Improved Sublinear Algorithms for Testing Permutation Freeness

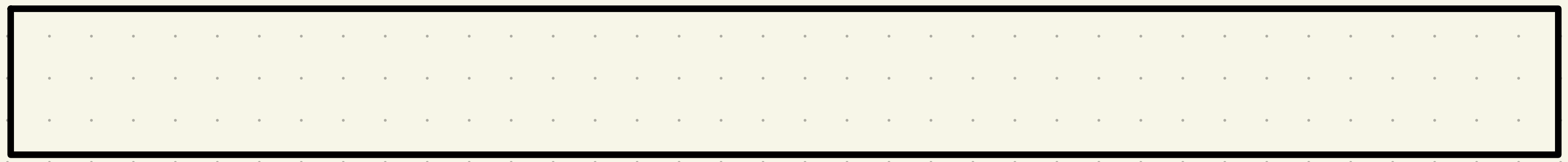
Ilan Newman

אוניברסיטת חיפה
University of Haifa
جامعة حيفا

Nithin Varma

cmi
CHENNAI MATHEMATICAL INSTITUTE

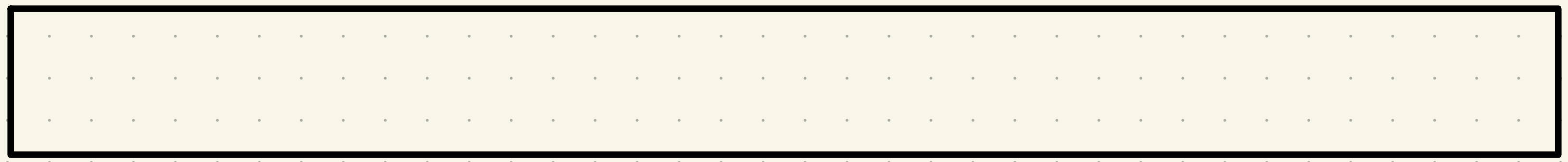
Ordered Patterns in Arrays



array A
of length n

- Let $\pi : [k] \to [k]$ be a bijection

Ordered Patterns in Arrays



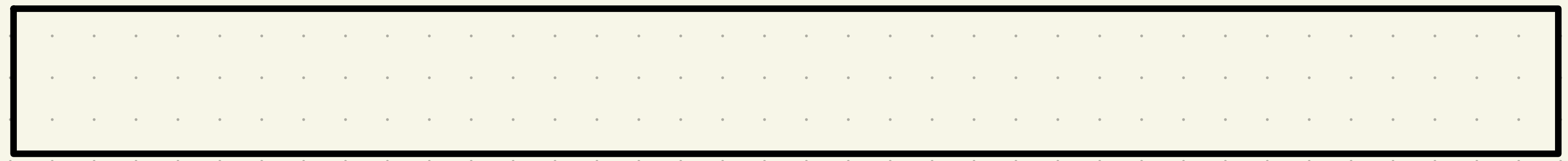
array A of length n

- Let $\pi : [k] \to [k]$ be a bijection
- A has a π-appearance if

$\exists$ indices $i_1 < i_2 < \dots < i_k$ such that

$A[i_a] > A[i_b]$ if $\pi(a) > \pi(b)$ $\forall a,b \in [k]$

Ordered Patterns in Arrays



array A of length n

- Let $\pi : [k] \to [k]$ be a bijection
- A has a π-appearance if

  $\exists$ indices $i_1 < i_2 < \ldots < i_k$ such that

  $$A[i_a] > A[i_b] \quad \text{if} \quad \pi(a) > \pi(b) \quad \forall a, b \in [k]$$
- A is π-free if it has no π-appearance

Examples

100	98	723	1.2	5.88

Examples

100	98	723	1.2	5.88

⭐ has a (2, 1, 3) - appearance

723 3

100 2

98 1

Examples

100	98	723	1.2	5.88

(1, 2, 3, 4) - free

- has a (2, 1, 3) - appearance
- " (3, 1, 2) - appearance
- " (3, 4, 1, 2) - appearance
- " (2, 3, 1) - appearance

Problem

Given array A, permutation π

- Is A π-free ?
- Is A ε-far from π-free ?

Problem

$\boxed{\text{Hamming distance of } A \text{ to every } \pi\text{-free array is } \geq \varepsilon n}$

Given array A, permutation π

- Is A π-free?
- Is A ε-far from π-free? $\varepsilon \in (0,1)$

Problem

$\boxed{\text{Hamming distance of } A \text{ to every } \pi\text{-free array is } \geq \varepsilon n}$

Given array A, permutation π

- Is A π-free?
- Is A ε-far from π-free?

Sortedness

|||

(2, 1) - freeness

Generalization of monotonicity testing on arrays

[EKKRV00, DGLRRS99, ... PRV18]

History

- Initiated by [NRRS'19]

History

- Initiated by [NRRS'19]

- Studied extensively for monotone patterns

$$(1, 2, \ldots, k) \quad \text{or} \quad (k, k-1, \ldots, 2, 1)$$

[NRRS'19, BCLW'19, BLW'19]

Highlights

OPTIMAL

- $O(\log n)$-query tester for monotone patterns of constant length

[BLW '19]

Highlights

- $O(\log n)$-query tester for monotone patterns of constant length

  [BLW '19]

- polylog n query tester for arbitrary patterns of length 3

  [NRRS '19]

1 2 3 4 6 5 10

Highlights

- $O(\log n)$-query tester for monotone patterns of constant length

  [BLW '19]

- polylog n query tester for arbitrary patterns of length 3

  [NRRS '19]

What about nonmonotone patterns of length > 3?

- Nonadaptive testers $O(n^{1-\frac{1}{k-1}})$ queries

[NRRS '19]

- Nonadaptive testers $O(n^{1-\frac{1}{k-1}})$ queries

[NRRS '19]

- Nonadaptive testers cannot do better!

[BC '18]

- Nonadaptive testers $O(n^{1-\frac{1}{k-1}})$ queries

[NRRS '19]

- Nonadaptive testers cannot do better!

[BC '18]

What about adaptive testers?

Our Result

Let $\varepsilon \in (0,1)$, $k \in \mathbb{N}$ and $\pi \in S_k$.

There is an ε-tester for π-freeness with query complexity $\tilde{O}\left(n^{o(1)}\right)$.

Our Result

Let $\varepsilon \in (0,1)$, $k \in \mathbb{N}$ and $\pi \in S_k$.

There is an ε-tester for π-freeness with query complexity $\tilde{O}(n^{o(1)})$.

The tester has one-sided error.

always accepts π-free arrays

Features of our result

- Strong sublinear-time guarantee

Features of our result

- Strong sublinear-time guarantee

- Our techniques are general and work for all Π

Today: $\tilde{O}(\sqrt{n})$-query algorithm for testing π-freeness of $\pi \in S_4$

↳ every n.a. algo. for this problem has q.c. $\Omega(n^{2/3})$

Today: $\tilde{O}(\sqrt{n})$-query algorithm for testing π-freeness of $\pi \in S_4$

$$\left\{ \text{⭐ Find a } \pi\text{-appearance in an array that is } \varepsilon\text{-far from } \pi\text{-free} \right.$$

First Useful Fact

Array A is ε-far from π-free

$\Rightarrow$

Matching of π-appearances

of size $\geq \frac{\varepsilon n}{4}$

First Useful Fact

Used by all algorithms to test π-freeness

Array A is ε-far from π-free

$\Rightarrow$

Matching of π-appearances of size $\geq \frac{\varepsilon n}{4}$

First Useful Fact

Used by all algorithms to test π-freeness

Array A is ε-far from π-free

$$\Rightarrow$$

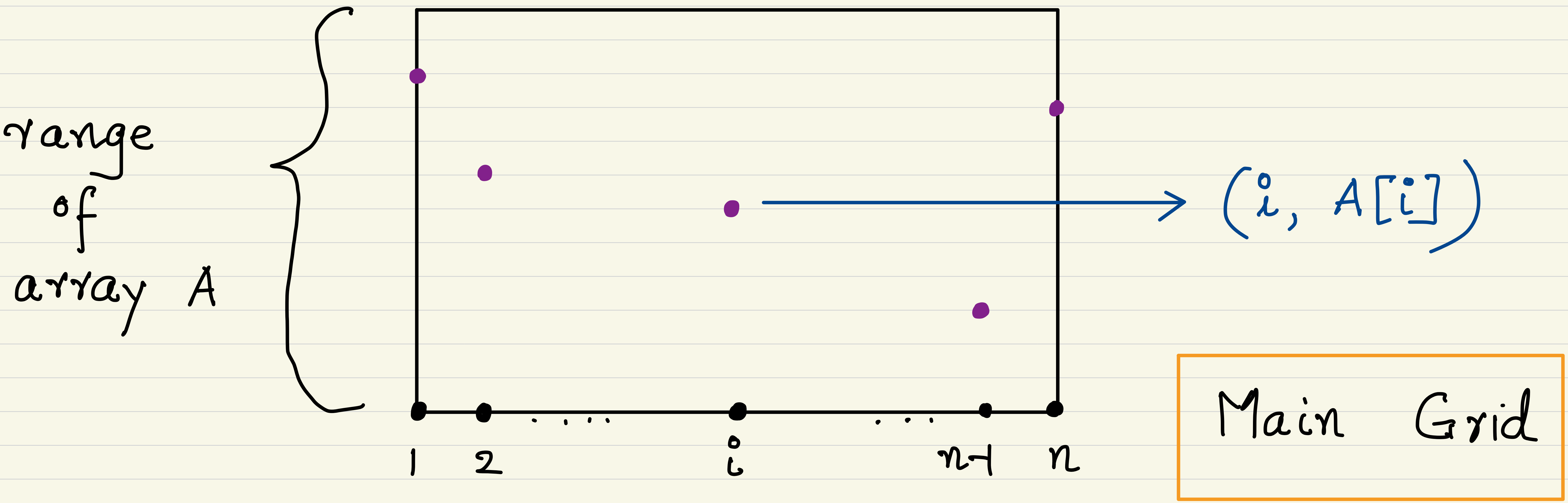
Matching of π-appearances of size $\geq \frac{\varepsilon n}{4}$

