NEW SUBLINEAR ALGORITHMS AND LOWER BOUNDS FOR LIS ESTIMATION

Ilan Newman and **Nithin Varma**
University of Haifa

אוניברסיטת חיפה
University of Haifa
جامعة حيفا

Talk outline

- LIS, distance to sortedness, and a brief history of LIS estimation

Talk outline

- LIS, distance to sortedness, and a brief history of LIS estimation
- Overview of our results

Talk outline

- LIS, distance to sortedness, and a brief history of LIS estimation
- Overview of our results
- Overview of techniques in the design of our nonadaptive LIS estimation algorithm

Talk outline

- LIS, distance to sortedness, and a brief history of LIS estimation
- Overview of our results
- Overview of techniques in the design of our nonadaptive LIS estimation algorithm
- Conclusions and open problems

Longest Increasing Subsequence (LIS)

2	4	1	6	-100	-20	-80	0	1	10	12

- LIS = Longest nondecreasing sequence of array values
- Determining length of LIS: a fundamental problem

Longest Increasing Subsequence (LIS)

2	4	1	6	-100	-20	-80	0	1	10	12

- LIS = Longest nondecreasing sequence of array values
- Determining length of LIS: a fundamental problem
- Classical **exact** algorithms running in $O(n \log n)$ time using dynamic programming [Fredman75, AldousDiaconis99]

Longest Increasing Subsequence (LIS)

2	4	1	6	-100	-20	-80	0	1	10	12

- LIS = Longest nondecreasing sequence of array values
- Determining length of LIS: a fundamental problem
- Classical **exact** algorithms running in $O(n \log n)$ time using dynamic programming [Fredman75, AldousDiaconis99]

Today: Sublinear-Time Approximation Algorithms for LIS

LIS length and distance to sortedness

- Distance of array to sortedness (monotonicity) = Minimum number of values to be removed to get a sorted subsequence

2	4	1	6	-100	-20	-80	0	1	10	12

LIS length and distance to sortedness

- Distance of array to sortedness (monotonicity) = Minimum number of values to be removed to get a sorted subsequence

2	4	1	6	-100	-20	-80	0	1	10	12

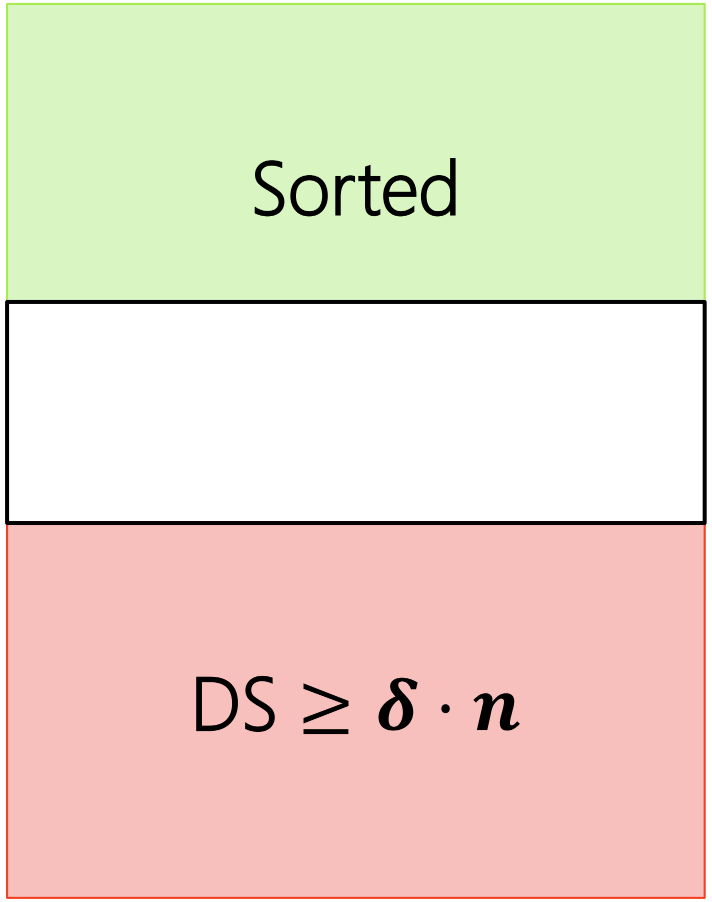
- Distance to sortedness (DS) + LIS length = Array length

$$DS + LIS = n$$

Testing sortedness

- Decision problem in the property testing model [RubinfeldSudan96, GoldreichGoldwasserRon98]

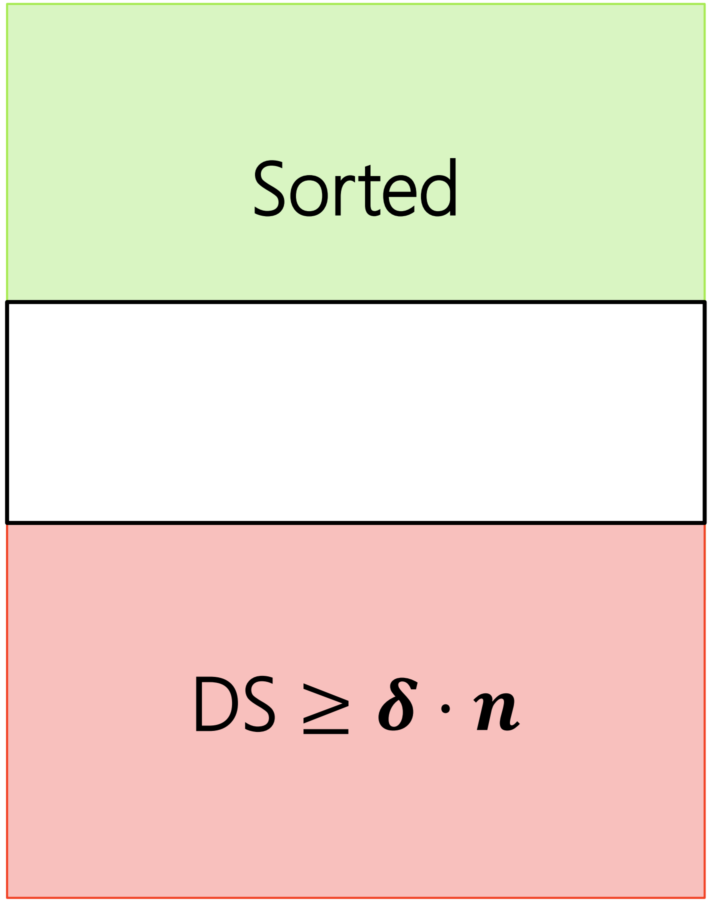
Universe of arrays



Testing sortedness

- Decision problem in the property testing model [RubinfeldSudan96, GoldreichGoldwasserRon98]
- Studied by [Ergun Kannan Kumar Rubinfeld Vishvanathan00, Fischer04, Bhattacharya Grigorescu Jung Raskhodnikova Woodruff12, Chakrabarty Seshadri13, Pallavoor Raskhodnikova Varma 18, Belovs18,..].

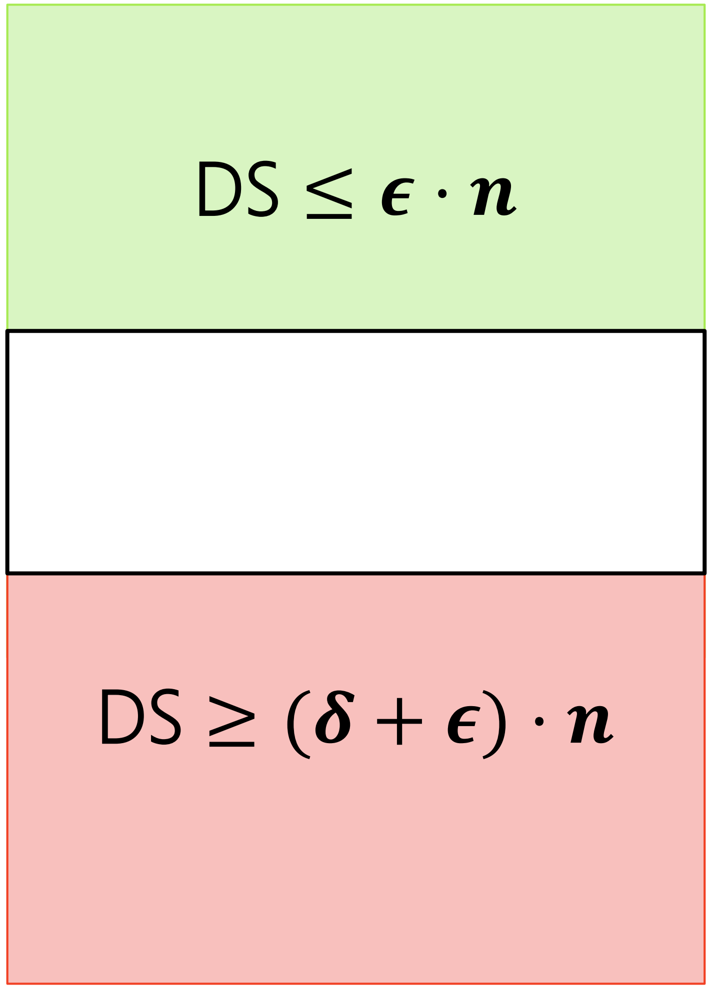
Universe of arrays



Tolerant testing sortedness

- Decision problem in the tolerant property testing model [Parnas Ron Rubinfeld 06]

Universe of arrays

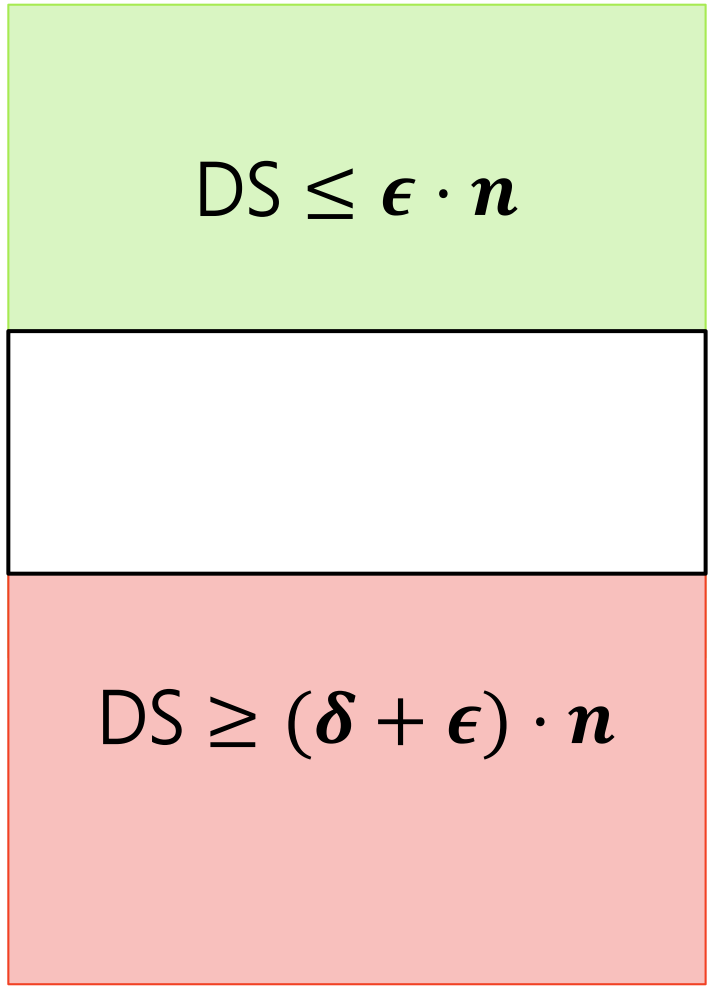


Tolerant testing sortedness

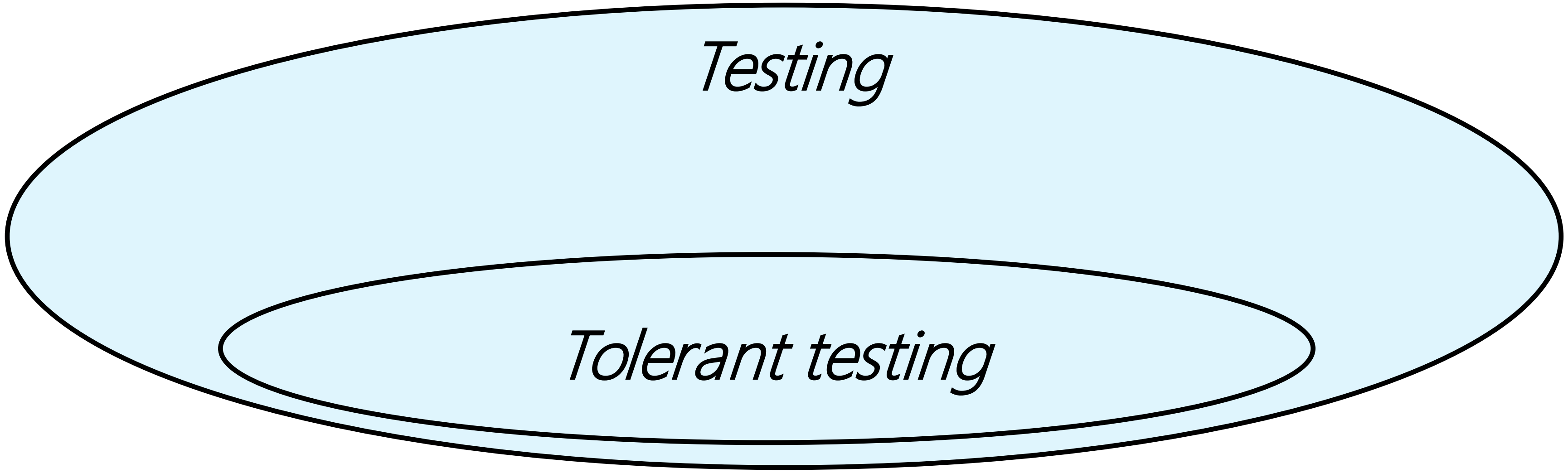
- Decision problem in the tolerant property testing model [Parnas Ron Rubinfeld 06]

Can solve this if one can estimate DS or LIS to within $\pm(\delta/2)n$

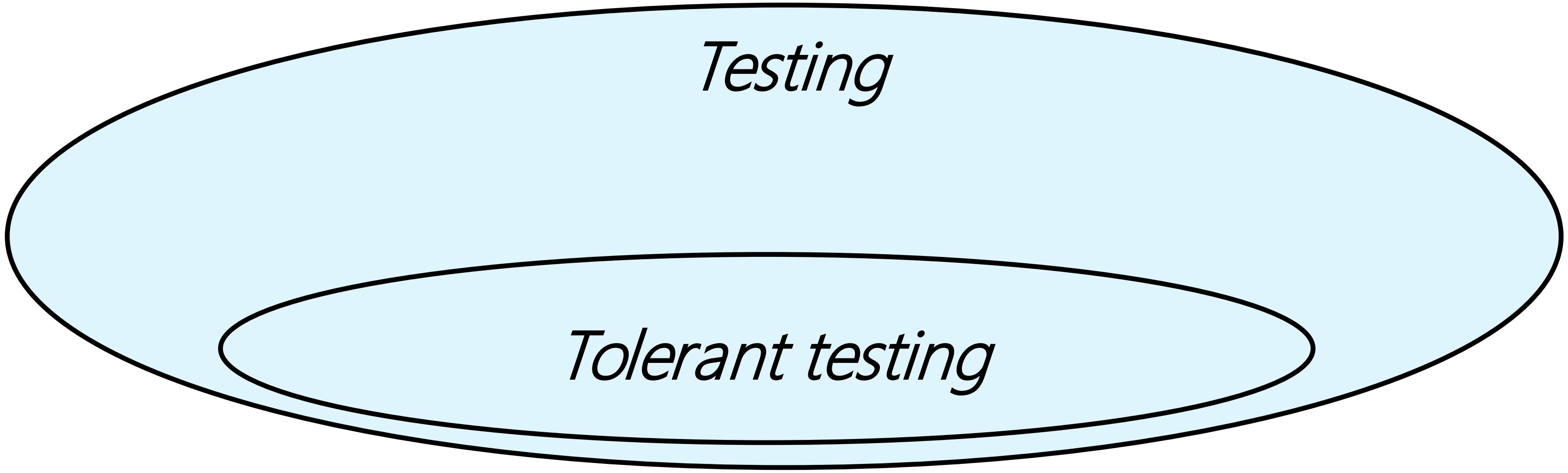
Universe of arrays



Relationship between models



Relationship between models



Lower bound on query complexity of testing is a lower bound on query complexity of additive error LIS estimation

