $\mathbb{R}^{d-1}$, i.e., the updates are known beforehand.

General case

- Computing the depth function in $\square_{\vec{v}}$ can be reduced to the offline dynamic problem in $\mathbb{R}^{d-1}$, i.e., the updates are known beforehand.

Offline Dynamic Maximum Depth

Input: An initially empty set $\mathcal{B}'$ of boxes in $\mathbb{R}^{d-1}$ and a sequence of n updates to $\mathcal{B}'$, each of which is of one of the following two types:

- **Insertion:** Insert a new box to $\mathcal{B}'$
- **Deletion:** Delete an existing box from $\mathcal{B}'$

General case

- Computing the depth function in $\square_{\vec{v}}$ can be reduced to the offline dynamic problem in $\mathbb{R}^{d-1}$, i.e., the updates are known beforehand.

Offline Dynamic Maximum Depth

Input: An initially empty set $\mathcal{B}'$ of boxes in $\mathbb{R}^{d-1}$ and a sequence of n updates to $\mathcal{B}'$, each of which is of one of the following two types:

- **Insertion:** Insert a new box to $\mathcal{B}'$
- **Deletion:** Delete an existing box from $\mathcal{B}'$

Output: A sequence $(\delta_1, \ldots, \delta_n)$, where $\delta_i \in \mathbb{N}_0$ is the maximum depth of $\mathcal{B}'$ after the i-th update

General case

- Computing the depth function in $\square_{\vec{v}}$ can be reduced to the offline dynamic problem in $\mathbb{R}^{d-1}$, i.e., the updates are known beforehand.

Offline Dynamic Maximum Depth

Input: An initially empty set $\mathcal{B}'$ of boxes in $\mathbb{R}^{d-1}$ and a sequence of n updates to $\mathcal{B}'$, each of which is of one of the following two types:

- **Insertion:** Insert a new box to $\mathcal{B}'$
- **Deletion:** Delete an existing box from $\mathcal{B}'$

Output: A sequence $(\delta_1, \ldots, \delta_n)$, where $\delta_i \in \mathbb{N}_0$ is the maximum depth of $\mathcal{B}'$ after the i-th update

- We design an offline algorithm whose total update time depends on not only the number of updates but also the structure of the boxes.

General case

- A box B is trivial in dimension $j \in [d-1]$ if $\mathbf{proj}_j(B) = (-\infty, \infty)$.

General case

- A box B is trivial in dimension $j \in [d-1]$ if $\mathbf{proj}_j(B) = (-\infty, \infty)$.
- For a permutation σ of $[d-1]$, $i \in [d-1]$, and a box B in $\mathbb{R}^{d-1}$, we define a number $\phi_\sigma(B, i) \in \{0, 1\}$ as follows:

General case

- A box B is trivial in dimension $j \in [d-1]$ if $\mathbf{proj}_j(B) = (-\infty, \infty)$.
- For a permutation σ of $[d-1]$, $i \in [d-1]$, and a box B in $\mathbb{R}^{d-1}$, we define a number $\phi_\sigma(B, i) \in \{0, 1\}$ as follows:

$$\phi_\sigma(B,i) = \begin{cases} 1 & B \text{ is nontrivial in dim } i \text{ and} \\ & B \text{ is nontrivial in dim } j \text{ for some } j <_\sigma i \\ 0 & \text{Otherwise} \end{cases}$$

General case

- For a set $\mathcal{B}^*$ of boxes, define $\Phi_\sigma(\mathcal{B}^*) = \prod_{i=1}^{d-1}(1 + \sum_{B \in \mathcal{B}^*} \phi_\sigma(B, i))$.

General case

- For a set $\mathcal{B}^*$ of boxes, define $\Phi_\sigma(\mathcal{B}^*) = \prod_{i=1}^{d-1}(1 + \sum_{B \in \mathcal{B}^*} \phi_\sigma(B, i))$.

Lemma (offline algorithm)

Let σ be a permutation of $[d-1]$. Then Offline Dynamic Maximum Depth *in $\mathbb{R}^{d-1}$ can be solved in $O(n \log n \cdot \Phi_\sigma(\mathcal{B}^*))$ time, where $\mathcal{B}^*$ is the set of all boxes involved in the update sequence.*

General case

- For a set $\mathcal{B}^*$ of boxes, define $\Phi_\sigma(\mathcal{B}^*) = \prod_{i=1}^{d-1}(1 + \sum_{B \in \mathcal{B}^*} \phi_\sigma(B, i))$.