E.g. 100, 99, 78, 98, 77, 97, 76, 21

$(4,3,2,1)$ - appearances

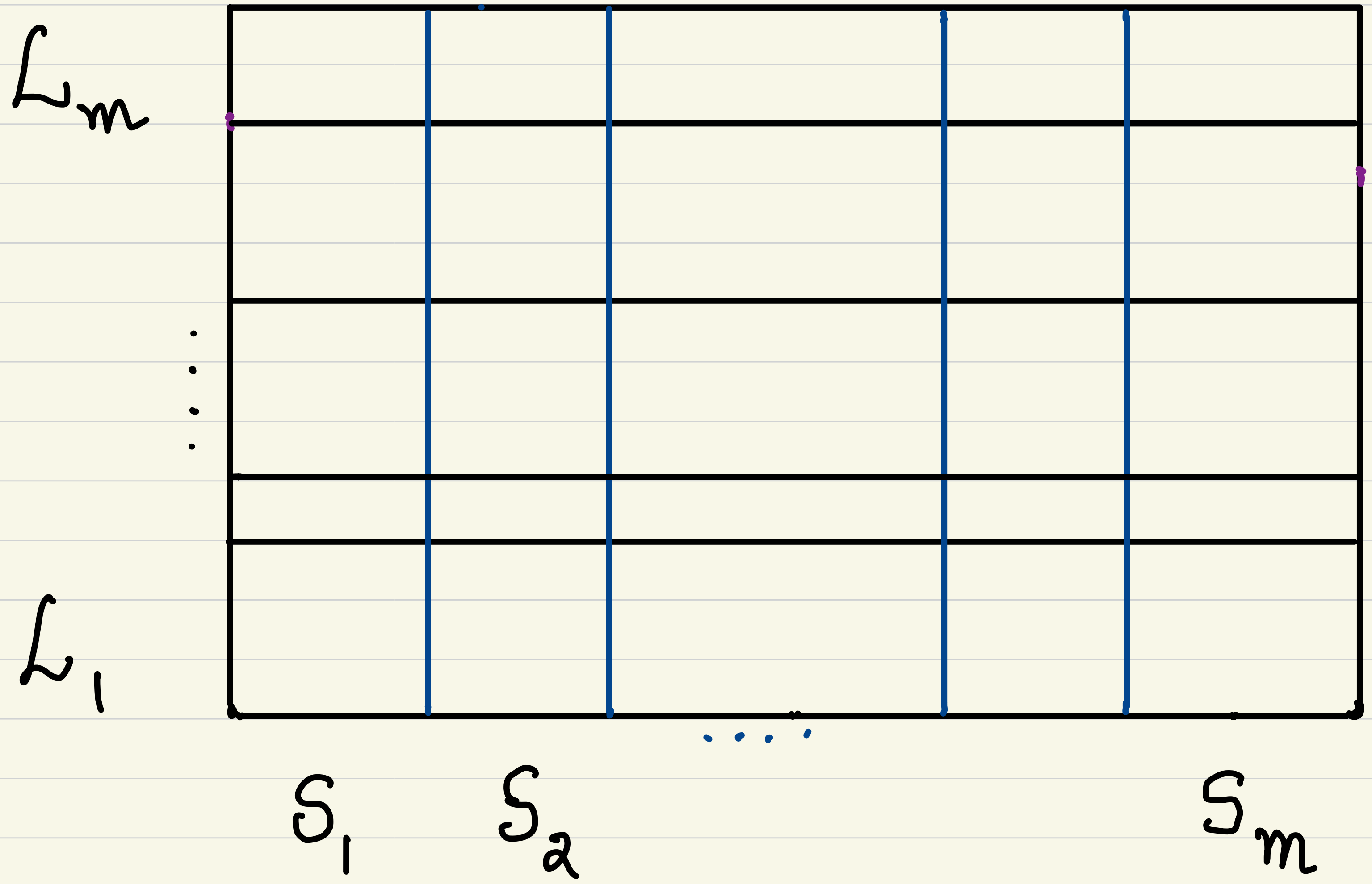
Key Ingredient

⭐ View the array as a grid of n points in $[n] \times \mathbb{R}$



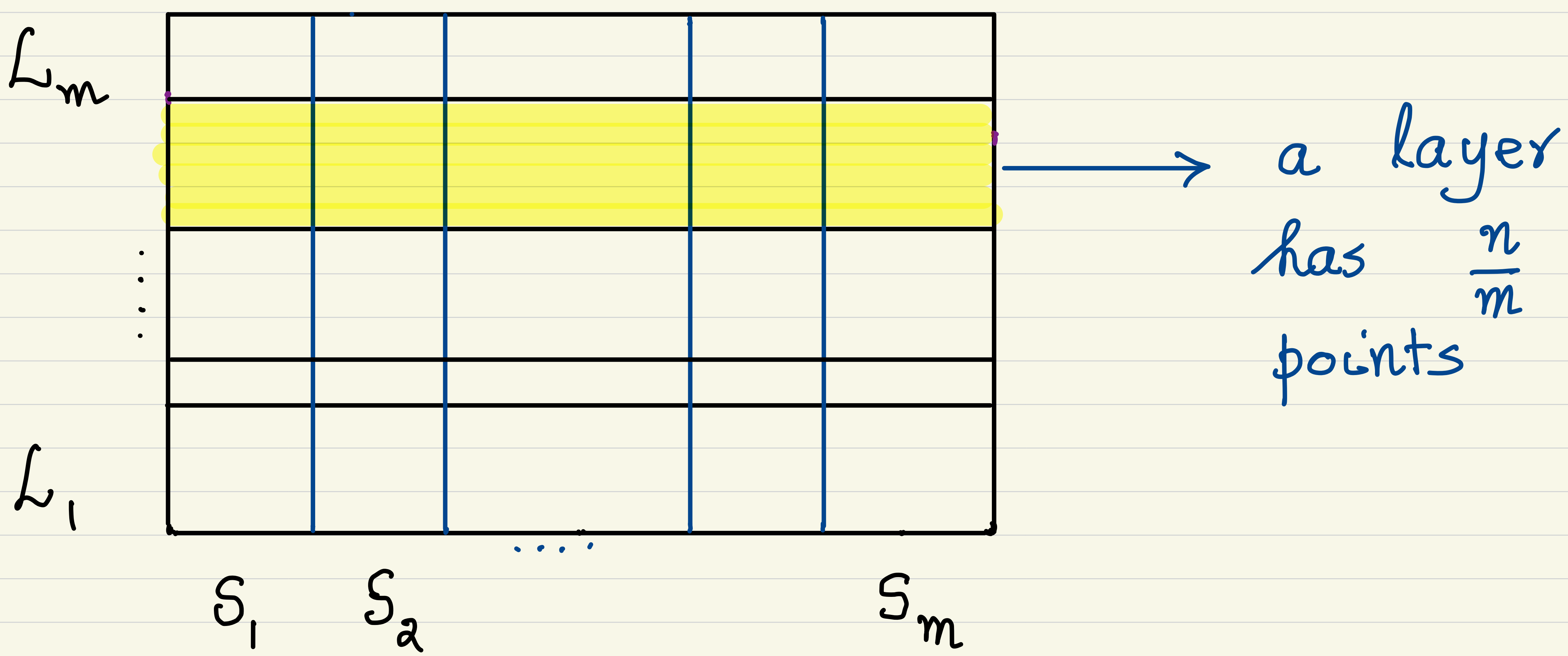
Grid of Points

⭐ Our algorithm makes use of an $m \times m$ partition of the main grid.



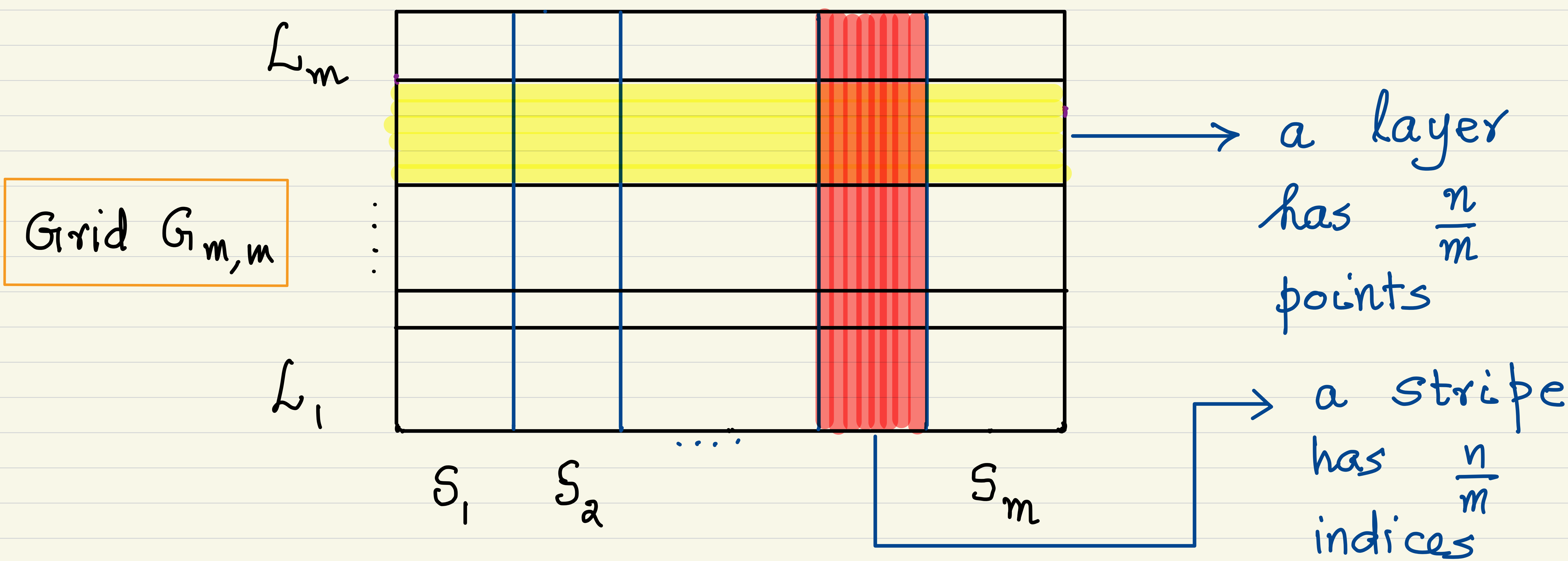
Grid of Points

- Our algorithm makes use of an $m \times m$ partition of the main grid.