LIS estimation algorithms

$\pm n/2$ approximation	$\log^{O(1)} n$ queries	[ParnasRonRubinfeld06, AilonChazelleComandurLiu07]

LIS estimation algorithms

$\pm n/2$ approximation	$\log^{O(1)} n$ queries	[ParnasRonRubinfeld06, AilonChazelleComandurLiu07]
$\pm \delta n$ approximation for $\delta \in (0,1)$	$\left(\frac{1}{\delta}\right)^{O\left(\frac{1}{\delta}\right)} \cdot \log^{O(1)} n$ queries	[SaksSeshadhri17]

LIS estimation algorithms

Meaningful only when LIS $\geq \delta n$

$\pm n/2$ approximation	$\log^{O(1)} n$ queries	[ParnasRonRubinfeld06, AilonChazelleComandurLiu07]
$\pm \delta n$ approximation for $\delta \in (0,1)$	$\left(\frac{1}{\delta}\right)^{O\left(\frac{1}{\delta}\right)} \cdot \log^{O(1)} n$ queries	[SaksSeshadhri17]

LIS estimation algorithms

Meaningful only when LIS $\geq \delta n$

$\pm n/2$ approximation	$\log^{O(1)} n$ queries	[ParnasRonRubinfeld06, AilonChazelleComandurLiu07]
$\pm \delta n$ approximation for $\delta \in (0,1)$	$\left(\frac{1}{\delta}\right)^{O\left(\frac{1}{\delta}\right)} \cdot \log^{O(1)} n$ queries	[SaksSeshadhri17]

- ❑ Algorithms are **adaptive**
- ❑ Query complexity not sublinear as soon as $\delta \leq 1/\log n$

LIS estimation algorithms

Let λ denote LIS$/n$

$\Omega(\lambda^3)$ multiplicative approximation	$\tilde{O}(\sqrt{n} \cdot \text{poly}(\frac{1}{\lambda}))$ **nonadaptive** queries	[RubinsteinSeddighinSongSun19]

For $t \in [0,1)$, a t multiplicative approximation is to output estimate $\in [t \cdot \text{LIS}, \text{LIS}]$

LIS estimation algorithms

Let λ denote LIS/n

$\Omega(\lambda^3)$ multiplicative approximation	$\tilde{O}(\sqrt{n} \cdot \text{poly}(\frac{1}{\lambda}))$ **nonadaptive** queries	[RubinsteinSeddighinSongSun19]
$\Omega(\lambda^\epsilon)$ multiplicative approximation for $\epsilon \in (0,1)$	$\tilde{O}(n^{1-\Omega(\epsilon)} \cdot \left(\frac{1}{\lambda}\right)^{O\left(\frac{1}{\epsilon}\right)})$ queries	[MitzenmacherSeddighin21]

For $t \in [0,1)$, a t multiplicative approximation is to output estimate $\in [t \cdot \text{LIS}, \text{LIS}]$

LIS estimation: Our focuses

- No nontrivial lower bound on the query complexity is known for LIS estimation

LIS estimation: Our focuses

- No nontrivial lower bound on the query complexity is known for LIS estimation
  - Lower bound of $\Omega(\log n)$ on the query complexity of adaptive algorithms follows from the same lower bound for testing sortedness [Fischer 04]

LIS estimation: Our focuses

- No nontrivial lower bound on the query complexity is known for LIS estimation
  - Lower bound of $\Omega(\log n)$ on the query complexity of adaptive algorithms follows from the same lower bound for testing sortedness [Fischer 04]
- Is n the right input parameter to express the complexity of LIS estimation algorithms?

LIS estimation: Our focuses

- No nontrivial lower bound on the query complexity is known for LIS estimation
  - Lower bound of $\Omega(\log n)$ on the query complexity of adaptive algorithms follows from the same lower bound for testing sortedness [Fischer 04]
- Is n the right input parameter to express the complexity of LIS estimation algorithms?
  - For Boolean arrays, $\pm\delta n$ error LIS estimation possible with $O(\frac{1}{\delta^2})$ queries [Berman Raskhodnikova Yaroslavtsev 14]

LIS estimation: Our focuses

- No nontrivial lower bound on the query complexity is known for LIS estimation
  - Lower bound of $\Omega(\log n)$ on the query complexity of adaptive algorithms follows from the same lower bound for testing sortedness [Fischer 04]
- Is n the right input parameter to express the complexity of LIS estimation algorithms?
  - For Boolean arrays, $\pm\delta n$ error LIS estimation possible with $O(\frac{1}{\delta^2})$ queries [Berman Raskhodnikova Yaroslavtsev 14]
  - For testing sortedness (and monotonicity), [Pallavoor Raskhodnikova V. 18] studied the number of distinct values r in an array and bridged a similar gap between the cases of Boolean and the real-valued arrays.

LIS estimation: Our focuses

- No nontrivial lower bound on the query complexity is known for LIS estimation
- Is n the right input parameter to express the complexity of LIS estimation algorithms?

Erasure-resilient testing sortedness

- Decision problem in the erasure-resilient testing model [Dixit Raskhodnikova Thakurta Varma 18]

Erasure-resilient testing sortedness

- Decision problem in the erasure-resilient testing model [Dixit Raskhodnikova Thakurta Varma 18]
- Some array values are erased (special symbol ⊥)

2	$\perp$	3	-10	$\perp$

Erasure-resilient testing sortedness

- Decision problem in the erasure-resilient testing model [Dixit Raskhodnikova Thakurta Varma 18]
- Some array values are erased (special symbol $\perp$)

2	$\perp$	3	-10	$\perp$

- Completion is a filling up of the erased values

2	4	3	-10	5

Erasure-resilient testing sortedness

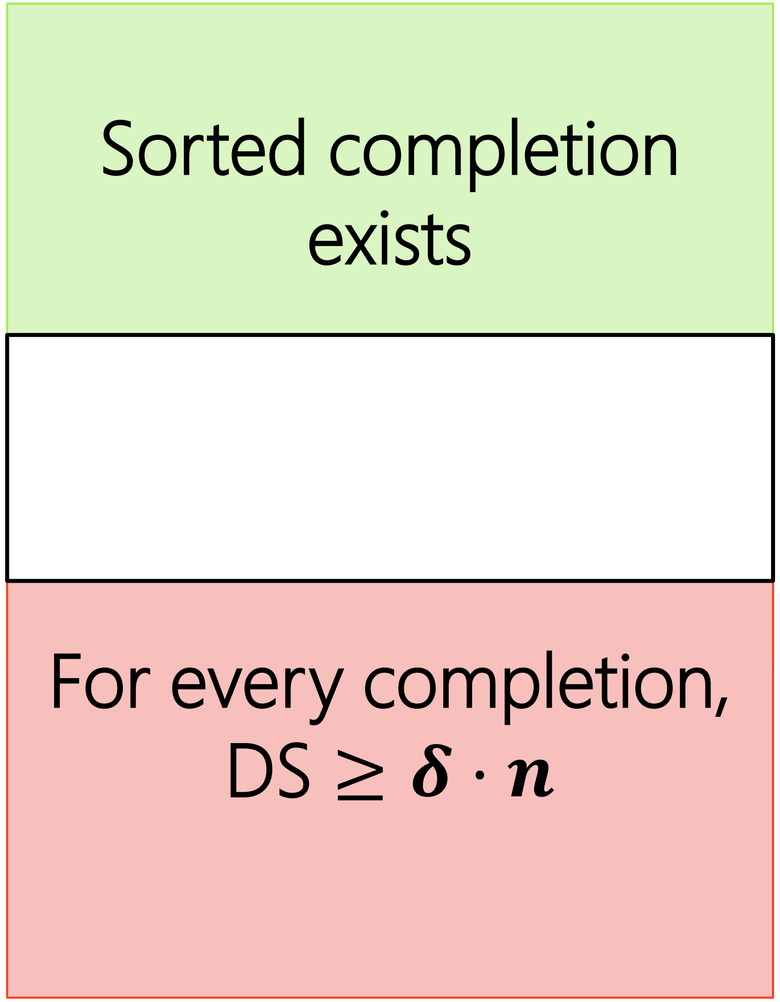
- Decision problem in the erasure-resilient testing model [Dixit Raskhodnikova Thakurta Varma 18]
- Some array values are erased (special symbol ⊥)

2	$\bot$	3	-10	$\bot$

- Completion is a filling up of the erased values

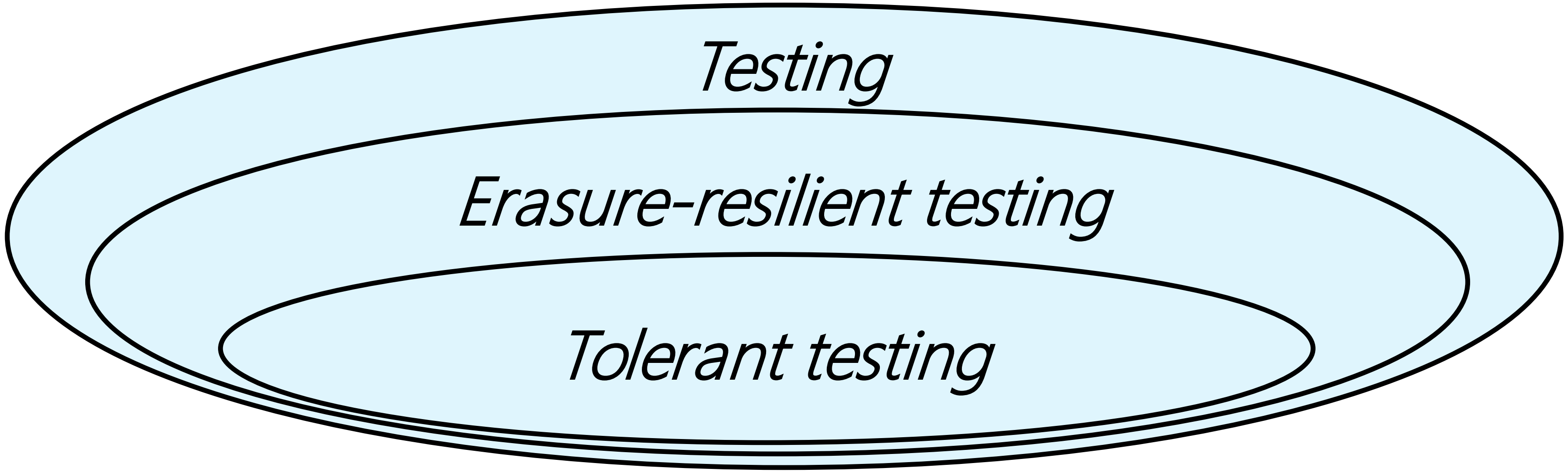
2	4	3	-10	5

Universe of arrays with or without erasures



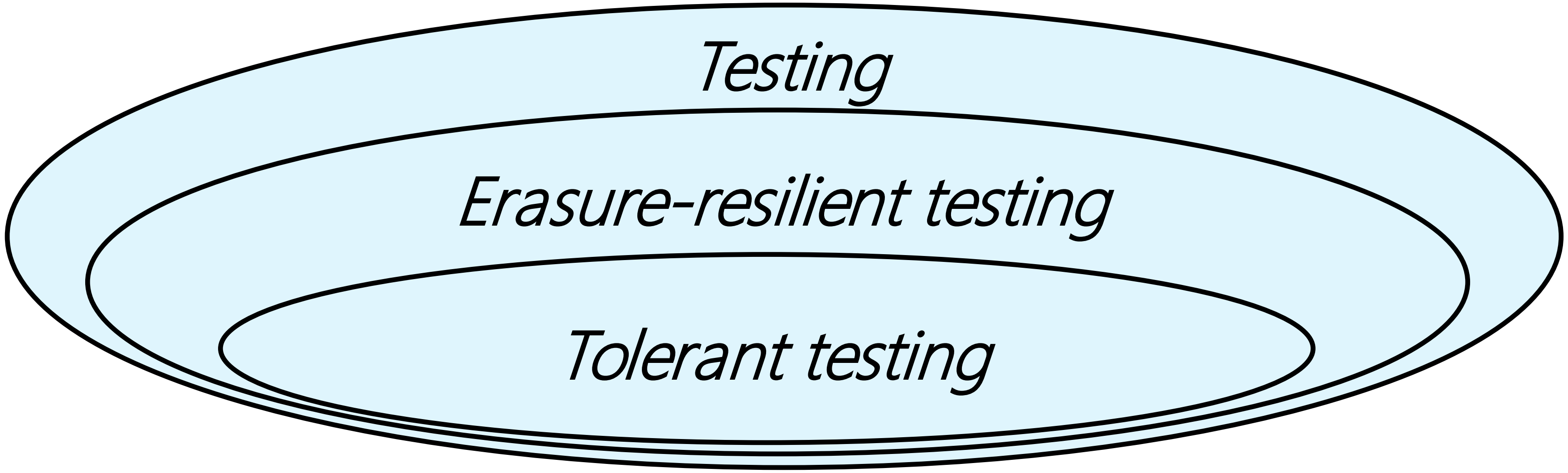
Relationship between models

[Dixit, Raskhodnikova, Thakurta & Varma '18]



Relationship between models

[Dixit, Raskhodnikova, Thakurta & Varma '18]



Lower bound on query complexity of testing, or erasure-resilient testing sortedness is a lower bound on query complexity of additive error LIS estimation

Towards LIS estimation lower bounds

Testing sortedness	$\Omega(\log n)$ queries	[Fischer04]

Towards LIS estimation lower bounds

Testing sortedness	$\Omega(\log n)$ queries	[Fischer04]
Erasure-resilient testing sortedness	$O(\log n)$ adaptive queries	[Dixit Raskhodnikova Thakurta Varma 18]

Towards LIS estimation lower bounds

Testing sortedness	$\Omega(\log n)$ queries	[Fischer04]
Erasure-resilient testing sortedness	$O(\log n)$ adaptive queries	[Dixit Raskhodnikova Thakurta Varma 18]
Erasure-resilient testing sortedness	$O(\log n)$ nonadaptive queries	[our work]

Towards LIS estimation lower bounds

Testing sortedness	$\Omega(\log n)$ queries	[Fischer04]
Erasure-resilient testing sortedness	$O(\log n)$ adaptive queries	[Dixit Raskhodnikova Thakurta Varma 18]
Erasure-resilient testing sortedness	**$O(\log n)$ nonadaptive queries**	[our work]

The only lower bound on query complexity LIS estimation we can get this way is $\Omega(\log n)$

LIS estimation lower bound: Our result

LIS estimation up to error $\pm\delta n$	$\log^{\Omega(\log \frac{1}{\delta})} n$ **nonadaptive** queries [our work]

LIS estimation lower bound: Our result

LIS estimation up to error $\pm\delta n$	$\log^{\Omega(\log \frac{1}{\delta})} n$ **nonadaptive** queries [our work]

- ❑ **No polylog-query nonadaptive algorithm** for LIS estimation up to additive error $\pm\delta n$ that works for all constant $\delta \in [0,1]$

LIS estimation lower bound: Our result

LIS estimation up to error $\pm\delta n$	$\log^{\Omega(\log \frac{1}{\delta})} n$ **nonadaptive** queries [our work]

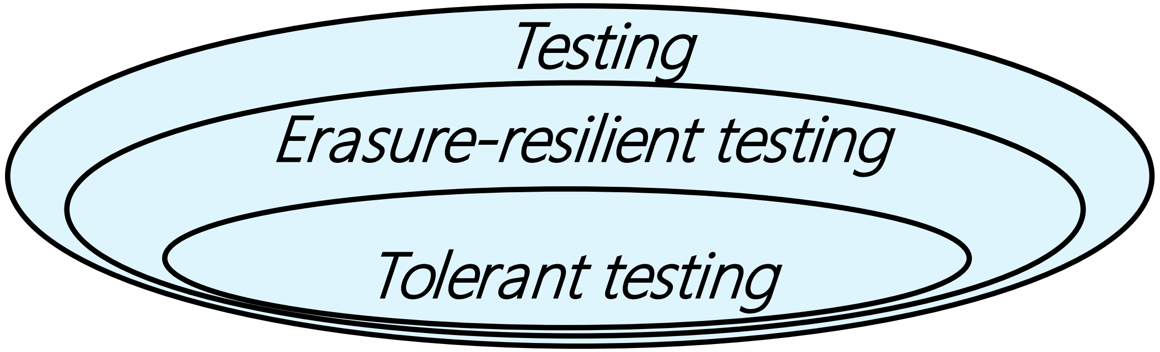
- ❑ **No polylog-query nonadaptive algorithm** for LIS estimation up to additive error $\pm\delta n$ that works for all constant $\delta \in [0,1]$
- ❑ First nontrivial lower bound on LIS estimation, or tolerant testing sortedness