Lemma (offline algorithm)

Let σ be a permutation of $[d-1]$. Then Offline Dynamic Maximum Depth *in $\mathbb{R}^{d-1}$ can be solved in $O(n \log n \cdot \Phi_\sigma(\mathcal{B}^*))$ time, where $\mathcal{B}^*$ is the set of all boxes involved in the update sequence.*

- For each $i \in [d-1]$, consider the boxes $B \in \mathcal{B}^*$ with $\phi_\sigma(B, i) = 1$.

General case

- For a set $\mathcal{B}^*$ of boxes, define $\Phi_\sigma(\mathcal{B}^*) = \prod_{i=1}^{d-1}(1 + \sum_{B\in\mathcal{B}^*} \phi_\sigma(B, i))$.

Lemma (offline algorithm)

Let σ be a permutation of $[d-1]$. Then Offline Dynamic Maximum Depth *in $\mathbb{R}^{d-1}$ can be solved in $O(n \log n \cdot \Phi_\sigma(\mathcal{B}^*))$ time, where $\mathcal{B}^*$ is the set of all boxes involved in the update sequence.*

- For each $i \in [d-1]$, consider the boxes $B \in \mathcal{B}^*$ with $\phi_\sigma(B, i) = 1$.
- Partition $\mathbb{R}^{d-1}$ in dimension i along the i-facets of these boxes.

General case

- For a set $\mathcal{B}^*$ of boxes, define $\Phi_\sigma(\mathcal{B}^*) = \prod_{i=1}^{d-1}(1 + \sum_{B \in \mathcal{B}^*} \phi_\sigma(B, i))$.

Lemma (offline algorithm)

Let σ be a permutation of $[d-1]$. Then Offline Dynamic Maximum Depth *in $\mathbb{R}^{d-1}$ can be solved in $O(n \log n \cdot \Phi_\sigma(\mathcal{B}^*))$ time, where $\mathcal{B}^*$ is the set of all boxes involved in the update sequence.*

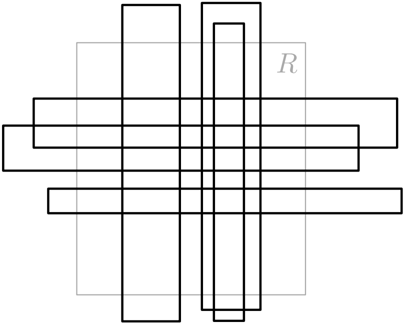
- For each $i \in [d-1]$, consider the boxes $B \in \mathcal{B}^*$ with $\phi_\sigma(B, i) = 1$.
- Partition $\mathbb{R}^{d-1}$ in dimension i along the i-facets of these boxes.
- This partitions $\mathbb{R}^{d-1}$ into $O(\Phi_\sigma(\mathcal{B}^*))$ regions. Inside each region, all boxes in $\mathcal{B}^*$ are trivial in all but one dimensions.

General case

- We can maintain $\mathrm{dep}_R(\mathcal{B}')$ for each region R in $O(\log n)$ time.

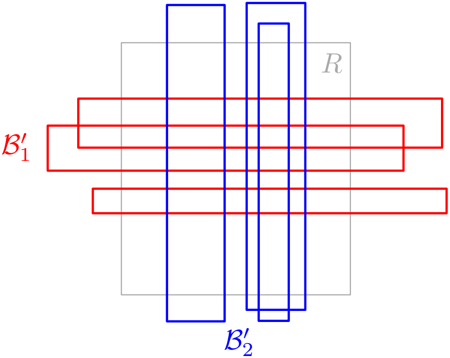
General case

- We can maintain $\text{dep}_R(\mathcal{B}')$ for each region R in $O(\log n)$ time.



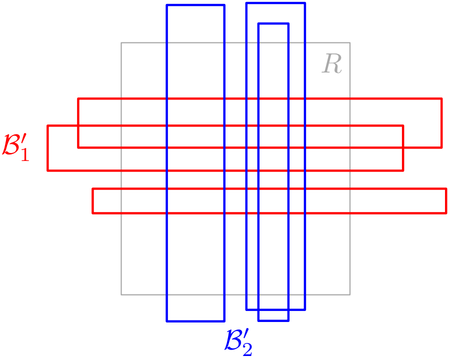
General case

- We can maintain $\mathrm{dep}_R(\mathcal{B}')$ for each region R in $O(\log n)$ time.