Grid of Points

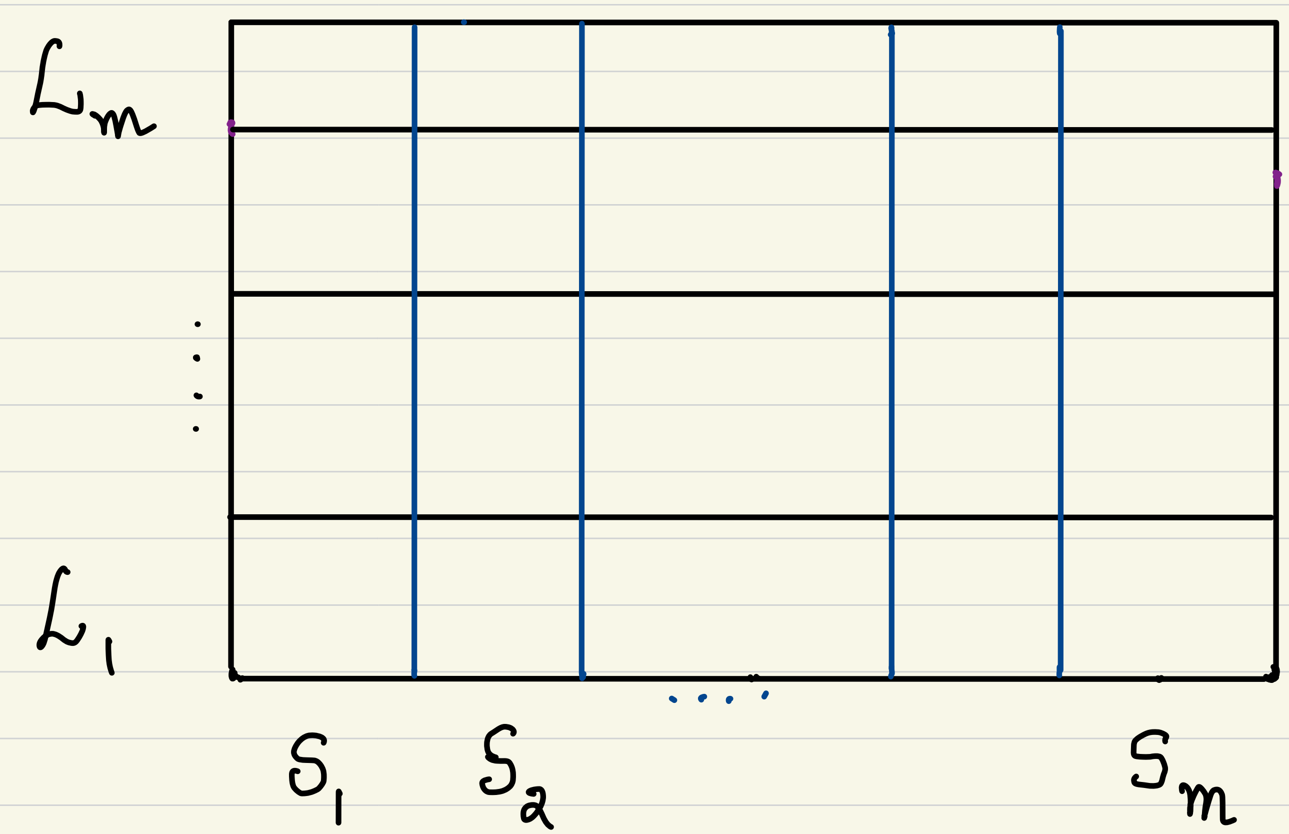
- Our algorithm makes use of an $m \times m$ partition of the main grid.



Gridding

- Determine layers so that each layer has $\sim \frac{n}{m}$ points

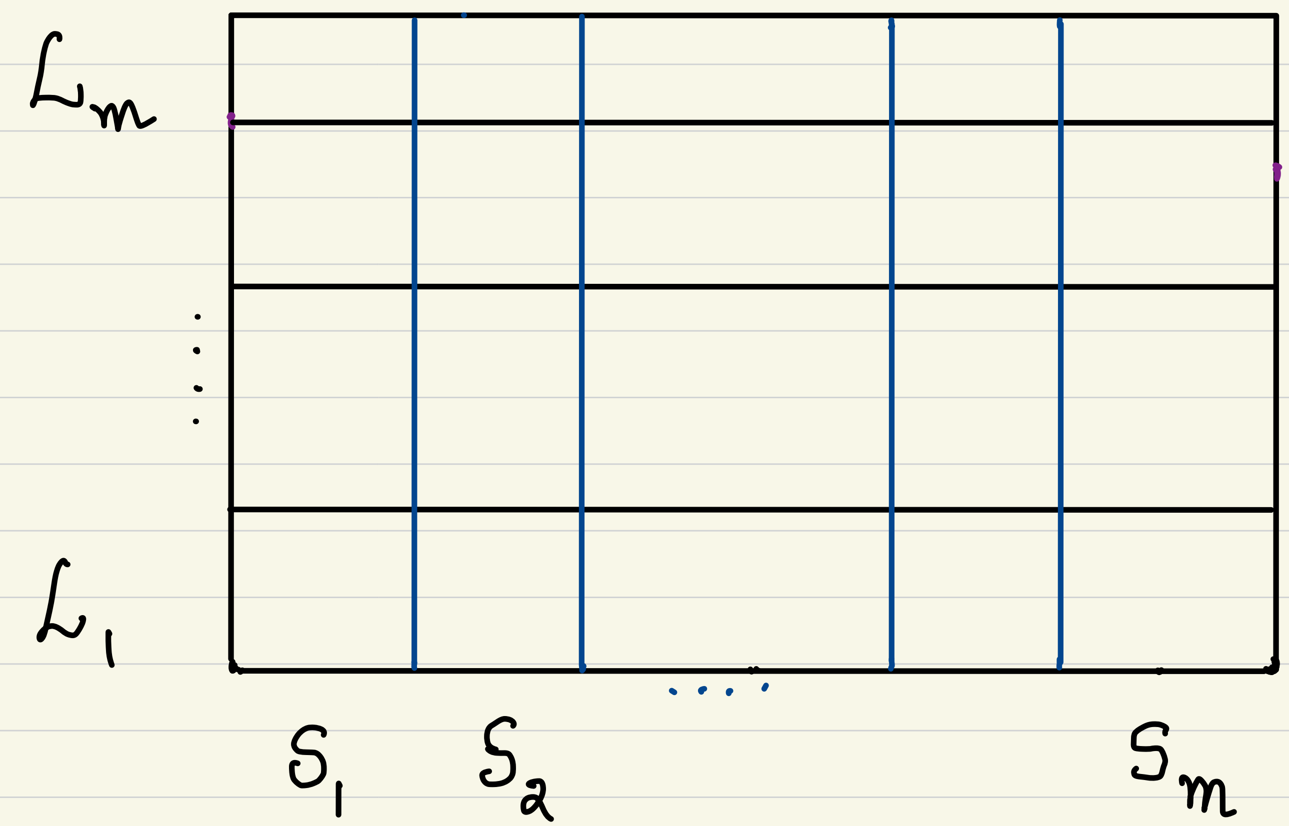
- Partition $[n]$ into m stripes of $\frac{n}{m}$ indices



$\tilde{O}(m)$ queries

Gridding

- Determine layers so that each layer has $\sim \frac{n}{m}$ points
- Partition $[n]$ into m stripes of $\frac{n}{m}$ indices