LIS estimation lower bound: Our result

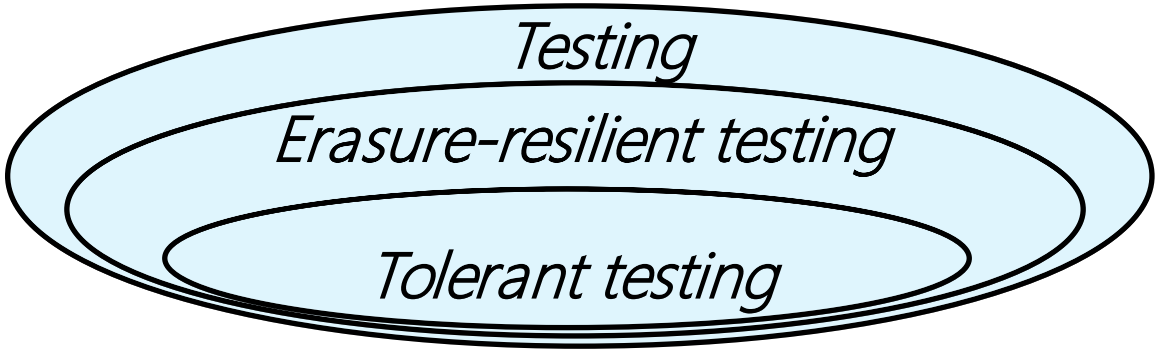
LIS estimation up to error $\pm\delta n$	$\log^{\Omega(\log\frac{1}{\delta})} n$ **nonadaptive** queries	[our work]
	$\left(\frac{1}{\delta}\right)^{O\left(\frac{1}{\delta}\right)} \cdot \log^{O(1)} n$ queries	[SaksSeshadhri17]

- ❑ **No polylog-query nonadaptive algorithm** for LIS estimation up to additive error $\pm\delta n$ that works for all constant $\delta \in [0,1]$
- ❑ First nontrivial lower bound on LIS estimation, or tolerant testing sortedness
- ❑ Adaptivity helps in tolerant testing sortedness

LIS estimation lower bound: Consequence

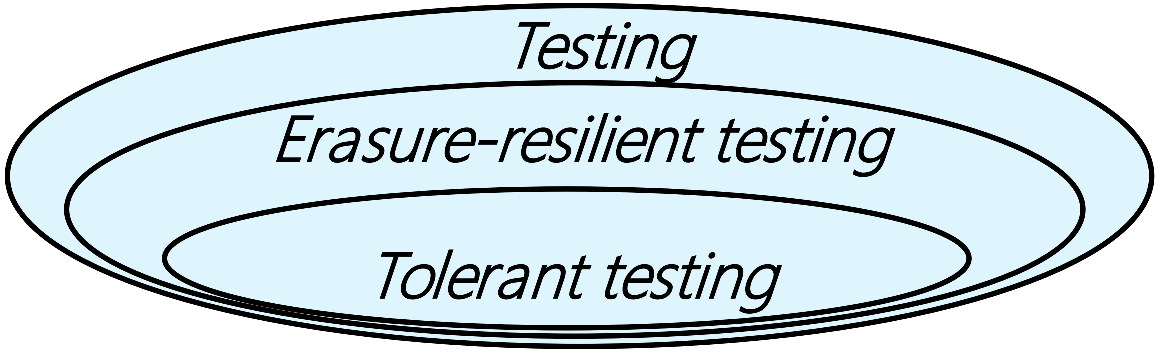


LIS estimation lower bound: Consequence



These containments are strict [Fischer Fortnow 06], [Dixit Raskhodnikova Thakurta Varma 18], [Raskhodnikova Ron-Zewi Varma 19].

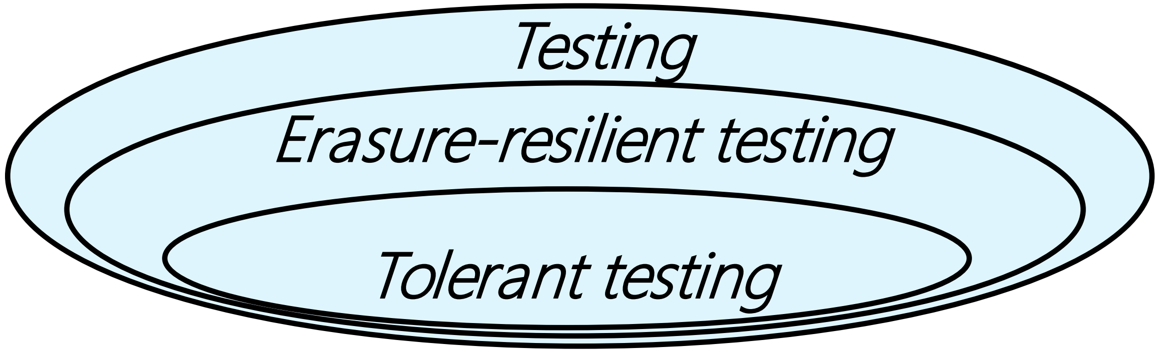
LIS estimation lower bound: Consequence



These containments are strict [Fischer Fortnow 06], [Dixit Raskhodnikova Thakurta Varma 18], [Raskhodnikova Ron-Zewi Varma 19].

Open Question [RRV19]: Are there natural properties for which such strict separations exist?

LIS estimation lower bound: Consequence

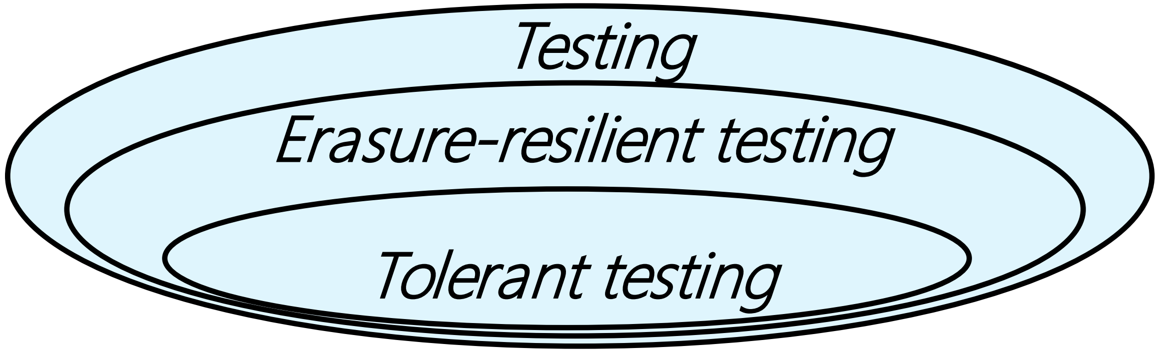


These containments are strict [Fischer Fortnow 06], [Dixit Raskhodnikova Thakurta Varma 18], [Raskhodnikova Ron-Zewi Varma 19].

Open Question [RRV19]: Are there natural properties for which such strict separations exist?

Testing vs. Tolerant testing	Unateness of $f:\{0,1\}^d \rightarrow \{0,1\}$ [Levi Waingarten 19]

LIS estimation lower bound: Consequence

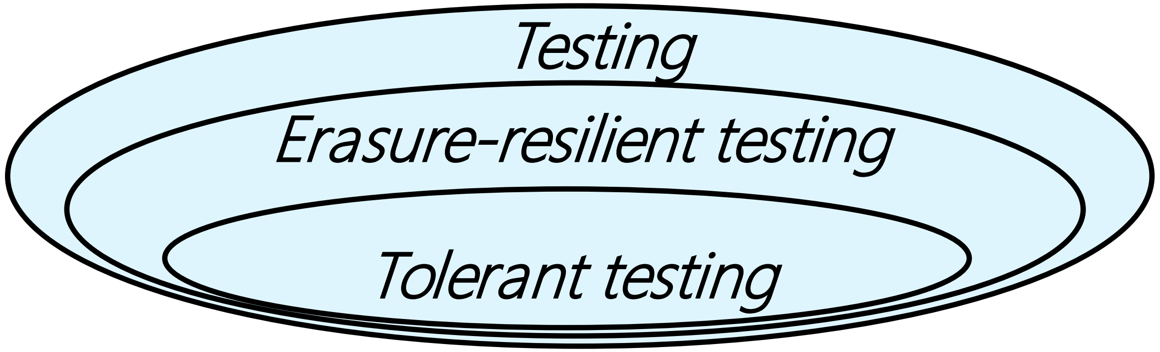


These containments are strict [Fischer Fortnow 06], [Dixit Raskhodnikova Thakurta Varma 18], [Raskhodnikova Ron-Zewi Varma 19].

Open Question [RRV19]: Are there natural properties for which such strict separations exist?

Testing vs. Tolerant testing	Unateness of $f: \{0,1\}^d \to \{0,1\}$ [Levi Waingarten 19]
Testing vs. Erasure-resilient testing	Monotonicity of $f: \{0,1\}^d \to \{0,1\}$; for nonadaptive algorithms [Pallavoor Raskhodnikova Waingarten 20]

LIS estimation lower bound: Consequence



These containments are strict [Fischer Fortnow 06], [Dixit Raskhodnikova Thakurta Varma 18], [Raskhodnikova Ron-Zewi Varma 19].

Open Question [RRV19]: Are there natural properties for which such strict separations exist?

Testing vs. Tolerant testing	Unateness of $f: \{0,1\}^d \to \{0,1\}$ [Levi Waingarten 19]
Testing vs. Erasure-resilient testing	Monotonicity of $f: \{0,1\}^d \to \{0,1\}$; for nonadaptive algorithms [Pallavoor Raskhodnikova Waingarten 20]
Erasure-resilient testing vs. Tolerant testing	Sortedness; for nonadaptive algorithms [our work]

LIS estimation algorithms: Our results

Let $r \leq n$ denote the number of distinct values in the input array

$\pm\delta n$ approximation for $\delta \in (0,1)$	$\tilde{O}(\frac{r}{\delta^3})$ uniform queries [our work]

LIS estimation algorithms: Our results

Let $r \leq n$ denote the number of distinct values in the input array

$\pm\delta n$ approximation for $\delta \in (0,1)$	$\tilde{O}(\frac{r}{\delta^3})$ uniform queries	[our work]
	$\left(\frac{1}{\delta}\right)^{O\left(\frac{1}{\delta}\right)} \cdot \log^{O(1)} n$ queries	[SaksSeshadhri17]

LIS estimation algorithms: Our results

Let $r \leq n$ denote the number of distinct values in the input array

$\pm\delta n$ approximation for $\delta \in (0,1)$	$\tilde{O}(\frac{r}{\delta^3})$ uniform queries	[our work]
	$\left(\frac{1}{\delta}\right)^{O\left(\frac{1}{\delta}\right)} \cdot \log^{O(1)} n$ queries	[SaksSeshadhri17]

- Only truly sublinear time algorithm with this approximation guarantee when $\delta \ll \frac{1}{\log n}$ (holds when $r = o(n)$)

LIS estimation algorithms: Our results

Let $r \leq n$ denote the number of distinct values in the input array

$\pm\delta n$ approximation for $\delta \in (0,1)$	$\tilde{O}(\frac{r}{\delta^3})$ uniform queries	[our work]
	$\left(\frac{1}{\delta}\right)^{O\left(\frac{1}{\delta}\right)} \cdot \log^{O(1)} n$ queries	[SaksSeshadhri17]

- ❑ Only truly sublinear time algorithm with this approximation guarantee when $\delta \ll \frac{1}{\log n}$ (holds when $r = o(n)$)
- ❑ Moreover, our algorithm is nonadaptive and makes uniform queries

LIS estimation algorithms: Our results

Let $r \leq n$ denote the number of distinct values in the input array

$\Omega(\lambda)$ multiplicative approximation, where $\lambda = \text{LIS}/n$	$\tilde{O}(\sqrt{r} \cdot \text{poly}(\frac{1}{\lambda}))$ **nonadaptive** queries	[our work]

LIS estimation algorithms: Our results

Let $r \le n$ denote the number of distinct values in the input array

$\Omega(\lambda)$ multiplicative approximation, where $\lambda = \text{LIS}/n$	$\tilde{O}(\sqrt{r} \cdot \text{poly}(\frac{1}{\lambda}))$ nonadaptive queries	[our work]
$\Omega(\lambda^3)$ multiplicative approximation, where $\lambda = \text{LIS}/n$	$\tilde{O}(\sqrt{n} \cdot \text{poly}(\frac{1}{\lambda}))$ nonadaptive queries	[RubinsteinSeddighinSongSun19]

- ❑ Better approximation guarantee than the $\Omega(\lambda^3)$ guarantee
- ❑ Better running time when $r \ll n$

Nonadaptive algorithm for LIS estimation

- **Problem:** Given access to array A with at most r distinct values, output an estimate est such that

$$\Omega(\lambda \cdot \text{LIS}) \leq \text{est} \leq O(\text{LIS}),$$

where $\lambda = \text{LIS}/n$

Nonadaptive algorithm for LIS estimation

- **Problem:** Given access to array A with at most r distinct values, output an estimate est such that

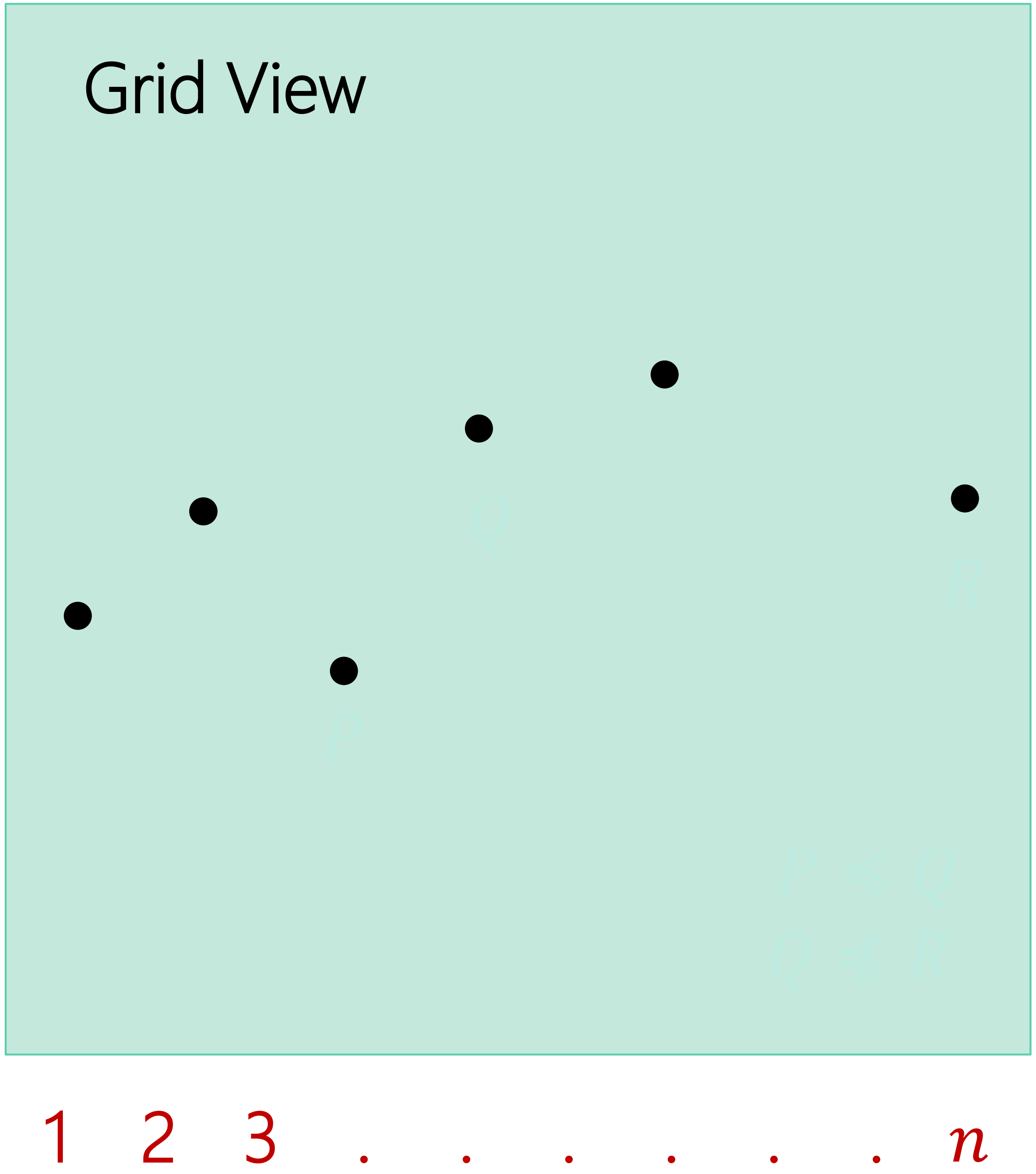
$$\Omega(\lambda \cdot \text{LIS}) \leq \text{est} \leq O(\text{LIS}),$$

  where $\lambda = \text{LIS}/n$

- **Simplifications:**
  - We will describe a $\tilde{O}(\sqrt{n} \cdot \text{poly}(\frac{1}{\lambda}))$ query algorithm
  - No value occurs more than $\sqrt{n}$ times
  - We know a lower bound on λ