General case

- We can maintain $\text{dep}_R(\mathcal{B}')$ for each region R in $O(\log n)$ time.



$$\text{dep}_R(\mathcal{B}'_1 \cup \mathcal{B}'_2) = \text{dep}_R(\mathcal{B}'_1) + \text{dep}_R(\mathcal{B}'_2)$$

General case

- **Back to the dynamic problem**

General case

- **Back to the dynamic problem**
- When inserting/deleting a box B...

General case

- **Back to the dynamic problem**
- When inserting/deleting a box B...
  - For each $\square_{\vec{v}}$ where B is good, update in $O(\log n)$ time.

General case

- **Back to the dynamic problem**
- When inserting/deleting a box B...
  - For each $\square_{\vec{v}}$ where B is good, update in $O(\log n)$ time.
  - For each $\square_{\vec{v}}$ where B is bad, apply our offline algorithm with the best ordering $\sigma \in \Sigma_{[d-1]}$ to compute the depth function

General case

- **Back to the dynamic problem**
- When inserting/deleting a box B...
  - For each $\square_{\vec{v}}$ where B is good, update in $O(\log n)$ time.
  - For each $\square_{\vec{v}}$ where B is bad, apply our offline algorithm with the best ordering $\sigma \in \Sigma_{[d-1]}$ to compute the depth function in time

$$O\left(\min_{\sigma \in \Sigma_{[d-1]}} (n/r)\log(n/r) \cdot \Phi_\sigma(\mathbf{proj}_{[d-1]}(\mathcal{B}_{\vec{v}}^{\text{bad}}))\right)$$

where $\mathcal{B}_{\vec{v}}^{\text{bad}}$ is the set of bad boxes for $\square_{\vec{v}}$.

General case

- **Back to the dynamic problem**
- When inserting/deleting a box B...
  - For each $\square_{\vec{v}}$ where B is good, update in $O(\log n)$ time.
  - For each $\square_{\vec{v}}$ where B is bad, apply our offline algorithm with the best ordering $\sigma \in \Sigma_{[d-1]}$ to compute the depth function in time

$$O\left(\min_{\sigma \in \Sigma_{[d-1]}} (n/r)\log(n/r) \cdot \Phi_\sigma(\mathbf{proj}_{[d-1]}(\mathcal{B}_{\vec{v}}^{\mathrm{bad}}))\right)$$

where $\mathcal{B}_{\vec{v}}^{\mathrm{bad}}$ is the set of bad boxes for $\square_{\vec{v}}$.

- Total update time $= \widetilde{O}(r^{d-1}) + \widetilde{O}((n/r)^{d-1} + nr^{d-3})$

General case

- **Back to the dynamic problem**
- When inserting/deleting a box B...
  - For each $\square_{\vec{v}}$ where B is good, update in $O(\log n)$ time.
  - For each $\square_{\vec{v}}$ where B is bad, apply our offline algorithm with the best ordering $\sigma \in \Sigma_{[d-1]}$ to compute the depth function in time

$$O\left(\min_{\sigma \in \Sigma_{[d-1]}} (n/r)\log(n/r) \cdot \Phi_\sigma(\mathbf{proj}_{[d-1]}(\mathcal{B}_{\vec{v}}^{\text{bad}}))\right)$$

    where $\mathcal{B}_{\vec{v}}^{\text{bad}}$ is the set of bad boxes for $\square_{\vec{v}}$.

- Total update time $= \widetilde{O}(r^{d-1}) + \widetilde{O}((n/r)^{d-1} + nr^{d-3})$
- Setting $r = \sqrt{n}$ gives us $\widetilde{O}(n^{(d-1)/2})$ update time.

Open questions

Open questions

- Better preprocessing time and space?

Open questions

- Better preprocessing time and space?
- Problems for other shapes (e.g., disks)?

Open questions

- Better preprocessing time and space?
- Problems for other shapes (e.g., disks)?
- Approximate data structures?

Thank you!
Q & A