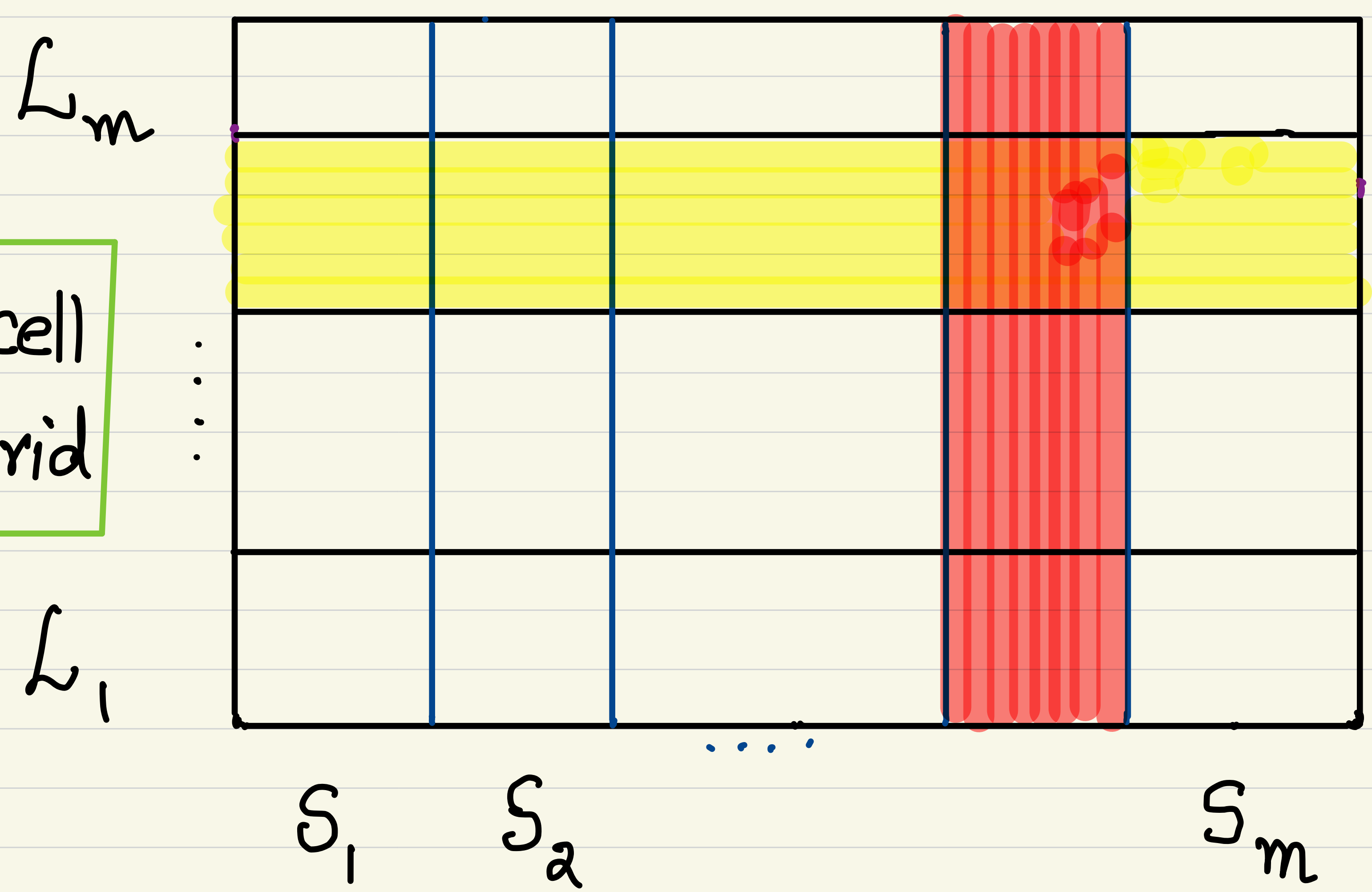


Set $m \leftarrow \sqrt{n}$

$\boxed{\tilde{O}(m) \text{ queries}}$

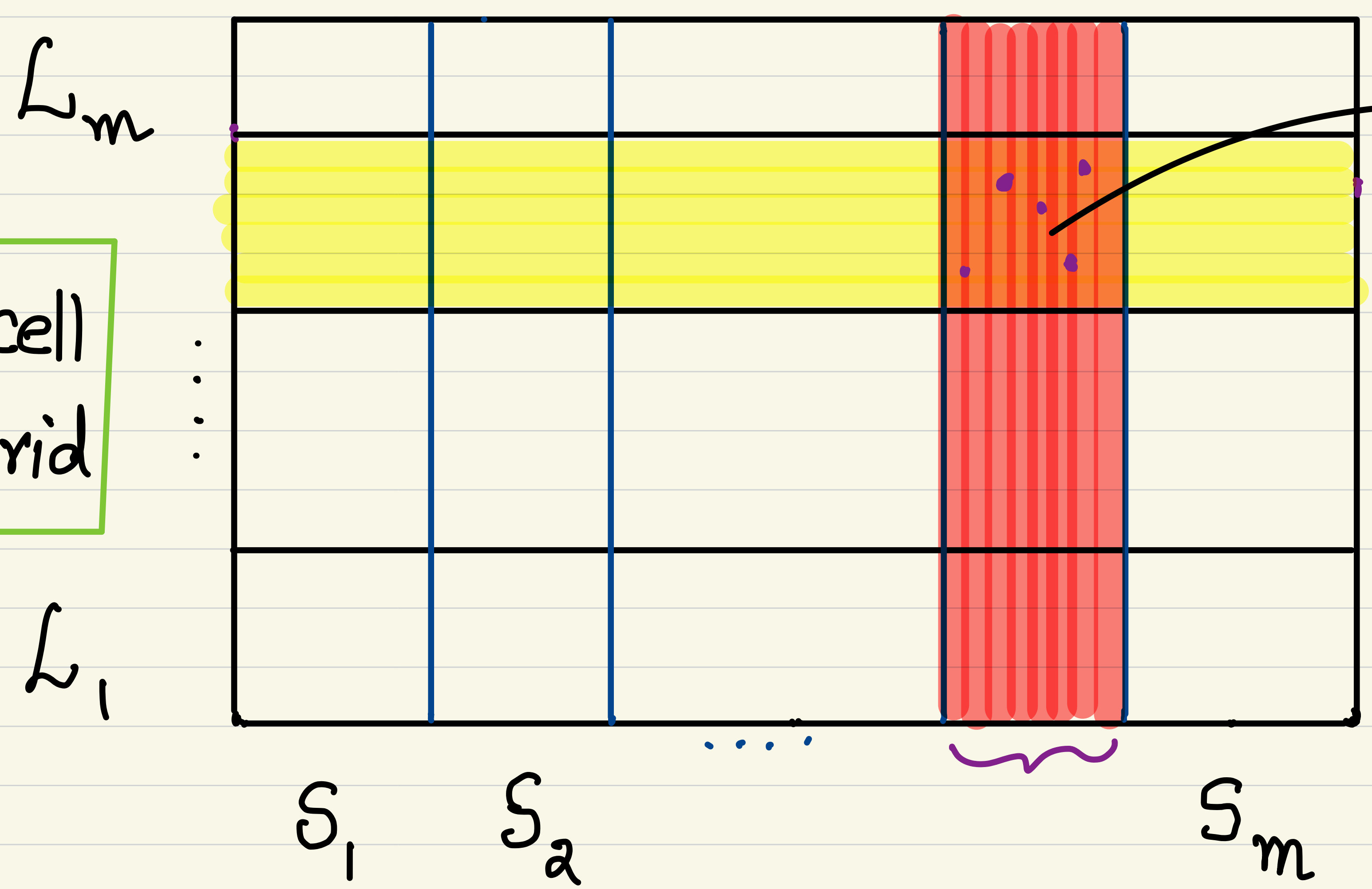
Boxes in the grid

Box: A cell in the grid



Boxes in the grid

Box: A cell in the grid



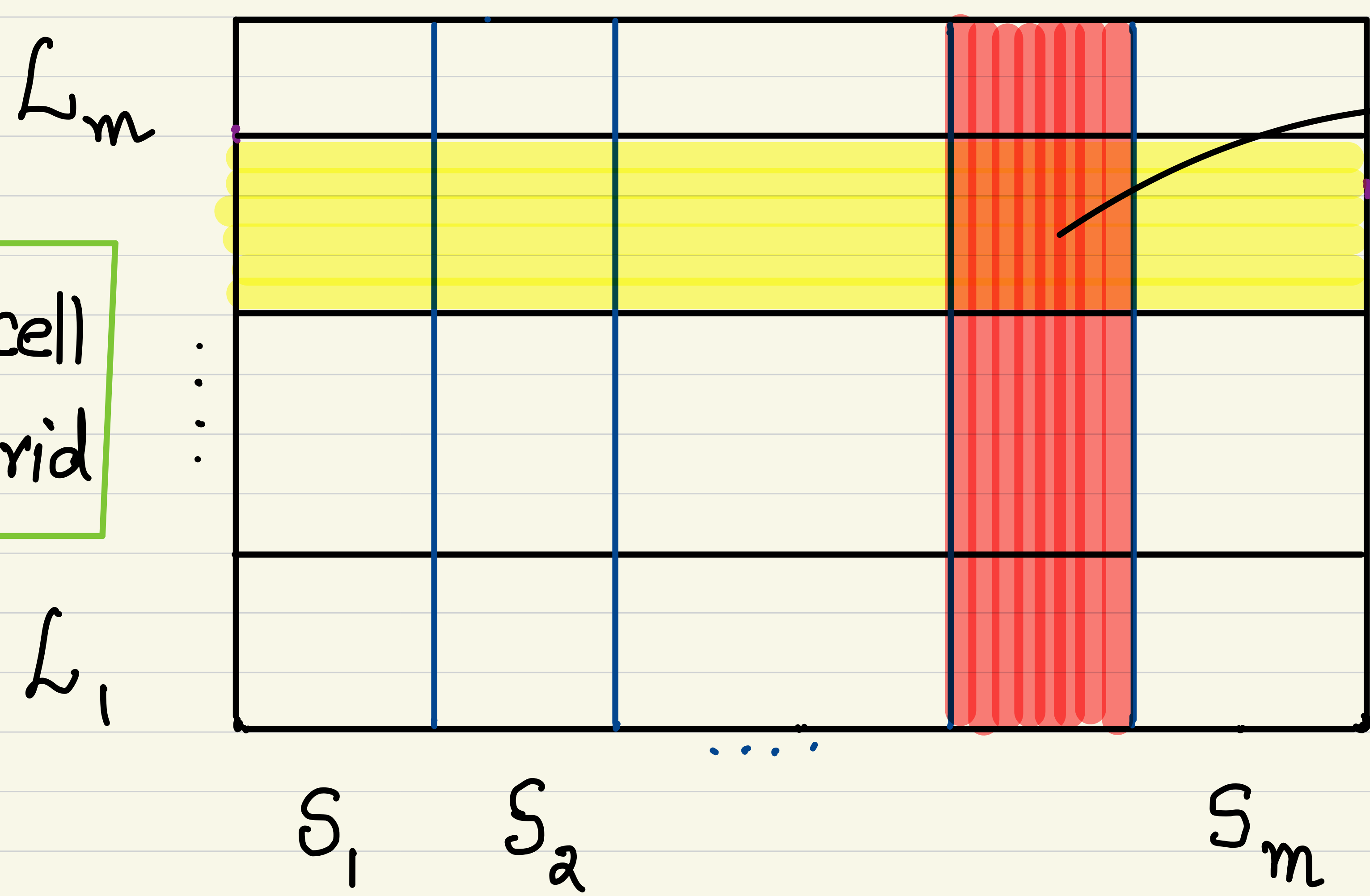
density of box (S_i, L_j)

$$= \frac{\#\text{ points in box }(S_i, L_j)}{|S_i|}$$

Boxes in the grid

Next Goal: Determine the distribution of points among the boxes by sampling

Box: A cell in the grid

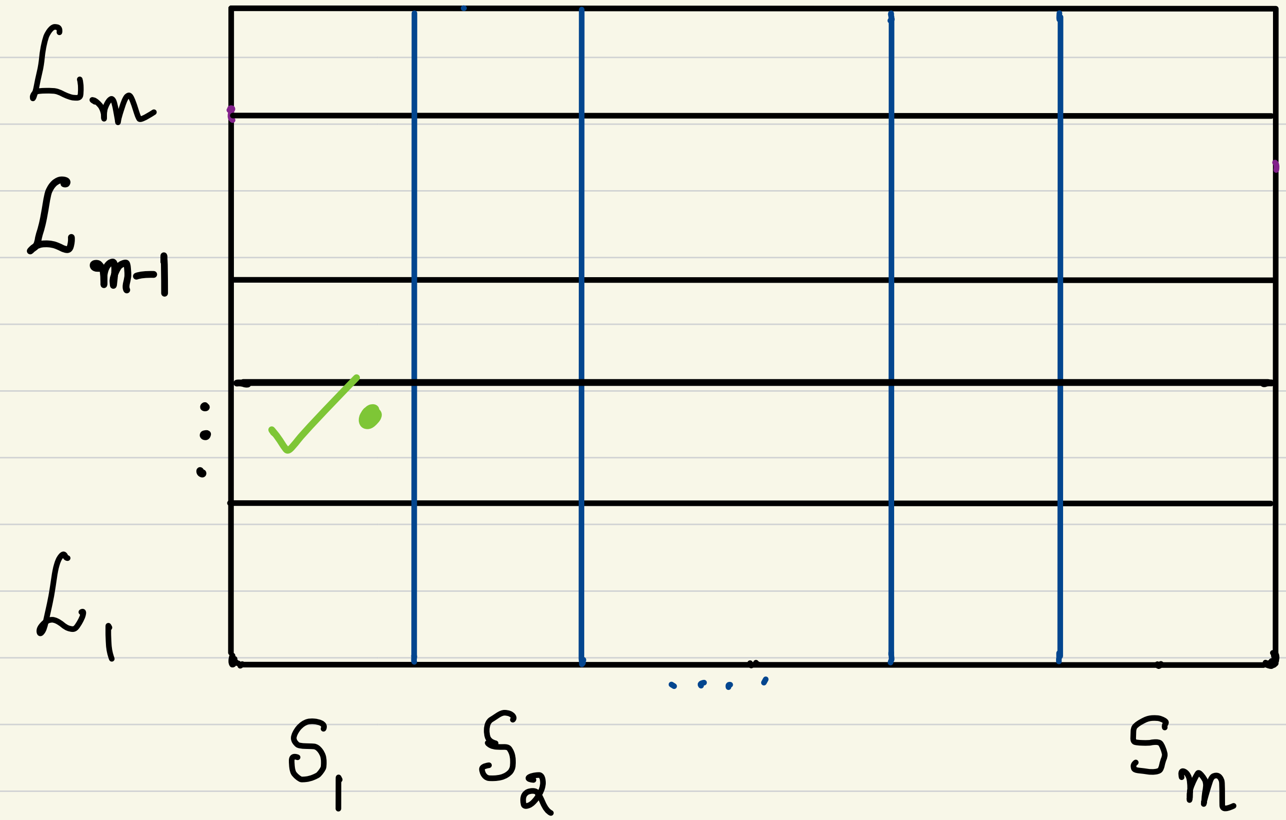


density of box (S_i, L_j)