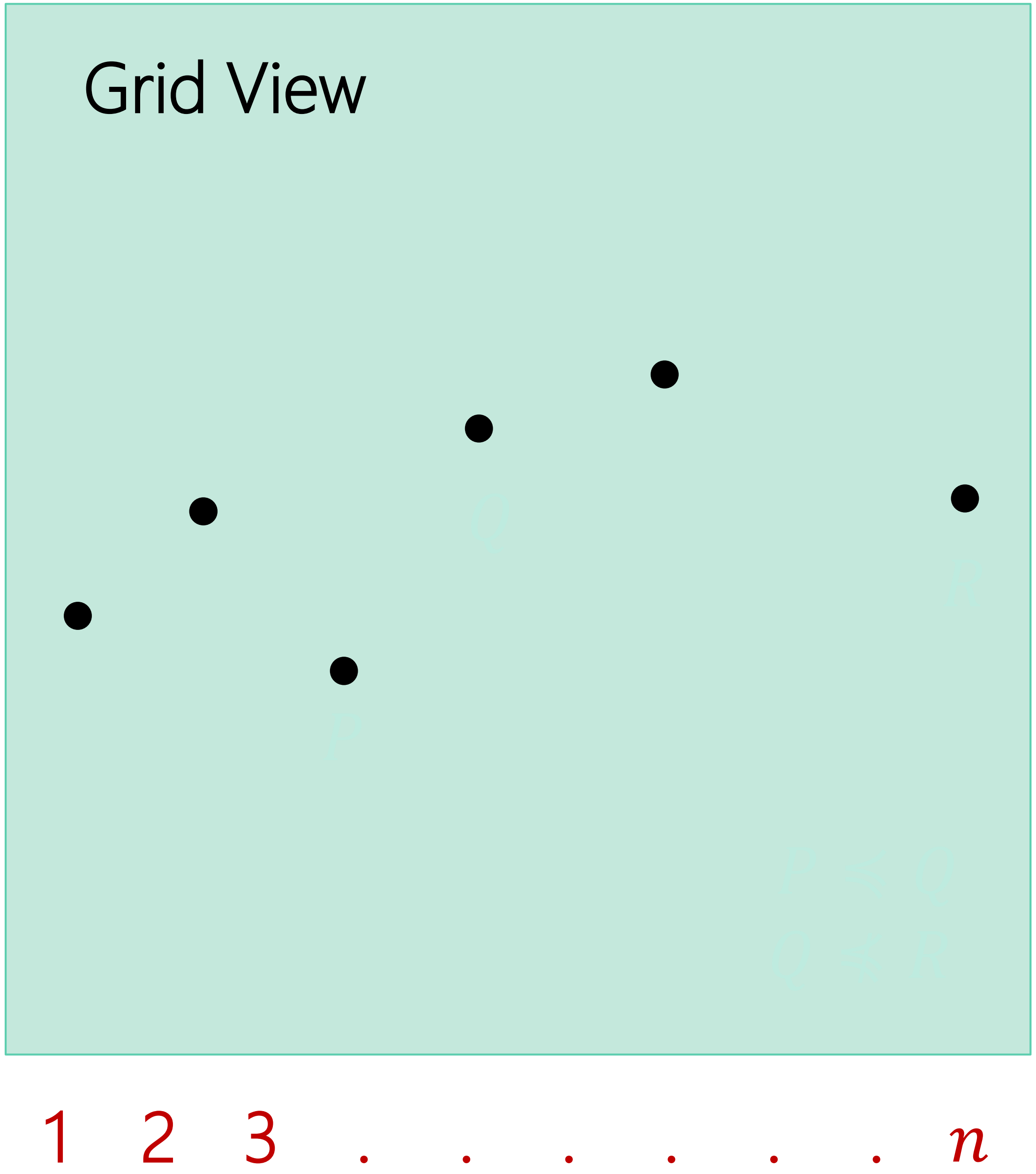
Nonadaptive algorithm

- Array of length n containing at most r distinct values



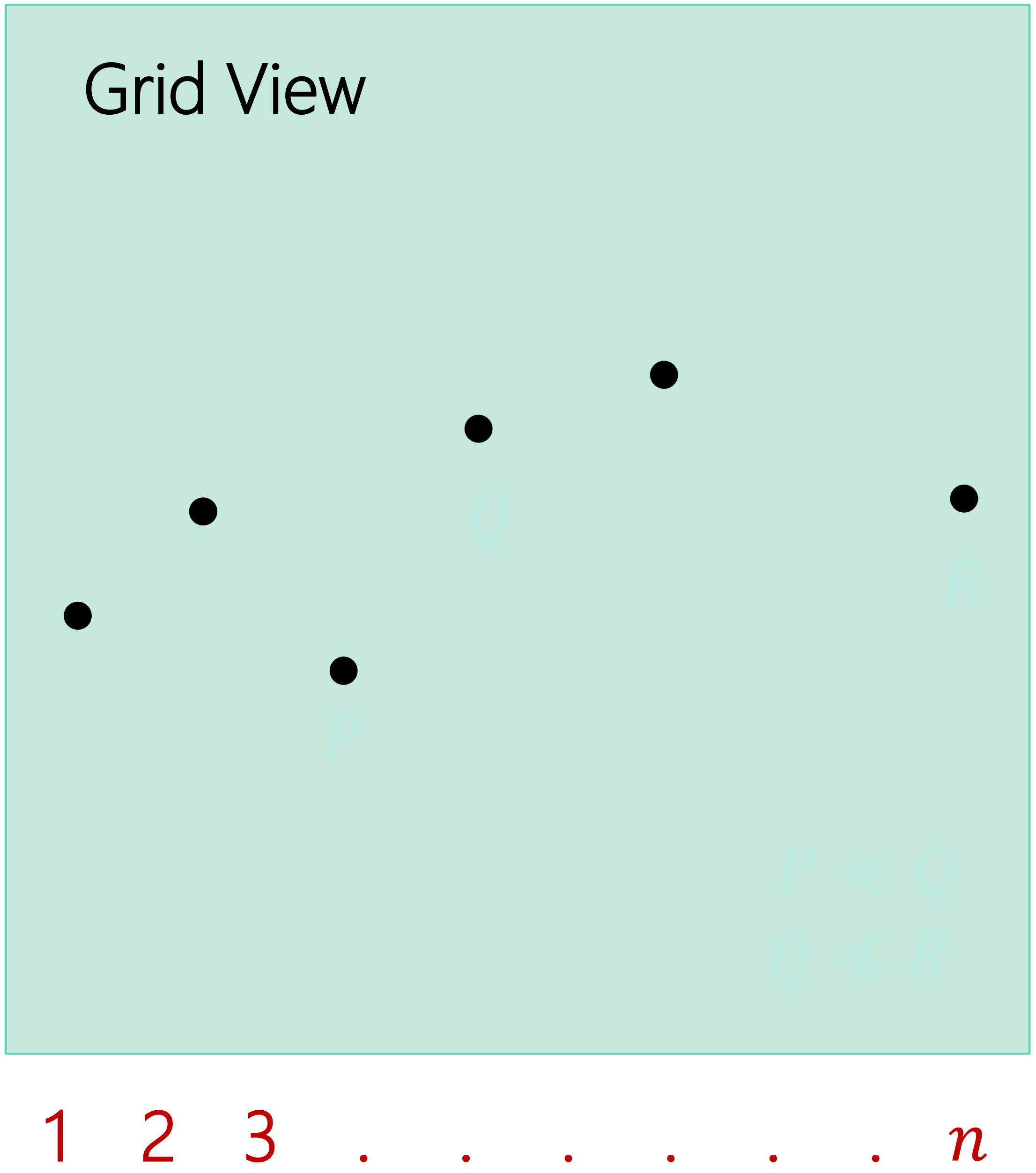
Nonadaptive algorithm

- Array of length n containing at most r distinct values
- Let λ be a known lower bound on LIS/n



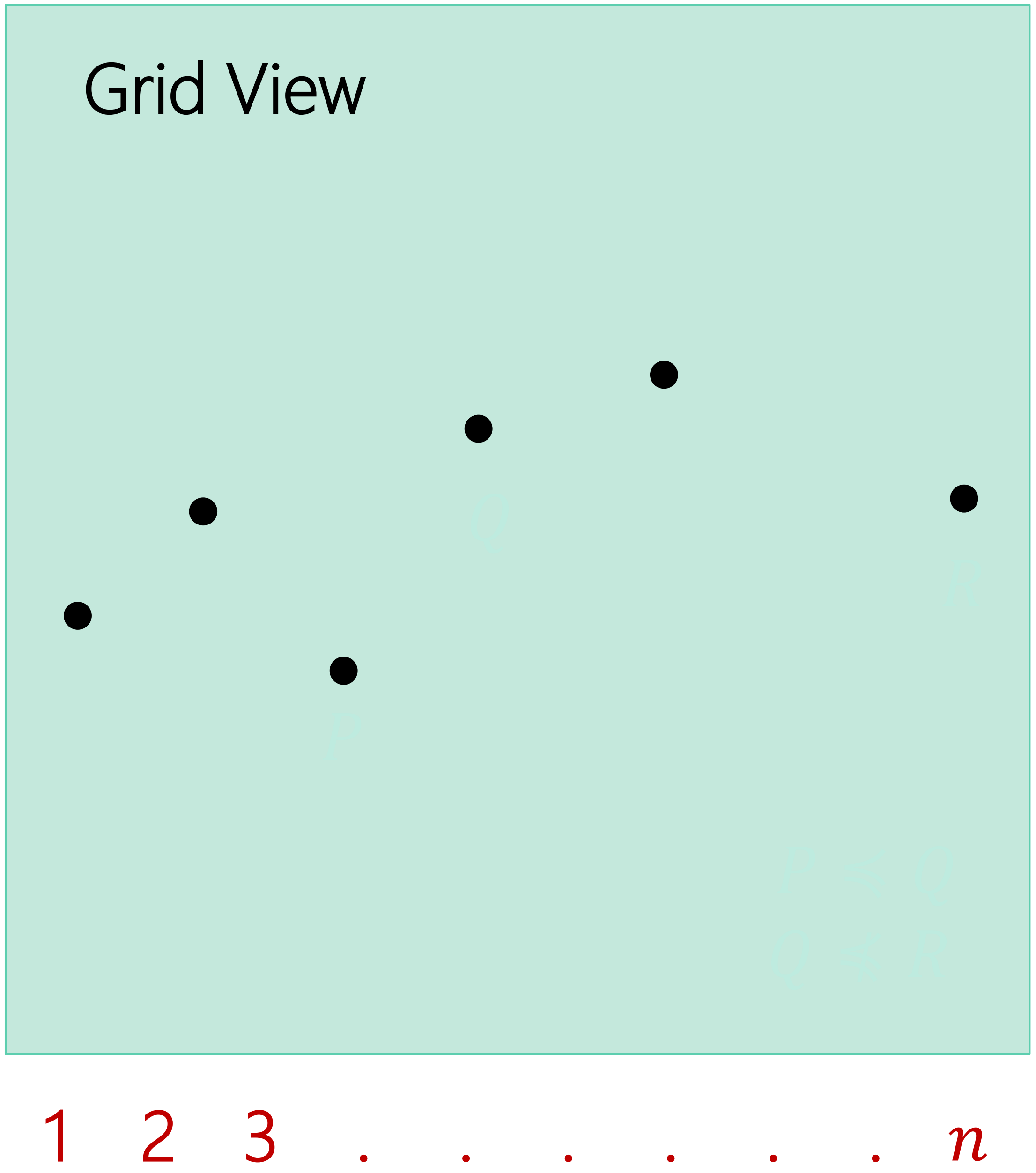
Nonadaptive algorithm

- Array of length n containing at most r distinct values
- Let λ be a known lower bound on LIS/n
- Visualize as a grid of (index, value) pairs



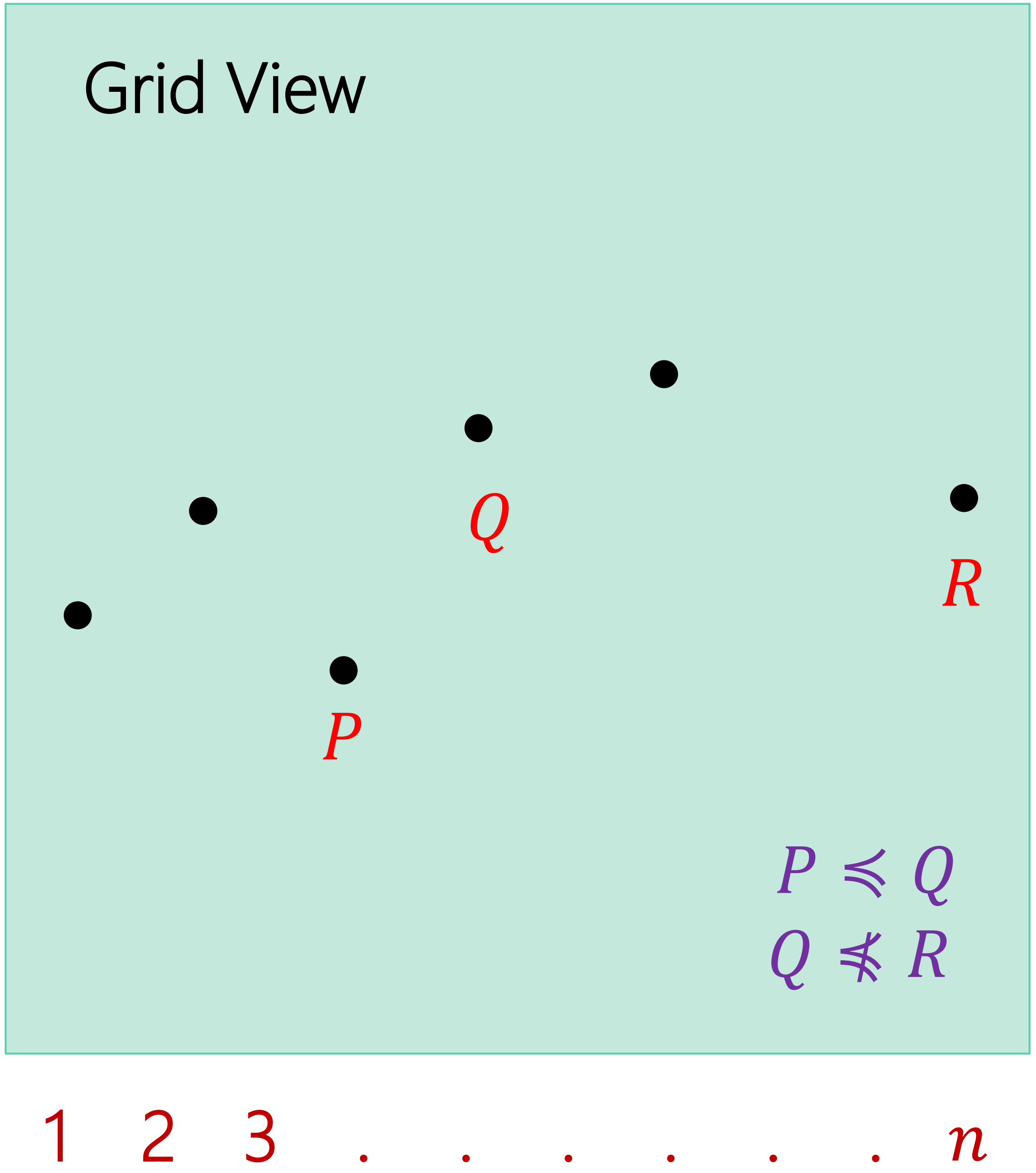
Nonadaptive algorithm

- Array of length n containing at most r distinct values
- Let λ be a known lower bound on LIS/n
- Visualize as a grid of (index, value) pairs
- Natural poset on points



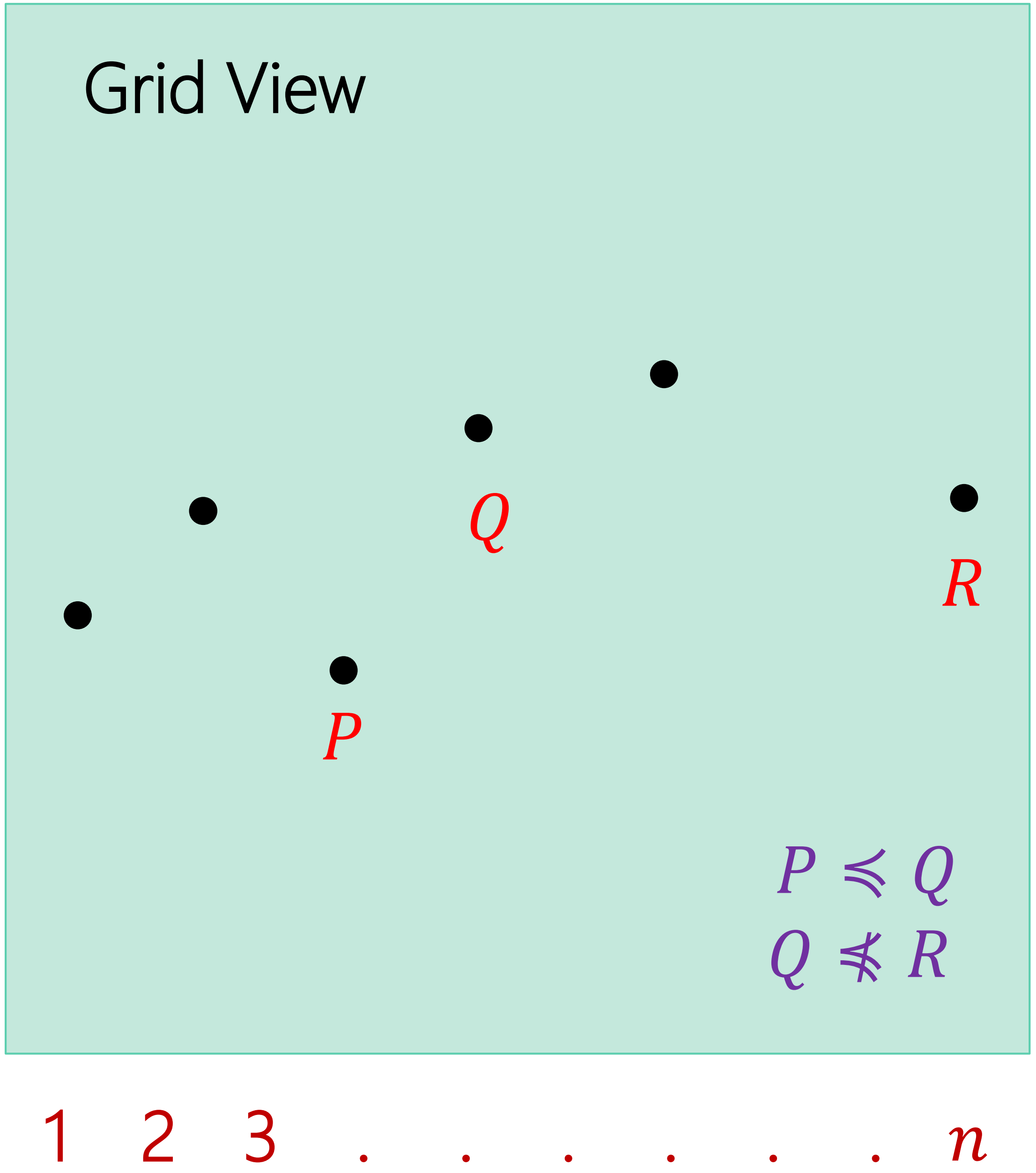
Nonadaptive algorithm

- Array of length n containing at most r distinct values
- Let λ be a known lower bound on LIS/n
- Visualize as a grid of (index, value) pairs
- Natural poset on points



Nonadaptive algorithm

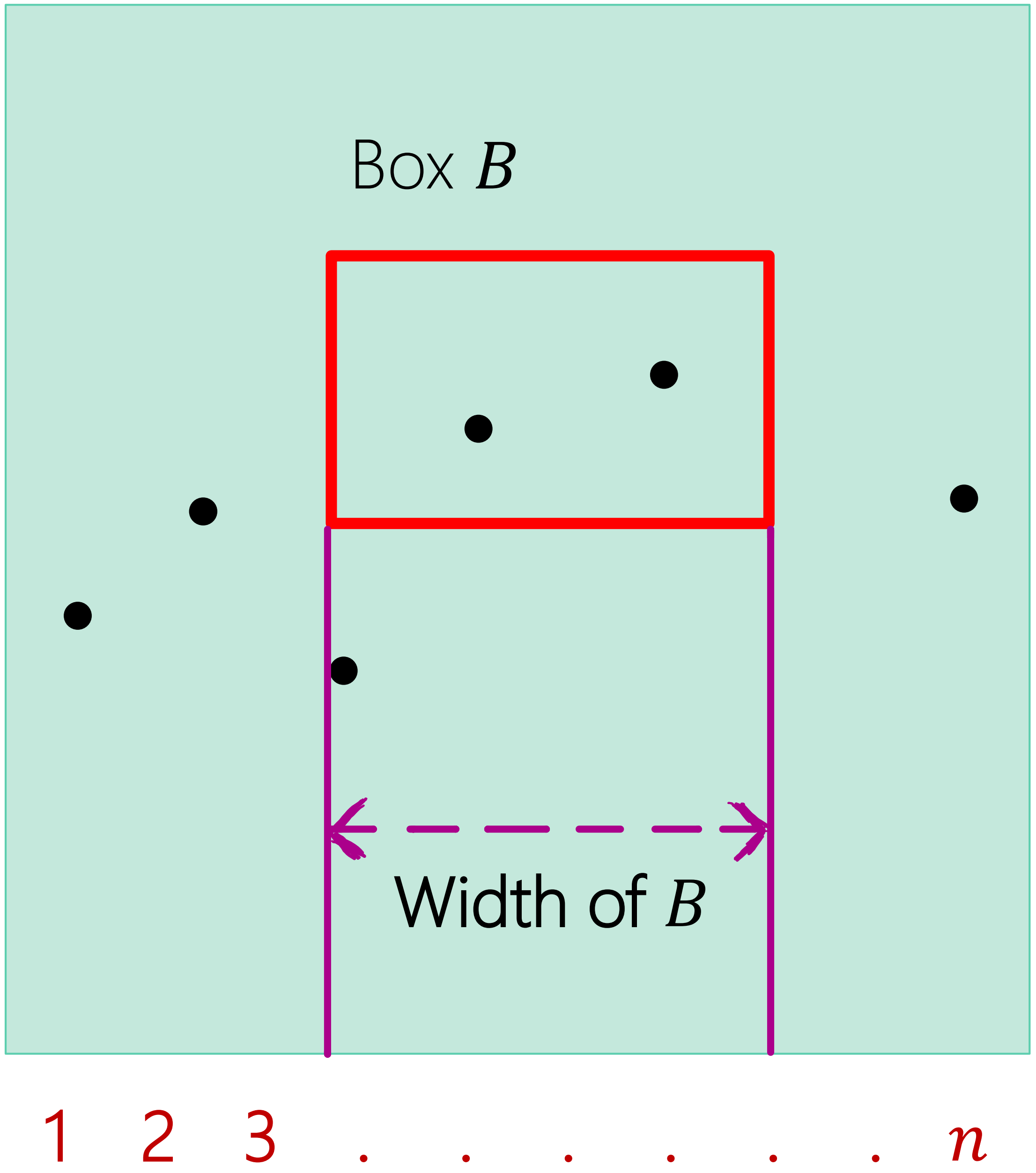
- Array of length n containing at most r distinct values
- Let λ be a known lower bound on LIS/n
- Visualize as a grid of (index, value) pairs
- Natural poset on points
- LIS = Longest chain in the poset of points



Density of boxes

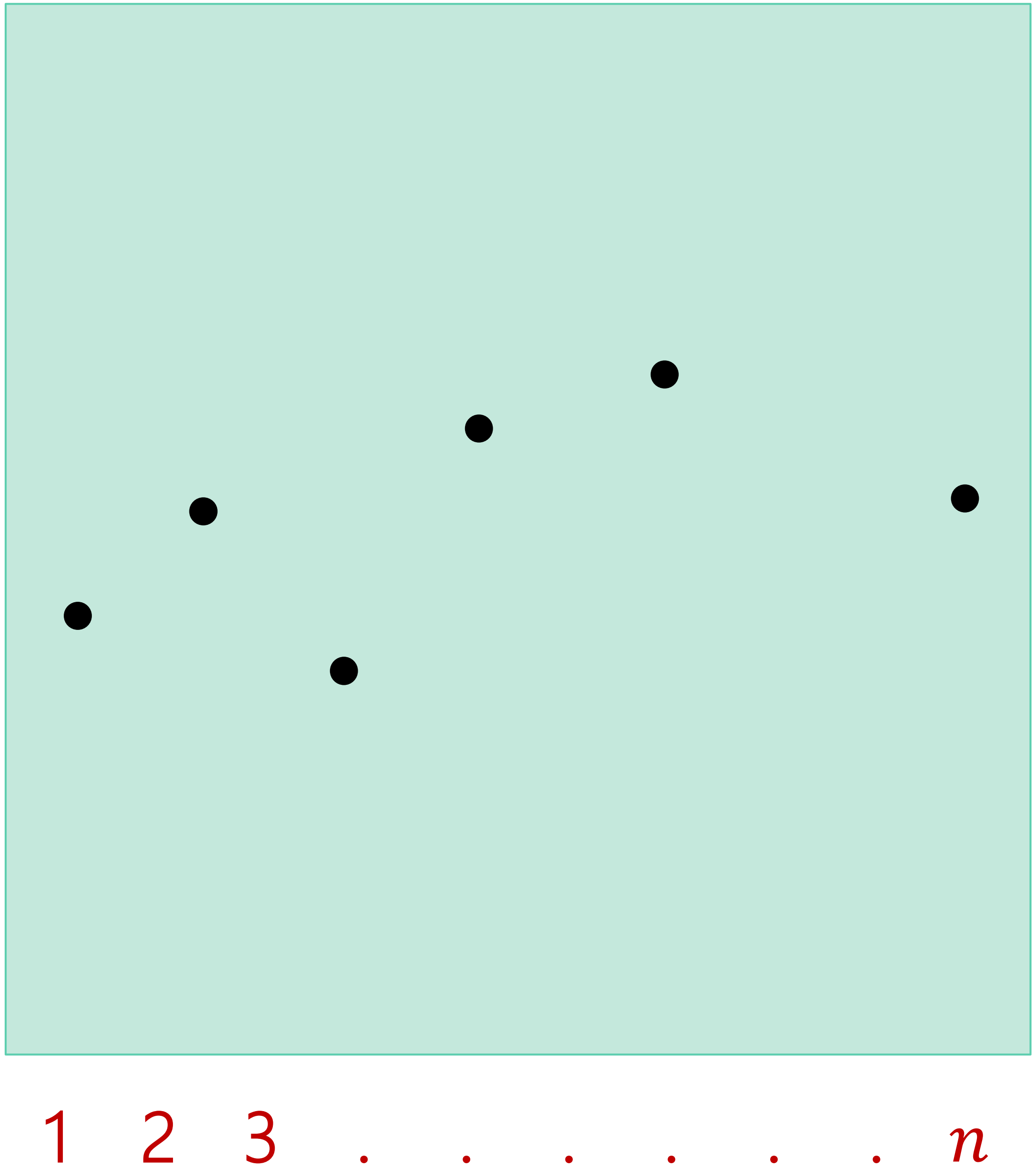
- Density of B:

$$\frac{\text{No. of points in } B}{\text{Width of } B}$$



Layering

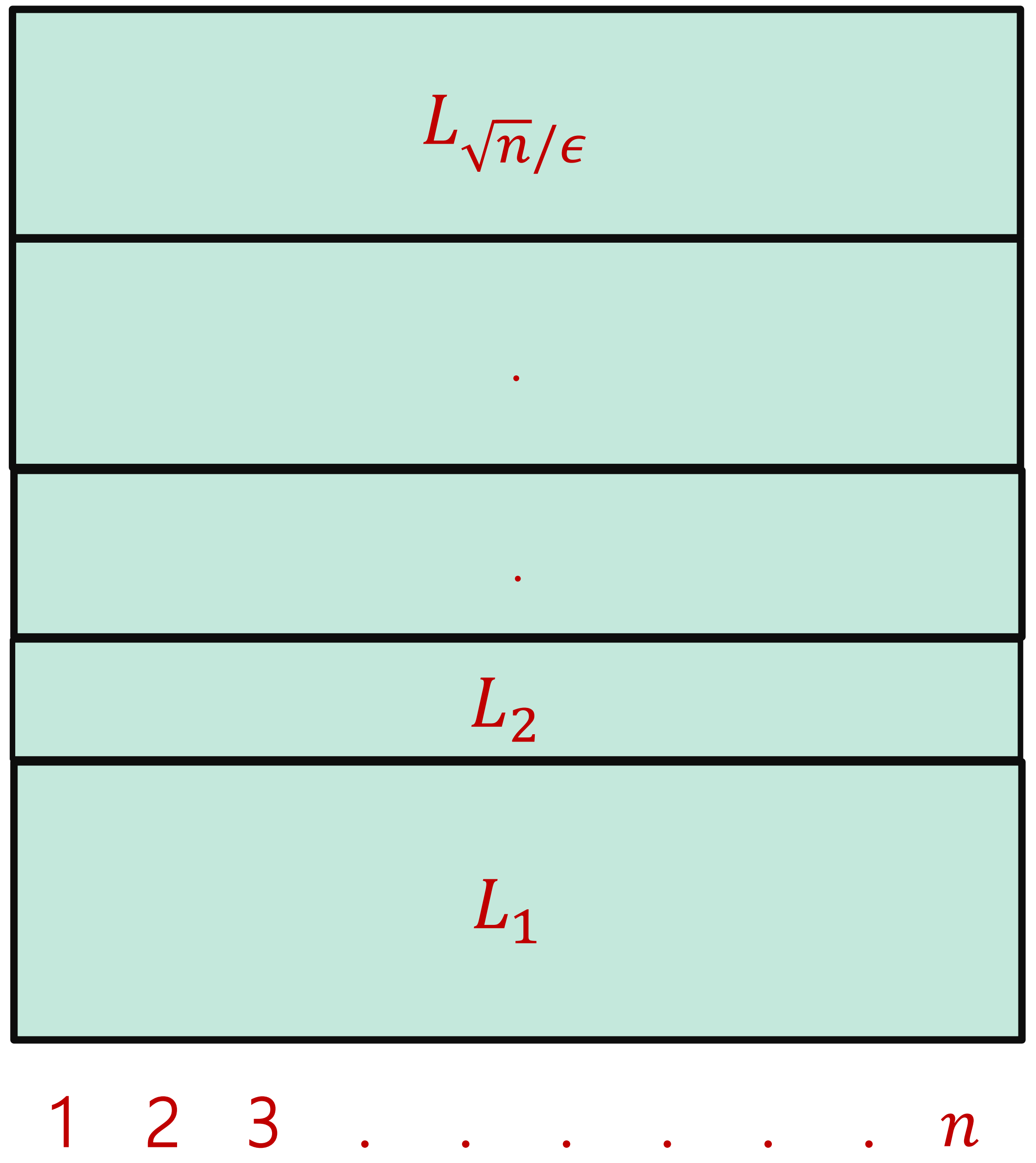
Let $\epsilon \in (0,1)$ be a parameter



Layering

Let $\epsilon \in (0,1)$ be a parameter

Sample $\tilde{\Theta}\left(\sqrt{n}\right)$ points uniformly at random and equipartition the range into roughly $\sqrt{n}/\epsilon$ intervals of equal density