$$= \frac{\#\text{ points in box } (S_i, L_j)}{|S_i|}$$

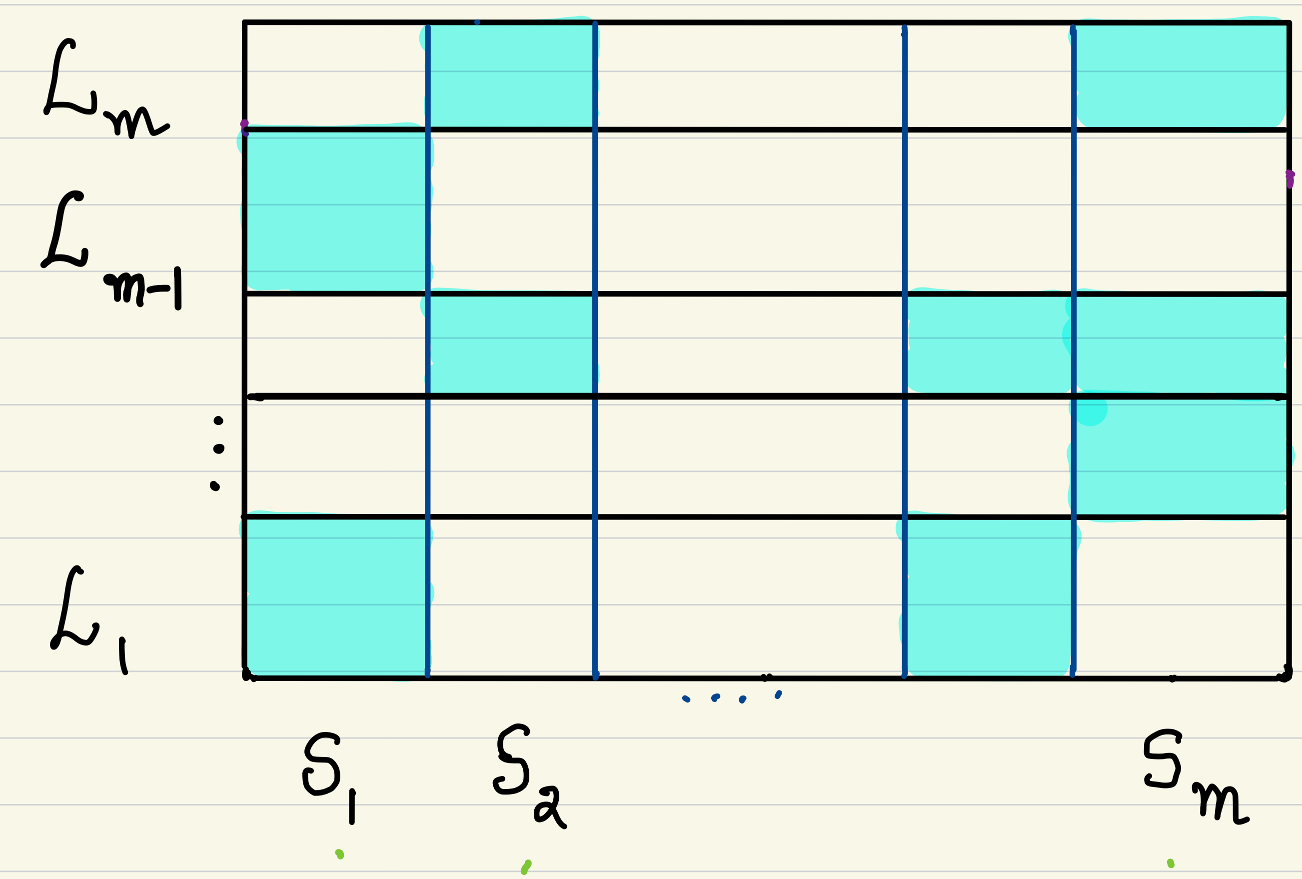
Gridding : Part 2

- ⭐ From each stripe, sample $\tilde{O}(1)$ $\rightarrow \log^{100} n$ points & mark the boxes with at least one sampled point



Gridding : Part 2

⭐ From each stripe, sample $\tilde{O}(1)$ points & mark the boxes with at least one sampled point



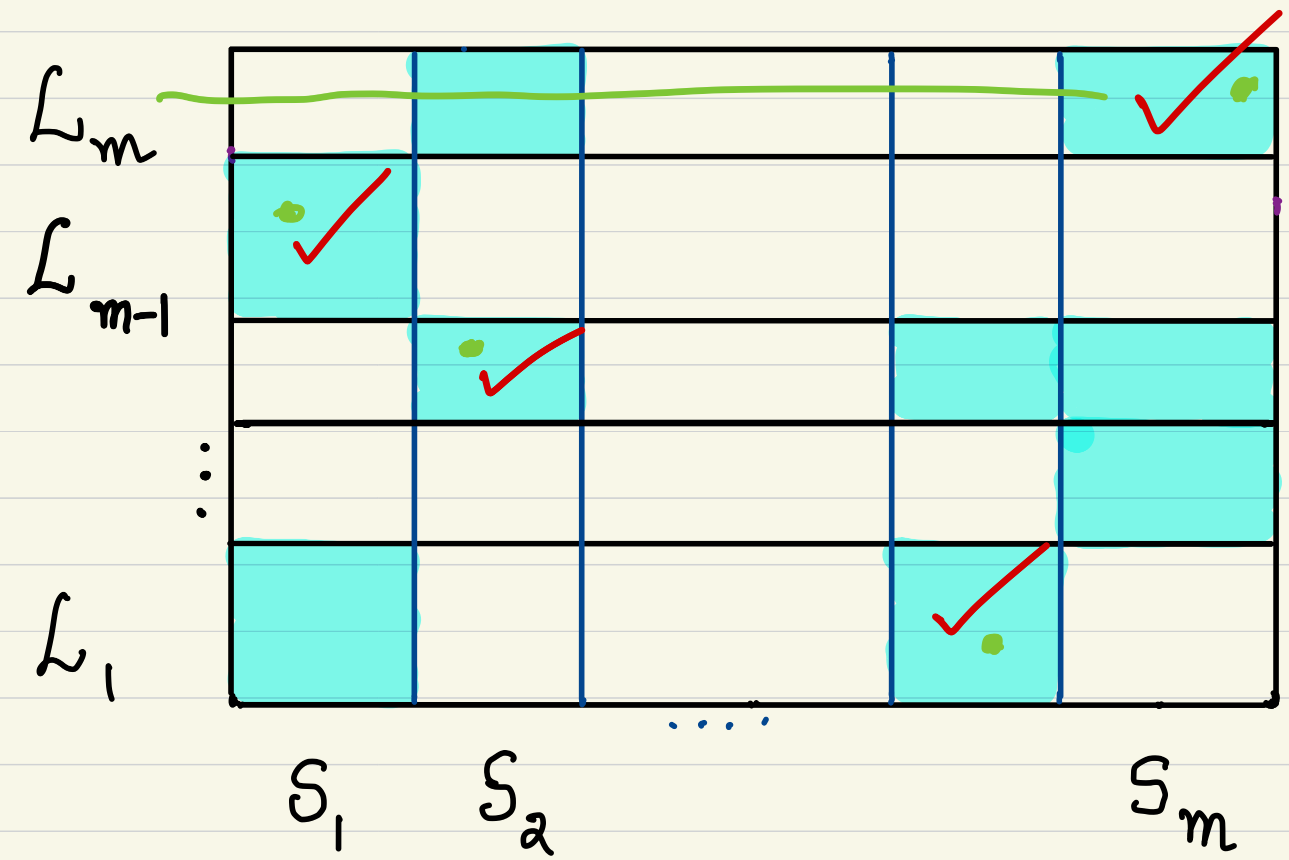
There are m^2 boxes, out of which we mark $\tilde{O}(m)$ boxes only

Marcus-Tardos Helps Us

Lemma [MT '04]: For any $\pi \in S_k$, $\exists$ a constant $\hat{K}(k)$ such that for any $r \in \mathbb{N}$, if grid $G_{r,r}$ has $\geq \hat{K}(k) \cdot r$ marked cells, then there is a π-appearance among the cells.

Gridding : Part 2

- From each stripe, sample $\tilde{O}(1)$ points & mark the boxes with at least one sampled point
- Reject if π-appearance found

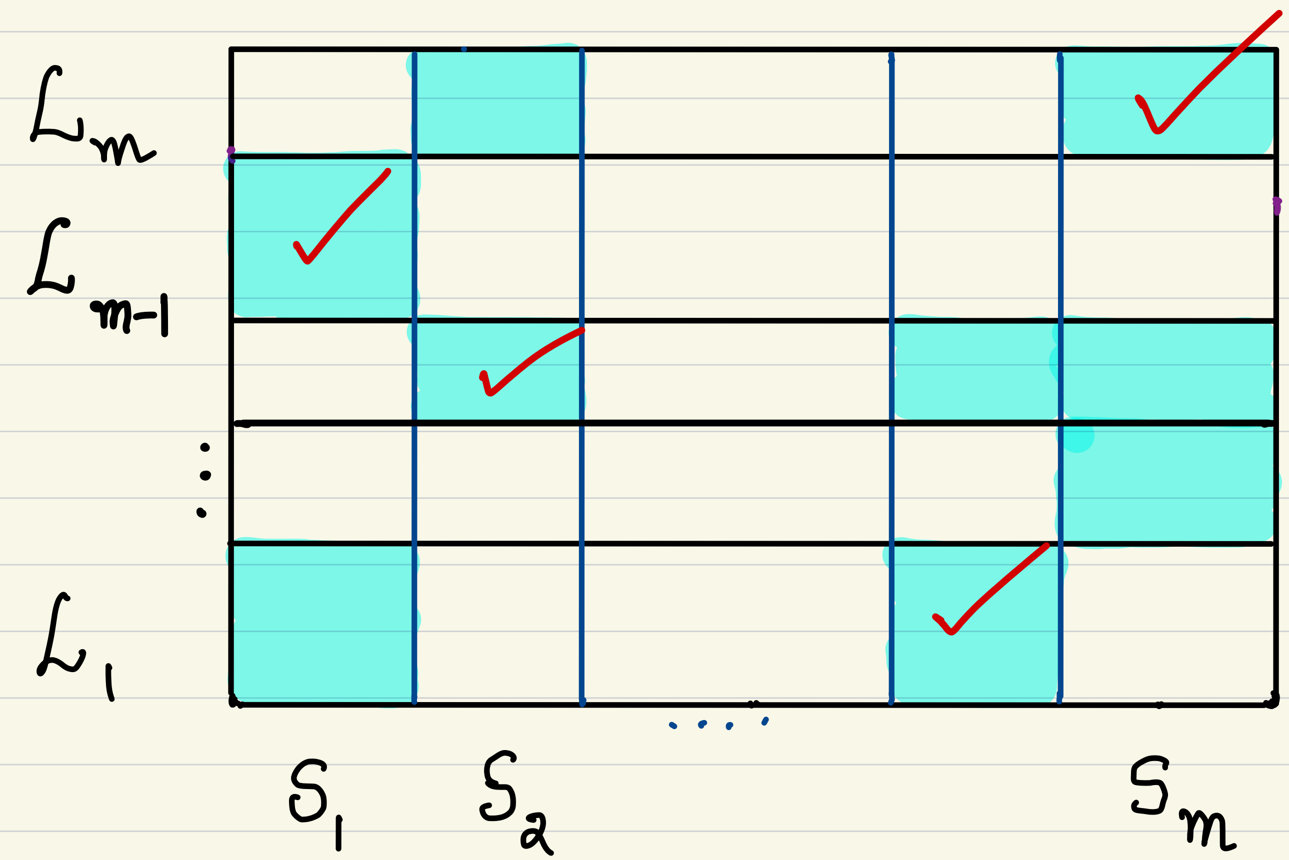


Suppose $\pi = (3, 2, 1, 4)$

There is a π-appearance among marked boxes

Gridding : Part 2

- From each stripe, sample $\tilde{O}(1)$ points & mark the boxes with at least one sampled point
- Reject if π-appearance found



Suppose $\pi = (3, 2, 1, 4)$

There is a π-appearance among marked boxes

After marking boxes

(MT '04)

- If there are more than $\underset{\sim}{\hat{K}(4) \cdot m}$ marked boxes, we are done!

  $\hookrightarrow$ imm. detect a Π-app.

- Assume we have $\leq \hat{K}(4) \cdot m$ marked boxes

After marking boxes

- If there are more than $\hat{K}(4) \cdot m$ marked boxes, we are done!

- Assume we have $\leq \hat{K}(4) \cdot m$ marked boxes

Can we ignore the unmarked boxes ?

After marking boxes

⭐ Lemma : With high probability, for each stripe S

- either has $\tilde{\Omega}(1)$ marked boxes

After marking boxes

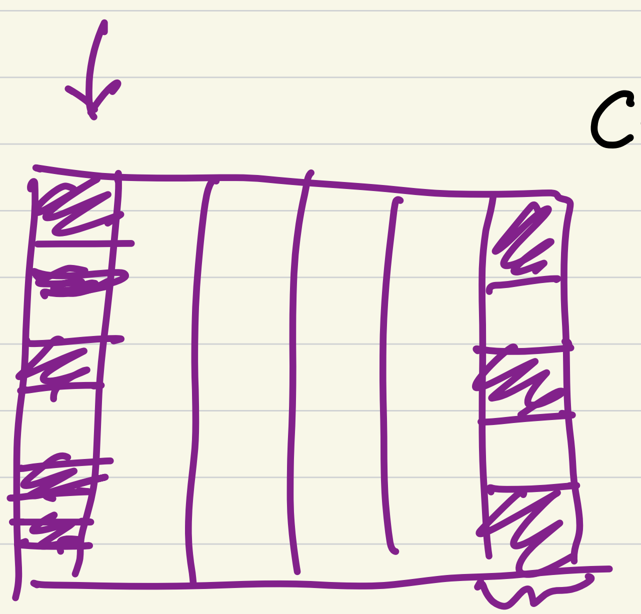
$\leqslant K\cdot(4)\cdot m$ marked boxes

Lemma : With high probability, for each stripe S

$\log^{100} n$

$\Omega(n)$

- either has $\tilde{\Omega}(1)$ marked boxes
- or union of marked boxes contain $(1 - o(1))\cdot|S|$ points



$o(1)\cdot|S|$ $o(1)\cdot n$

After marking boxes

⭐ <u>Lemma</u> : With high probability, for each stripe S

- either has $\tilde{\Omega}(1)$ marked boxes
- or union of marked boxes contain $(1-o(1))\cdot|S|$ points

OK to ignore unmarked boxes!

After marking boxes

- OK to ignore unmarked boxes

After marking boxes

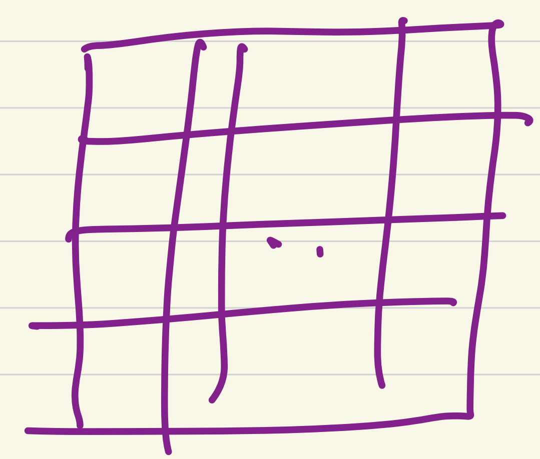
- OK to ignore unmarked boxes
- Ignore boxes with "low" density

After marking boxes

- OK to ignore unmarked boxes
- Ignore boxes with "low" density
- $O(m)$ dense boxes overall

After marking boxes

- OK to ignore unmarked boxes
- Ignore boxes with "low" density
- $O(m)$ dense boxes overall
- Only a small constant fraction of layers & stripes have $> d(\varepsilon)$ dense boxes



After marking boxes

- OK to ignore unmarked boxes
- Ignore boxes with "low" density
- $O(m)$ dense boxes overall
- Only a small constant fraction of layers & stripes have $> d(\varepsilon)$ dense boxes
- Ignore points in such layers & stripes

After Gridding

- $m \times m$ grid with $O(m)$ dense boxes

After Gridding

- $m \times m$ grid with $O(m)$ dense boxes
- Each layer / stripe has $\leq d$ dense boxes

After Gridding

- $m \times m$ grid with $O(m)$ dense boxes
- Each layer / stripe has $\leq d$ dense boxes
- There is a matching of π-appearances of size $\Omega(\varepsilon n)$ among dense boxes

After Gridding

- $m \times m$ grid with $O(m)$ dense boxes
- Each layer / stripe has $\leq d$ dense boxes
- There is a matching of π-appearances of size $\Omega(\varepsilon n)$

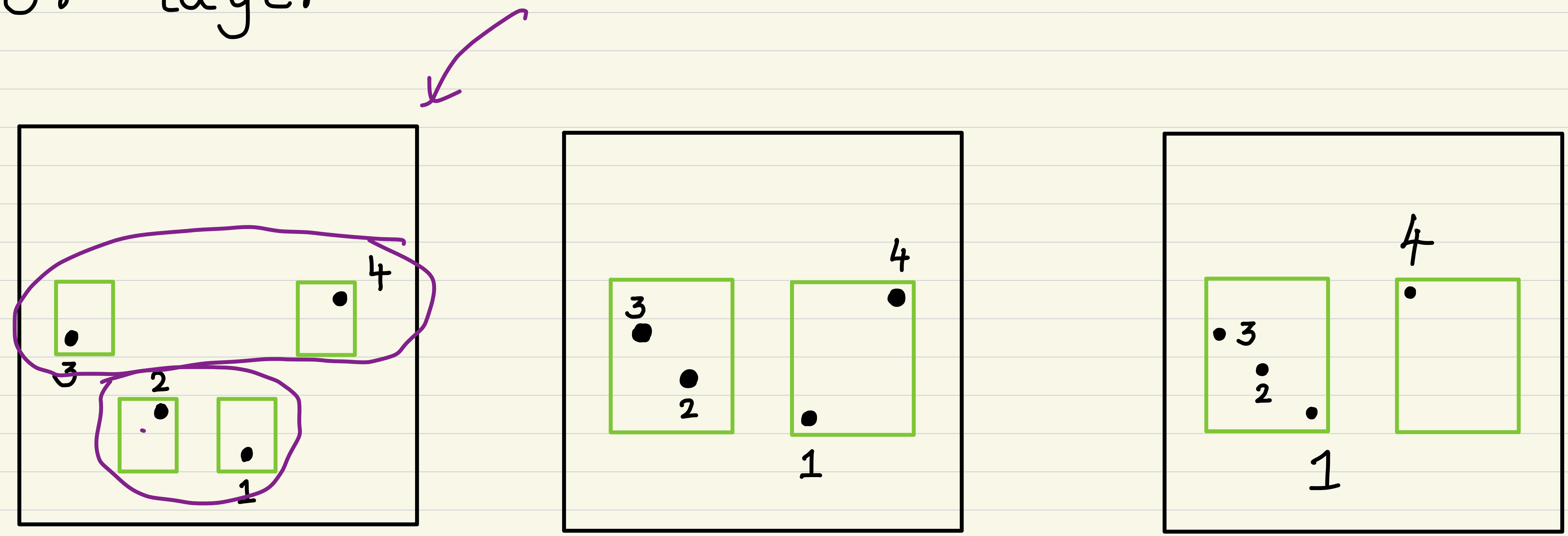
Where in the grid do these appear?

If Gridding step did not reject ...

- Most π-appearances have more than one leg in boxes that share a stripe or layer

If Gridding step did not reject ...

⭐ Most π-appearances have more than one leg in boxes that share a stripe or layer

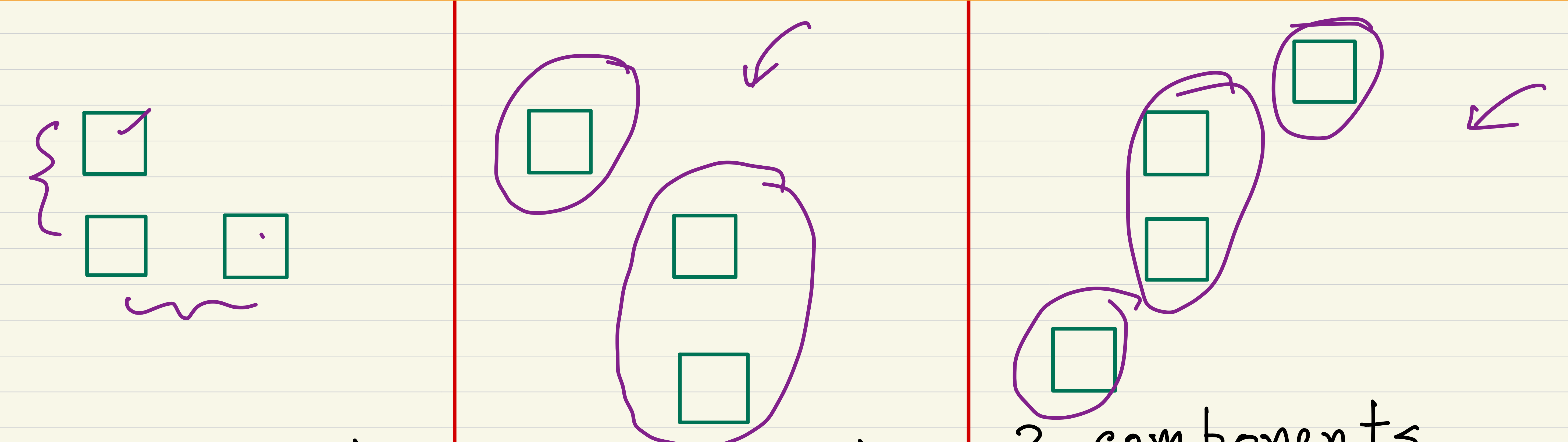


Different types of π-appearances

- Configuration: Arrangement of ≤ 4 boxes and a mapping of legs of π-appearance into them
- Two boxes B, B' in a configuration are directly-connected if they share a layer or a stripe