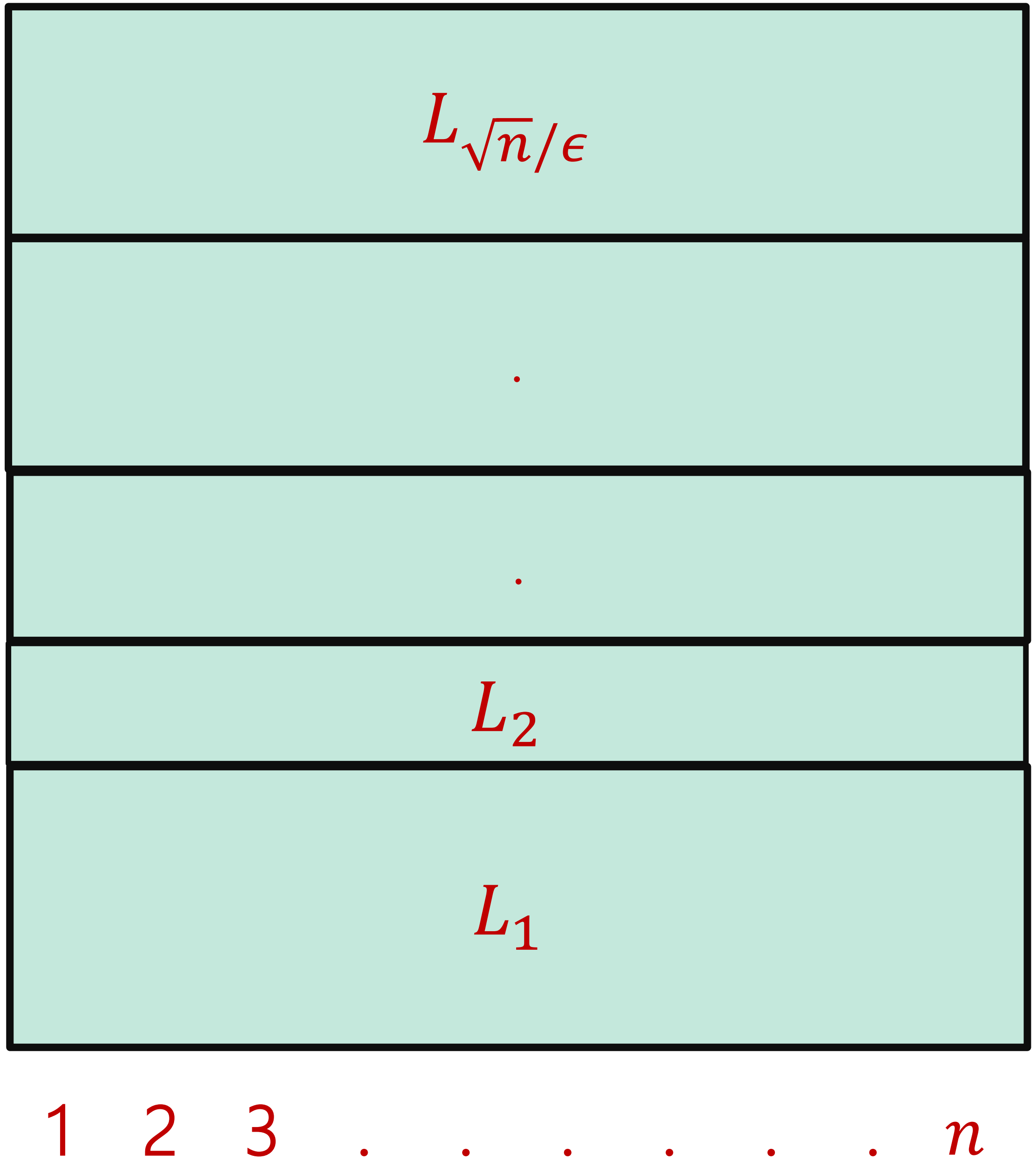


Layering

Let $\epsilon \in (0,1)$ be a parameter

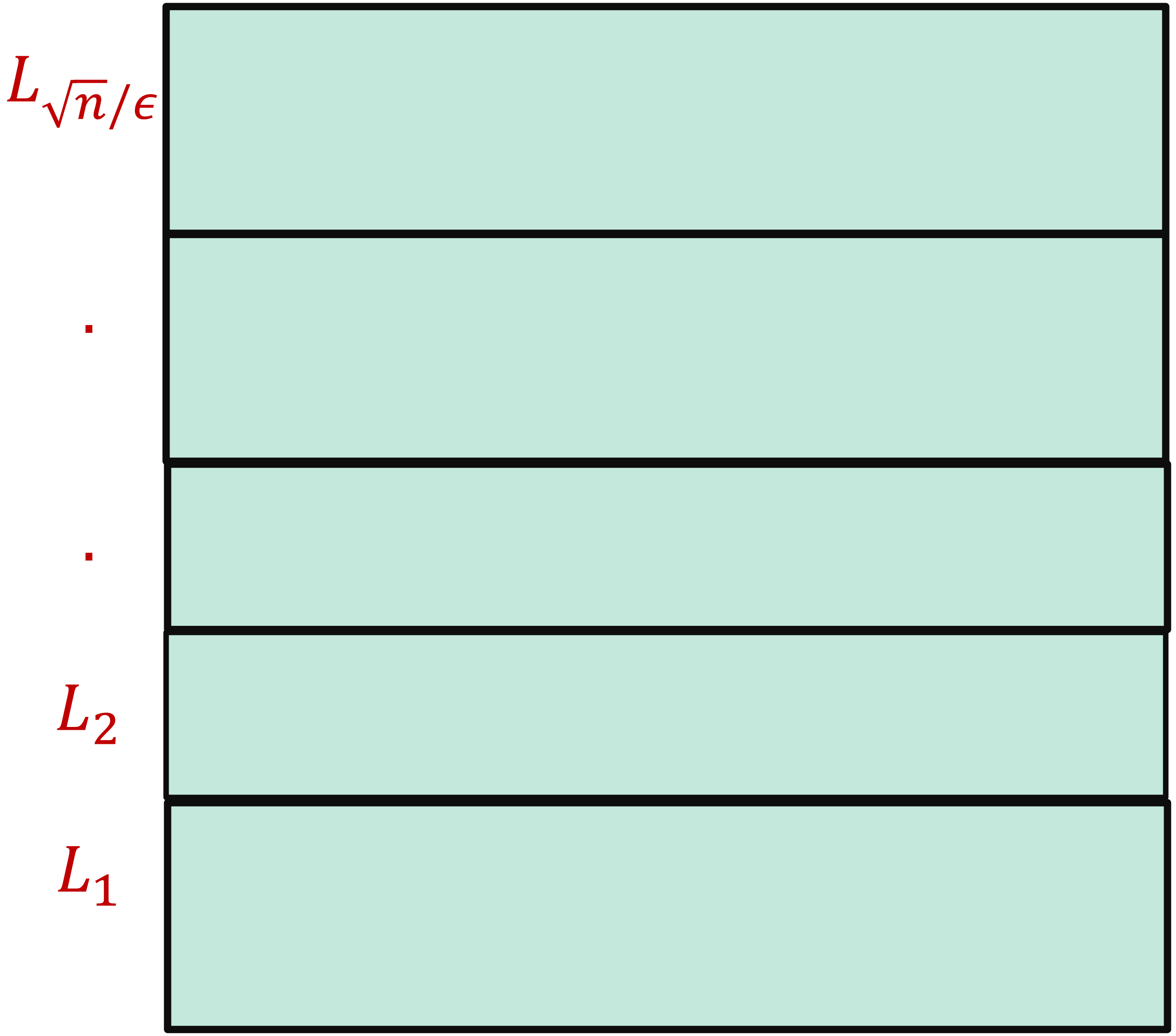
Sample $\tilde{\Theta}\,(\sqrt{n})$ points uniformly at random and equipartition the range into roughly $\sqrt{n}/\epsilon$ intervals of equal density

Density of each layer is roughly $\frac{1}{\sqrt{n}}$



Layering

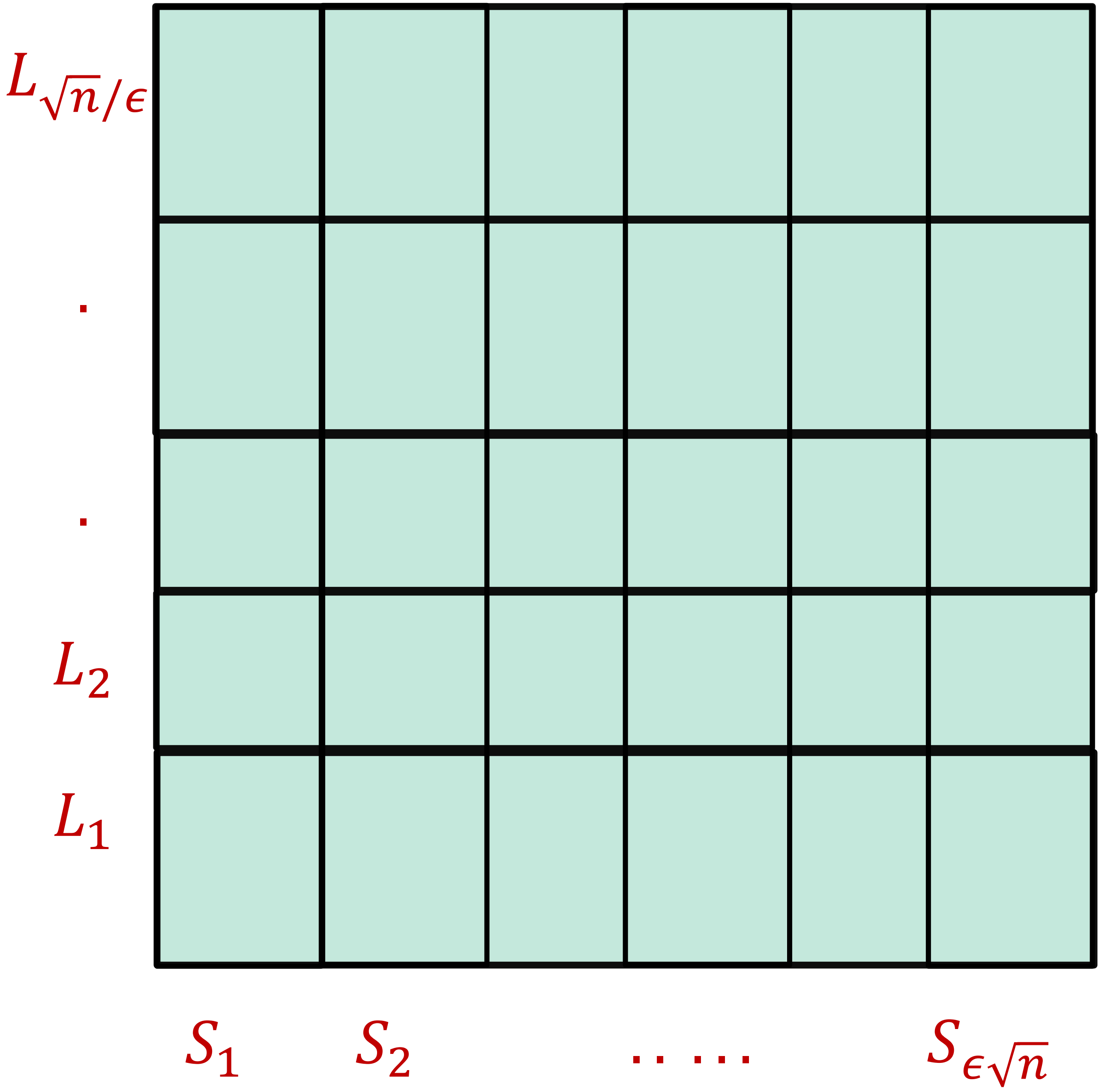
Partition the range into $\sqrt{n}/\epsilon$ **horizontal layers** of equal density



Layering

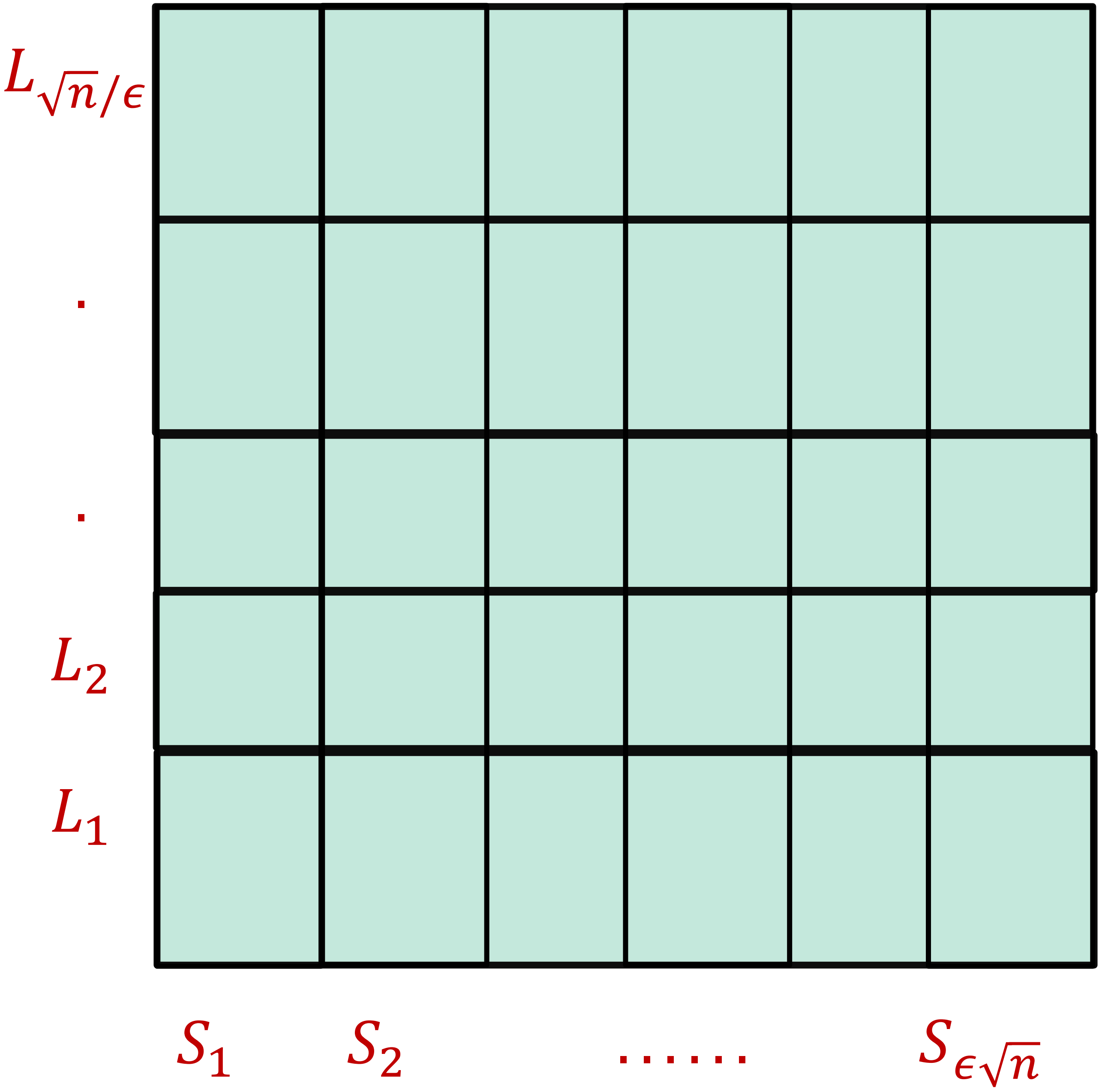
Partition the range into $\sqrt{n}/\epsilon$ **horizontal layers** of equal density

Equipartition index set into $\epsilon\sqrt{n}$ **vertical stripes**



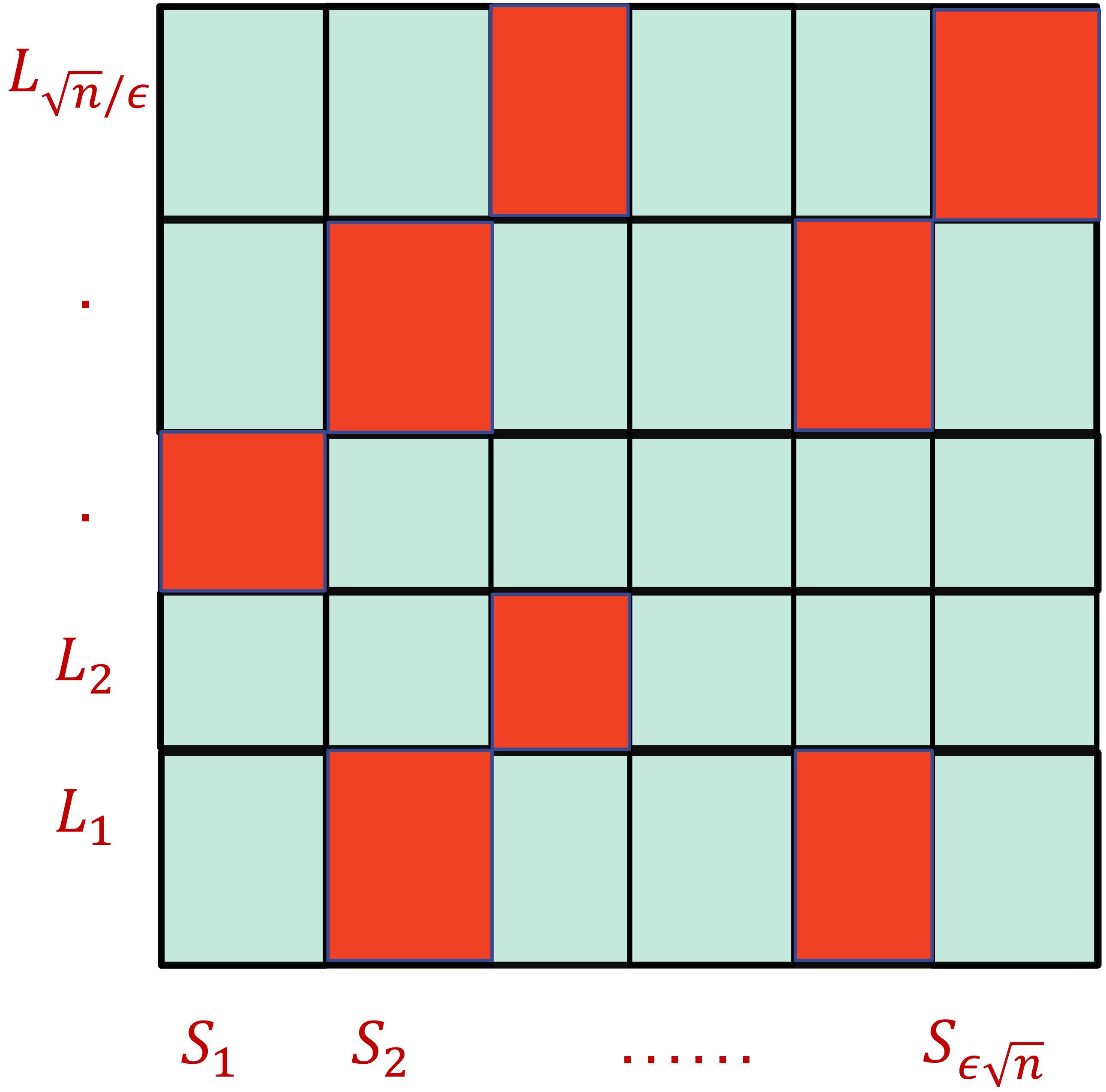
Tagging dense boxes

- Let $\beta = \epsilon^3 \lambda$, where λ is a known lower bound on LIS/n



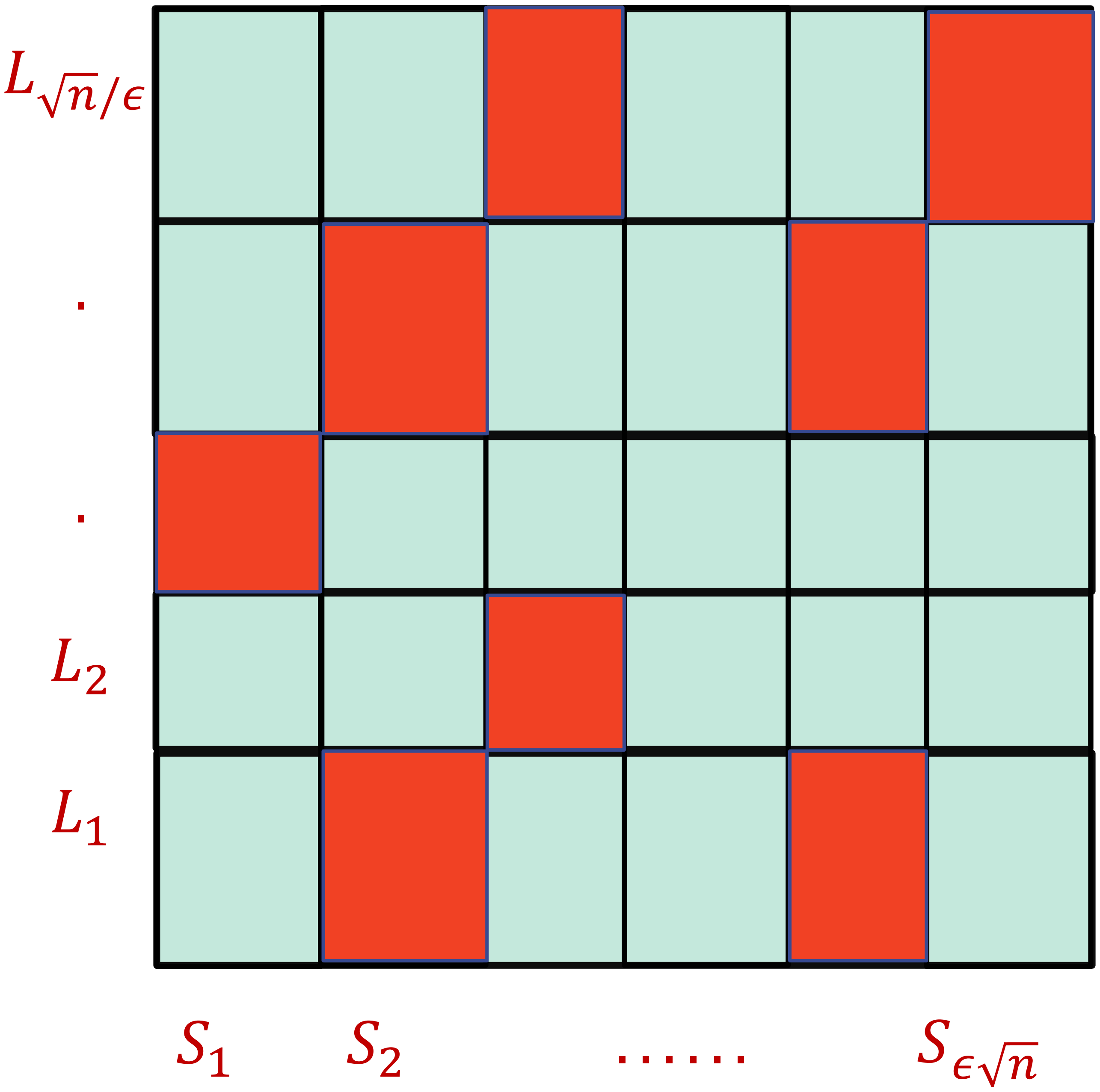
Tagging dense boxes

- Let $\beta = \epsilon^3 \lambda$, where λ is a known lower bound on LIS/n
- Identify boxes that contain β fraction of points in its stripe



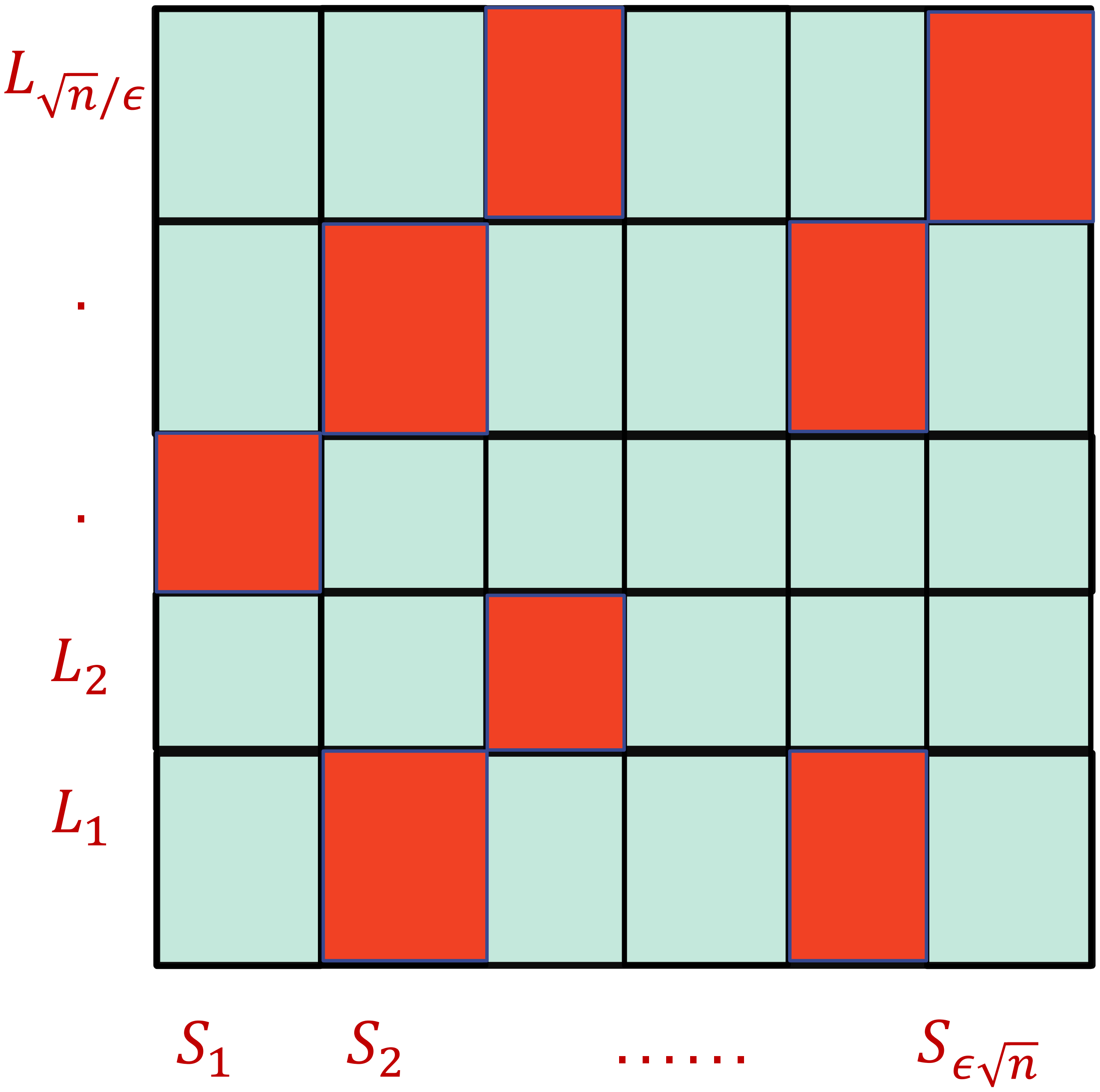
Tagging dense boxes

- Let $\beta = \epsilon^3\lambda$, where λ is a known lower bound on LIS/n
- Identify boxes that contain at least
  β fraction of points in its stripe
  by making $\tilde{\Theta}(\frac{1}{\beta})$ queries from each stripe.



Tagging dense boxes

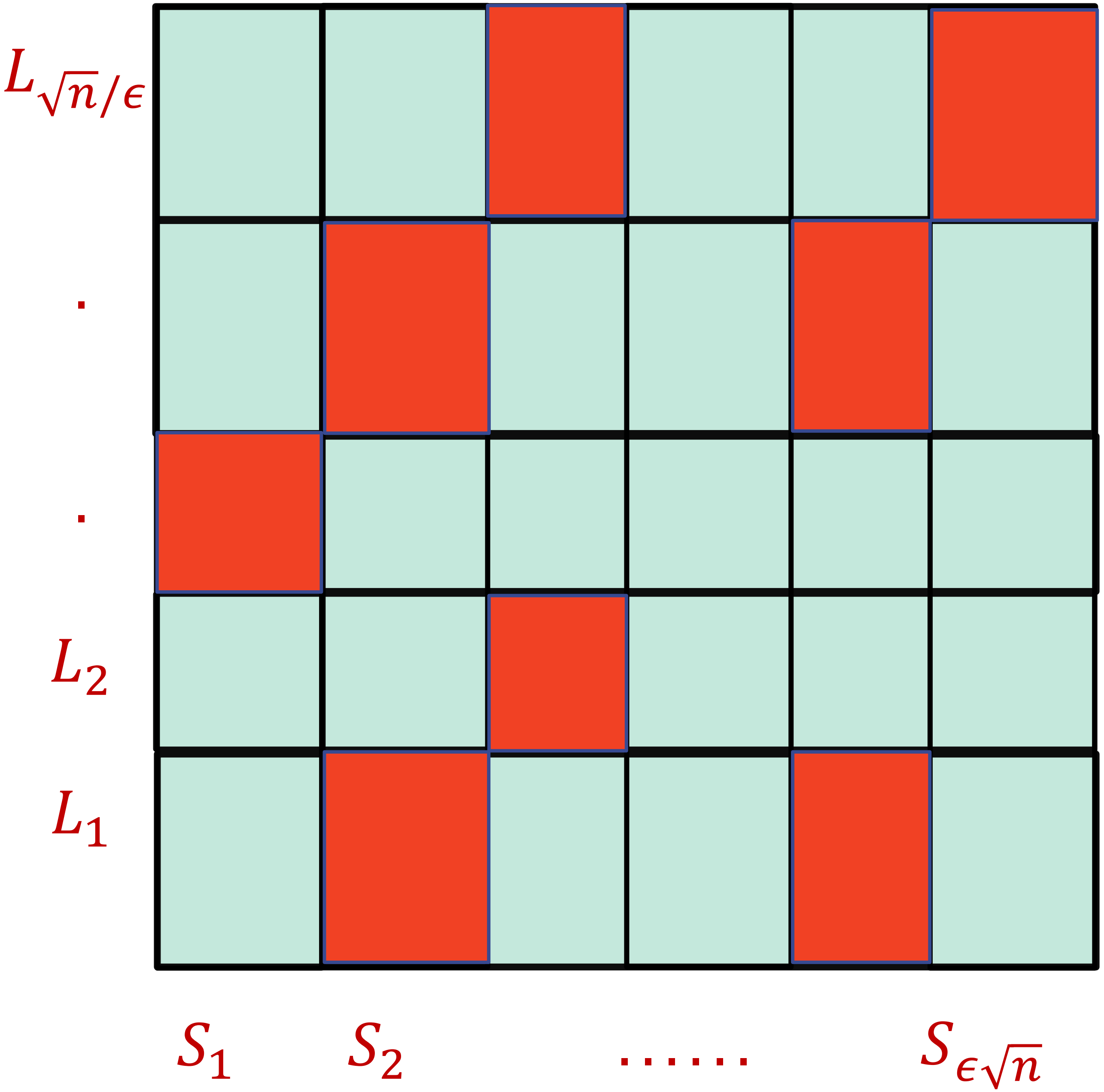
- Let $\beta = \epsilon^3 \lambda$, where λ is a known lower bound on LIS/n
- Identify boxes that contain at least β fraction of points in its stripe by making $\tilde{\Theta}(\frac{1}{\beta})$ queries from each stripe.
- Overall $\tilde{\Theta}(\sqrt{n})$ queries