Connected components in configuration

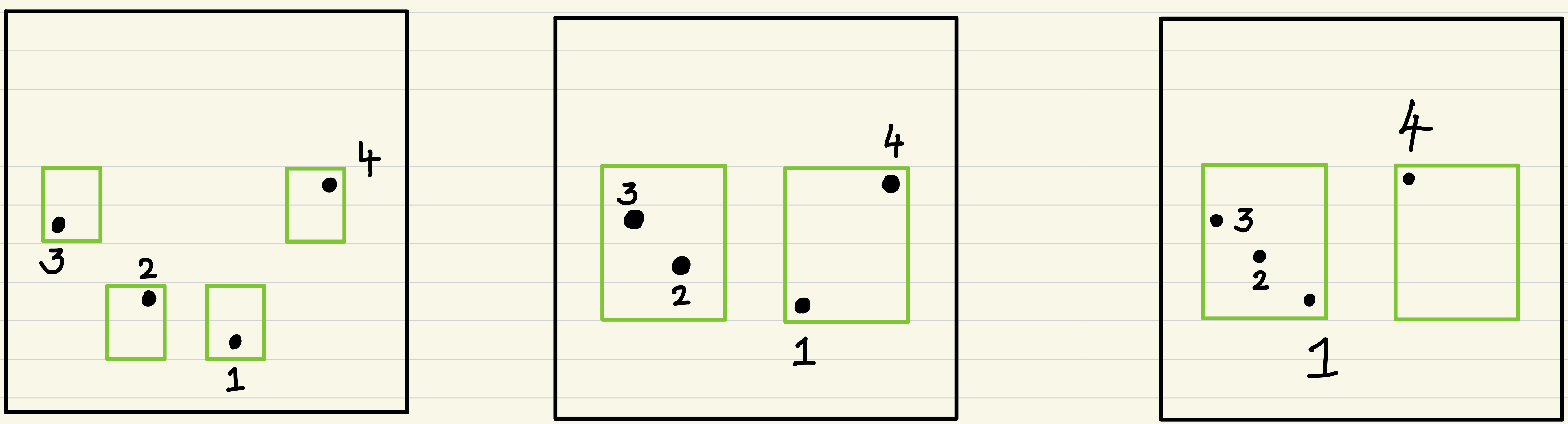
- Transitive closure of directly-connected relation is the connected relation



1 component | 2 components | 3 components

How to detect these?

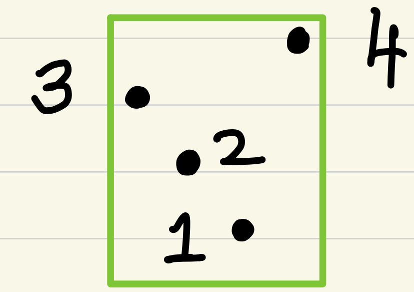
⭐ Only constantly many distinct types of configurations



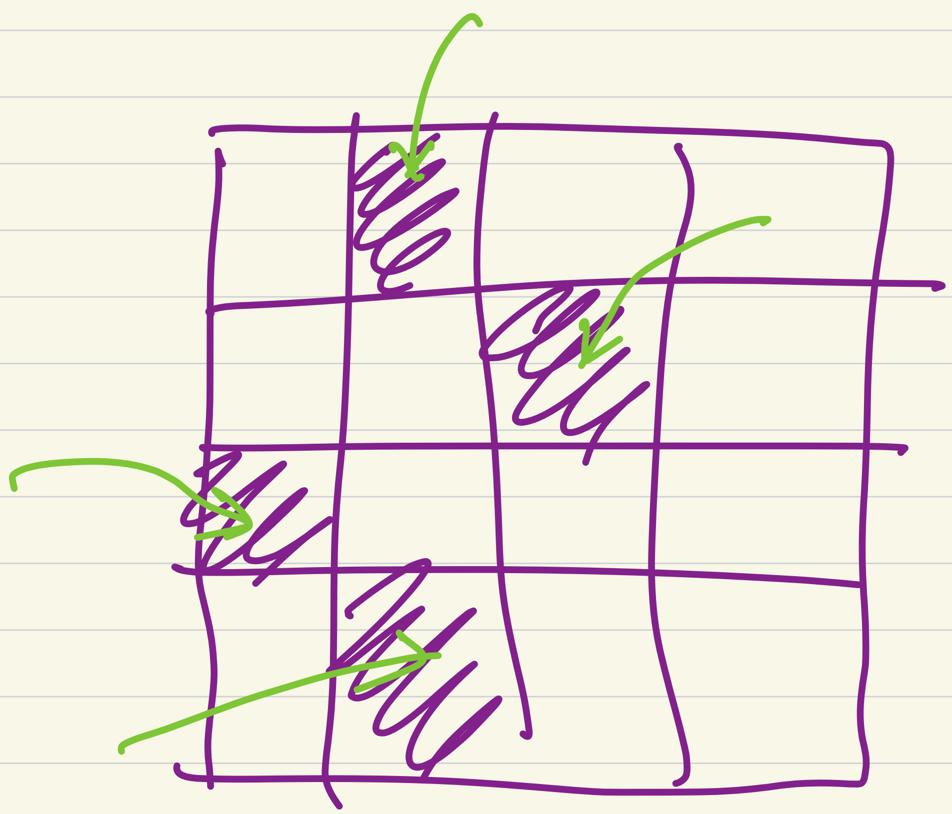
How to detect these? : High-level Schema

- for each configuration
  - Detect a π-appearance among dense boxes forming the configuration

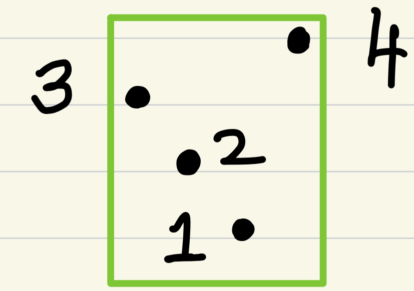
Case 1



⭐ $\Omega(\varepsilon n)$ π-appearances have all four legs in a single dense box



Case 1



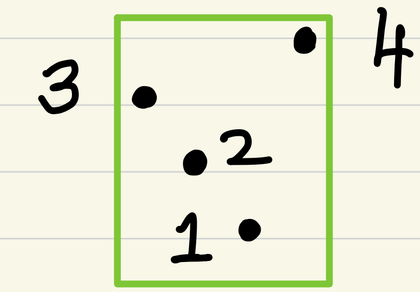
- $\Omega(\varepsilon n)$ π-appearances have all four legs in a single dense box
- $O(m)$ dense boxes overall

$$\Rightarrow \Omega(\varepsilon n / m) \quad \pi\text{-appearances}$$

~~per~~ dense box

Case 1

$\boxed{\Theta(\sqrt{n}) \text{ queries}}$

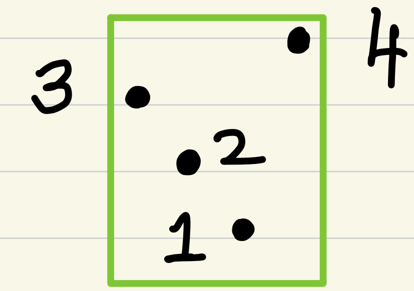


- $\Omega(\varepsilon n)$ π-appearances have all four legs in a single dense box
- $O(m)$ dense boxes overall

  $m = \sqrt{n}$

  $\Rightarrow \Omega(n/m)$ π-appearances per dense box
- Algorithm: Sample a random dense box and query all points in it

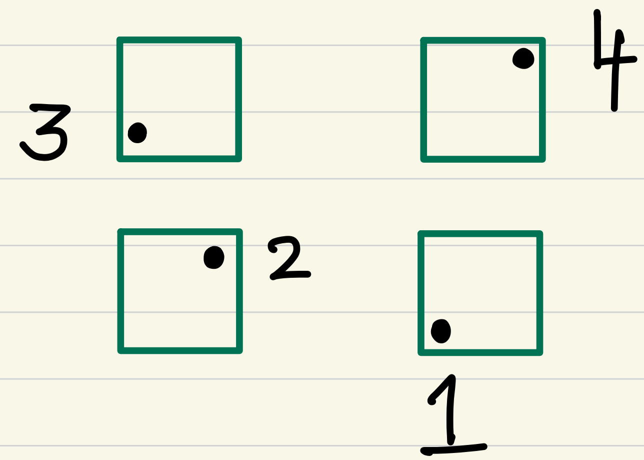
Case 1



- $\Omega(\varepsilon n)$ π-appearances have all four legs in a single dense box
- $O(m)$ dense boxes overall

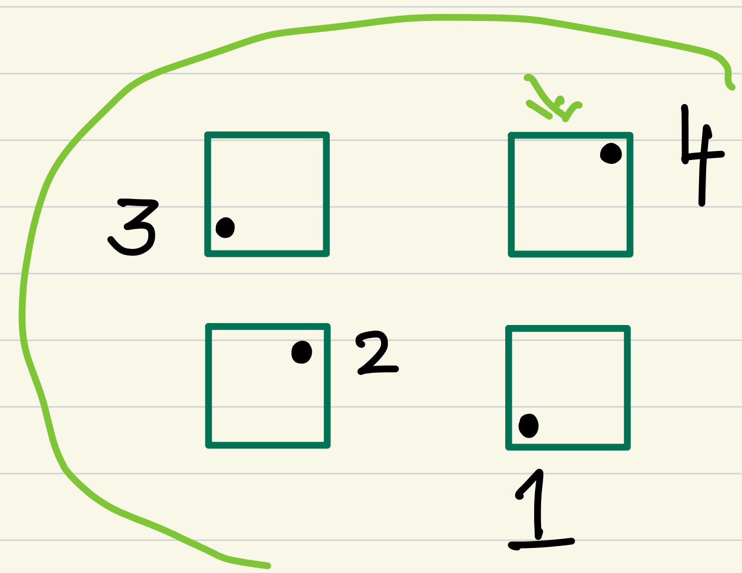
  $\Rightarrow \Omega(n/m)$ π-appearances per dense box
- <u>Algorithm</u>: Sample a random dense box and query all points in it

Case 1a: $\Omega(\varepsilon n)$ π-appearances are present as



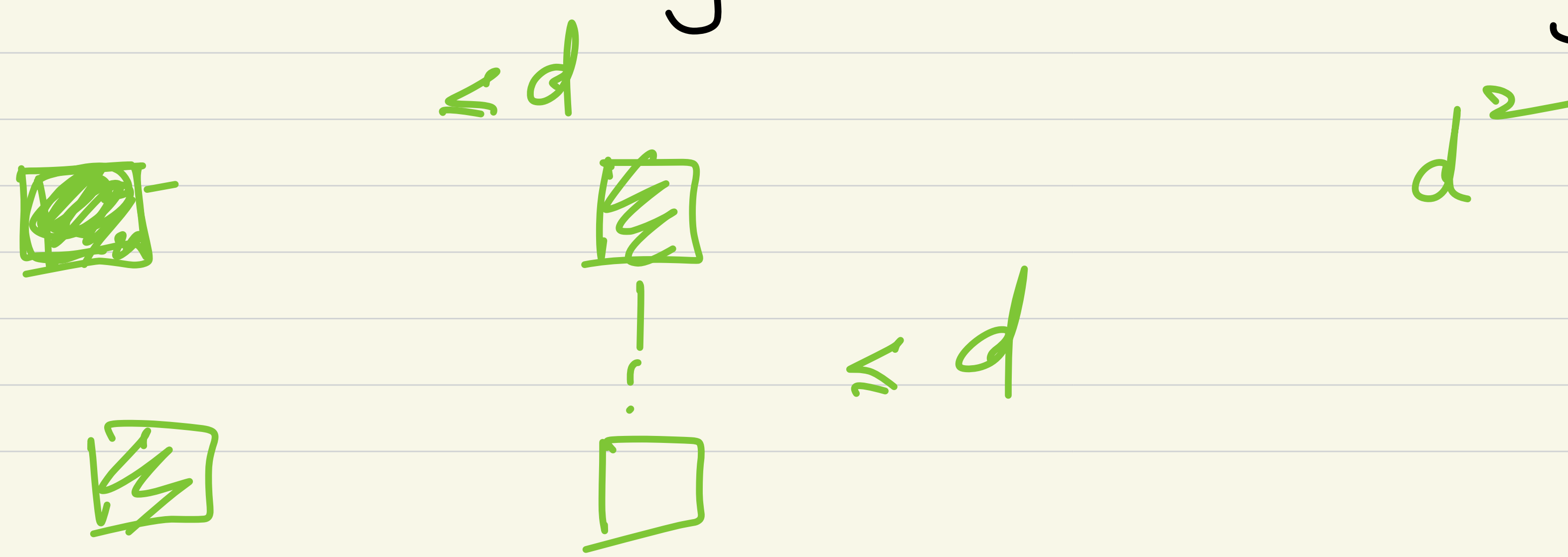
1-component configuration

Case 1a: $\Omega(\varepsilon n)$ π-appearances are present as

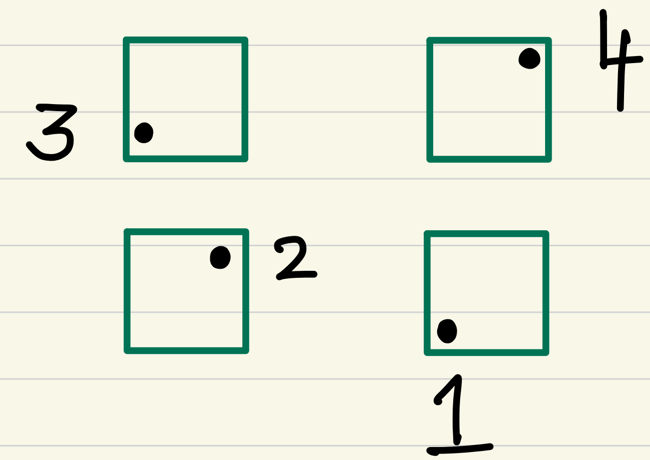


1-component configuration

- Each dense box belongs to $O(d^3)$ "copies" of such box-arrangements in grid



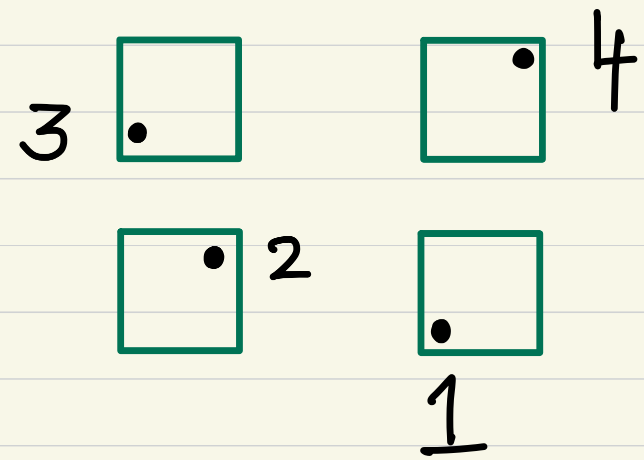
Case 1a: $\Omega(\varepsilon n)$ π-appearances are present as



1-component configuration

- Each dense box belongs to $O(d^3)$ "copies" of such box-arrangements in grid
- A random dense box participates in $\Omega(n/m)$ such appearances

Case 1a: $\Omega(\varepsilon n)$ Π-appearances are present as

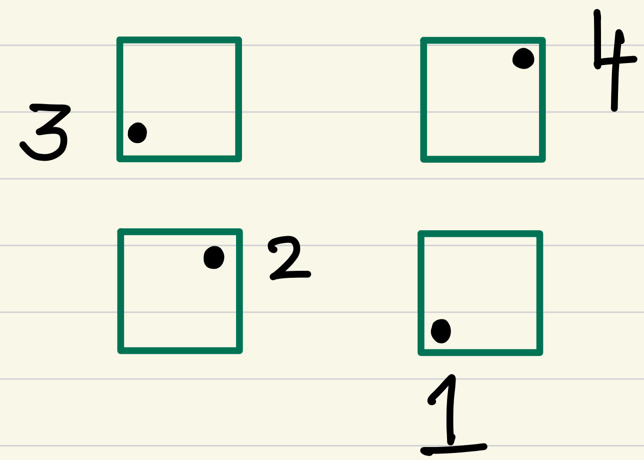


1-component configuration

- Sample a uniformly random dense box B and query all points in all copies of 1-component configurations involving B

Case 1a: $\Omega(\varepsilon n)$ π-appearances are present as

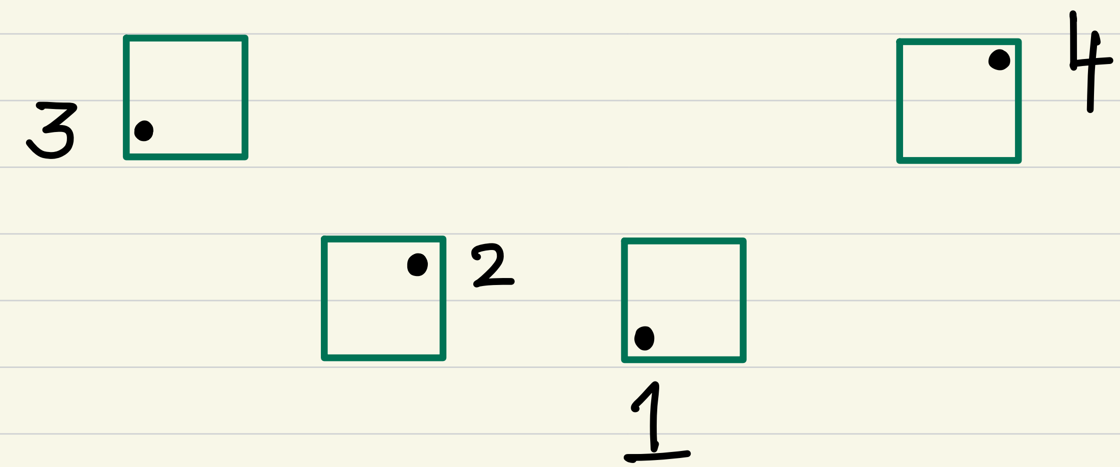
$\boxed{O(\sqrt{n}) \text{ queries}}$



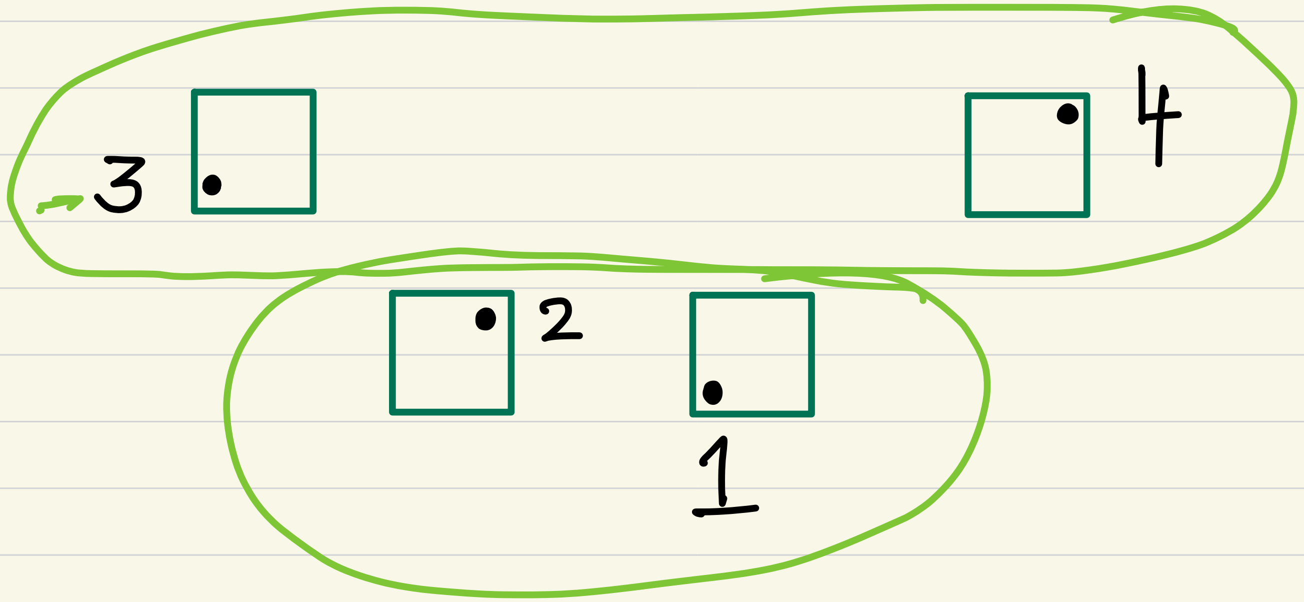
1-component configuration

- Sample a uniformly random dense box B and query all points in all copies of 1-component configurations involving B

Case 2 : $\Omega(\varepsilon n)$ π-appearances in



Case 2 : $\Omega(\varepsilon n)$ π-appearances in



1 2

- for each pair of dense boxes sharing a layer :

  ★ Test (1, 2)-freeness & (2, 1)-freeness

Solving a more general problem

- Detect a ν-appearance with a specific leg mapping, where ν is a subpattern of π

Many more ideas needed

- Reducing the complexity to $n^{o(1)}$

- Generalizing to larger patterns

Open Problems

- True complexity of testing Π-freeness

Open Problems

- True complexity of testing Π-freeness
- What about patterns of super constant length?

Open Problems

- True complexity of testing Π-freeness
- What about patterns of super constant length ? $\leftarrow$
- Approximating the distance of an array to Π-freeness $\leftarrow$

Open Problems

- True complexity of testing π-freeness
- What about patterns of super constant length?
- Approximating the distance of an array to π-freeness

Thank You!