Tagging dense boxes

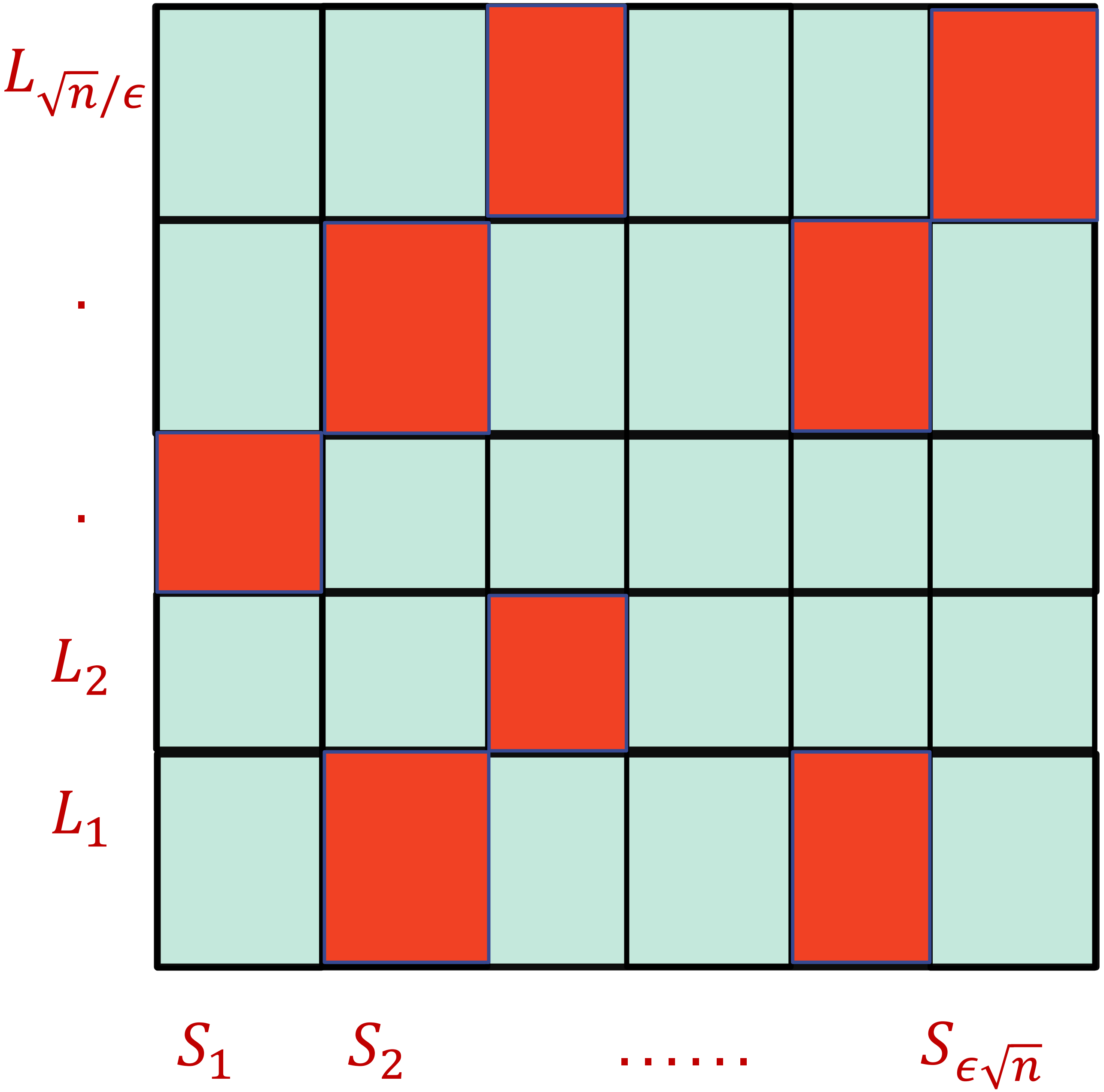
- Let $\beta = \epsilon^3 \lambda$, where λ is a known lower bound on LIS/n
- Identify boxes that contain at least
  β fraction of points in its stripe
  by making $\tilde{\Theta}(\frac{1}{\beta})$ queries from each stripe.
- Overall $\tilde{\Theta}(\sqrt{n})$ queries

Assume that the boxes have roughly β fraction of points



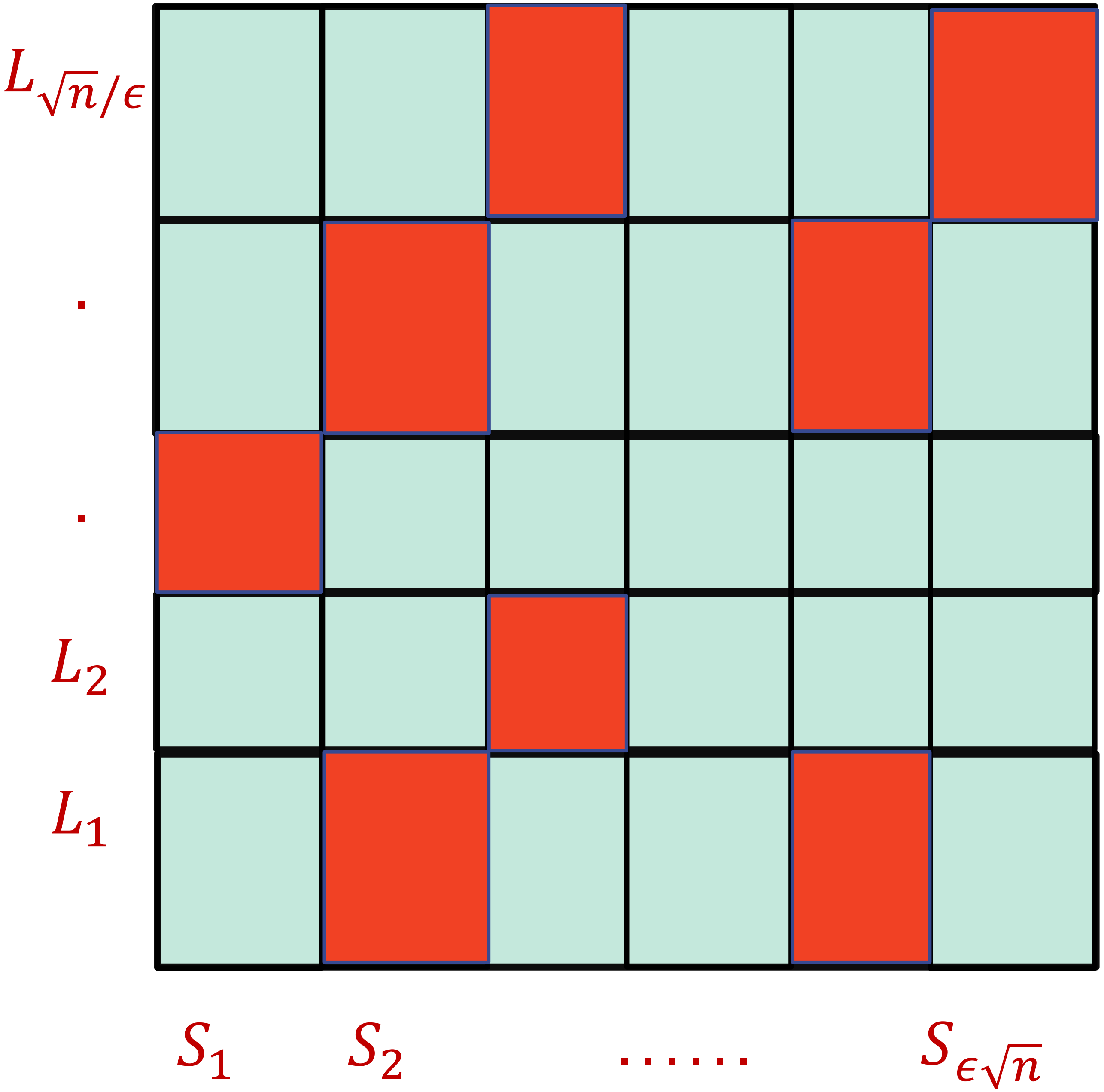
Ignore all non-dense boxes!

- At most $2\sqrt{n}/\epsilon$ boxes in any LIS



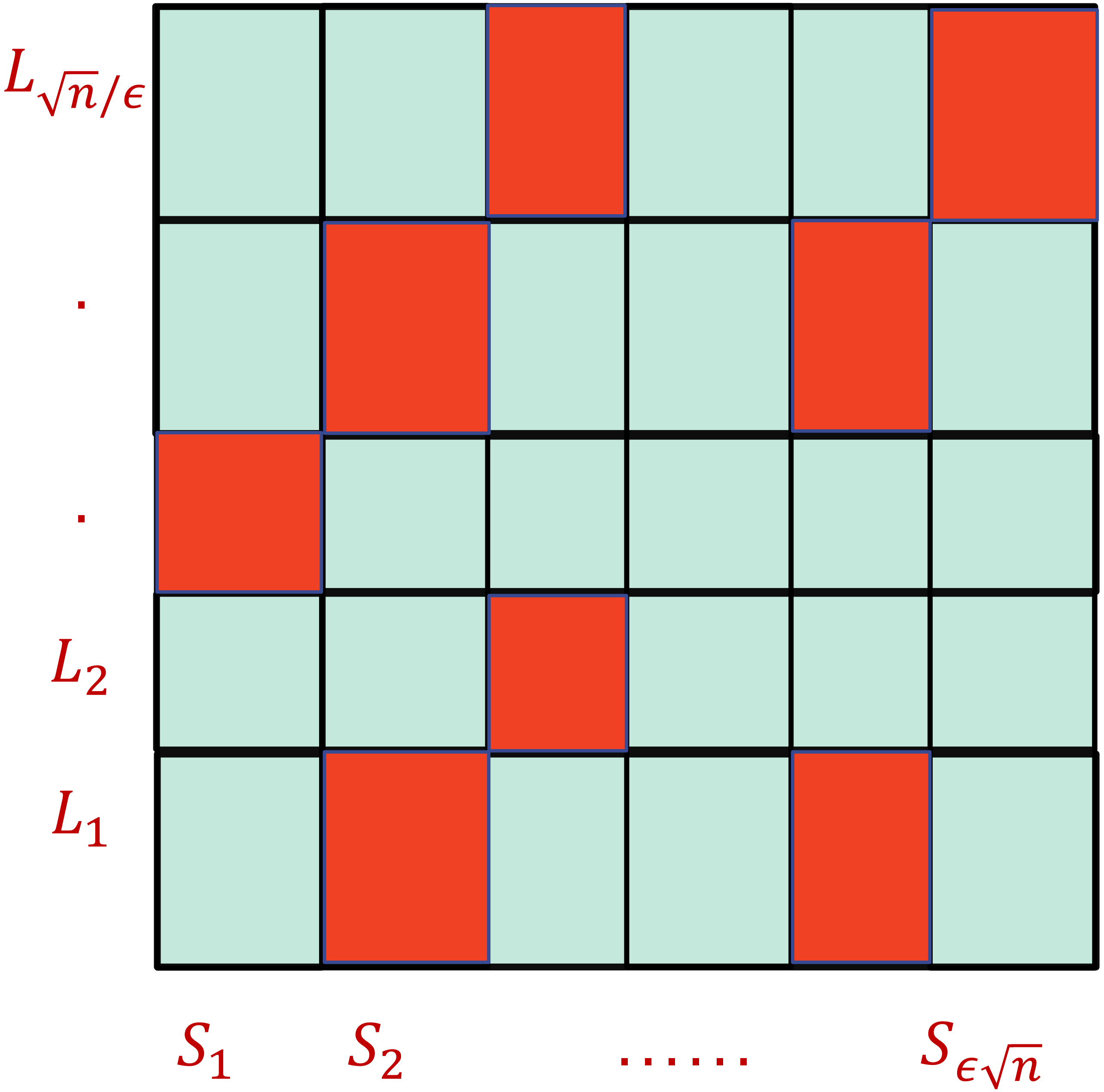
Ignore all non-dense boxes!

- At most $2\sqrt{n}/\epsilon$ boxes in any LIS
- Each non-dense box contributes at most $\beta\sqrt{n} = \epsilon^3\lambda\sqrt{n}$ points



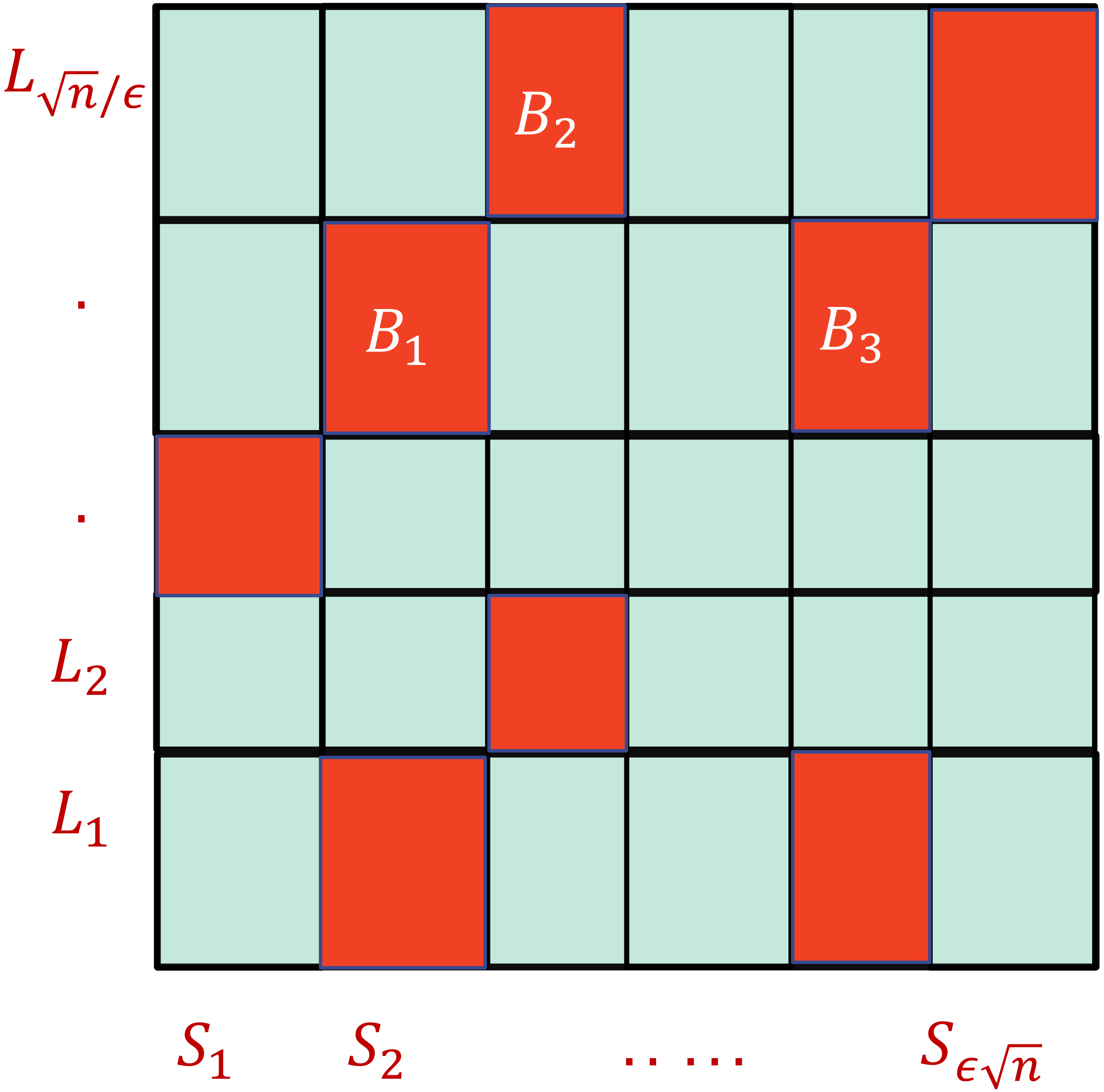
Ignore all non-dense boxes!

- At most $2\sqrt{n}/\epsilon$ boxes in any LIS
- Each non-dense box contributes at most $\beta\sqrt{n} = \epsilon^3\lambda\sqrt{n}$ points
- By ignoring non-dense boxes, we lose $2\epsilon^2\lambda n \leq 2\,\epsilon^2 \cdot$LIS many points



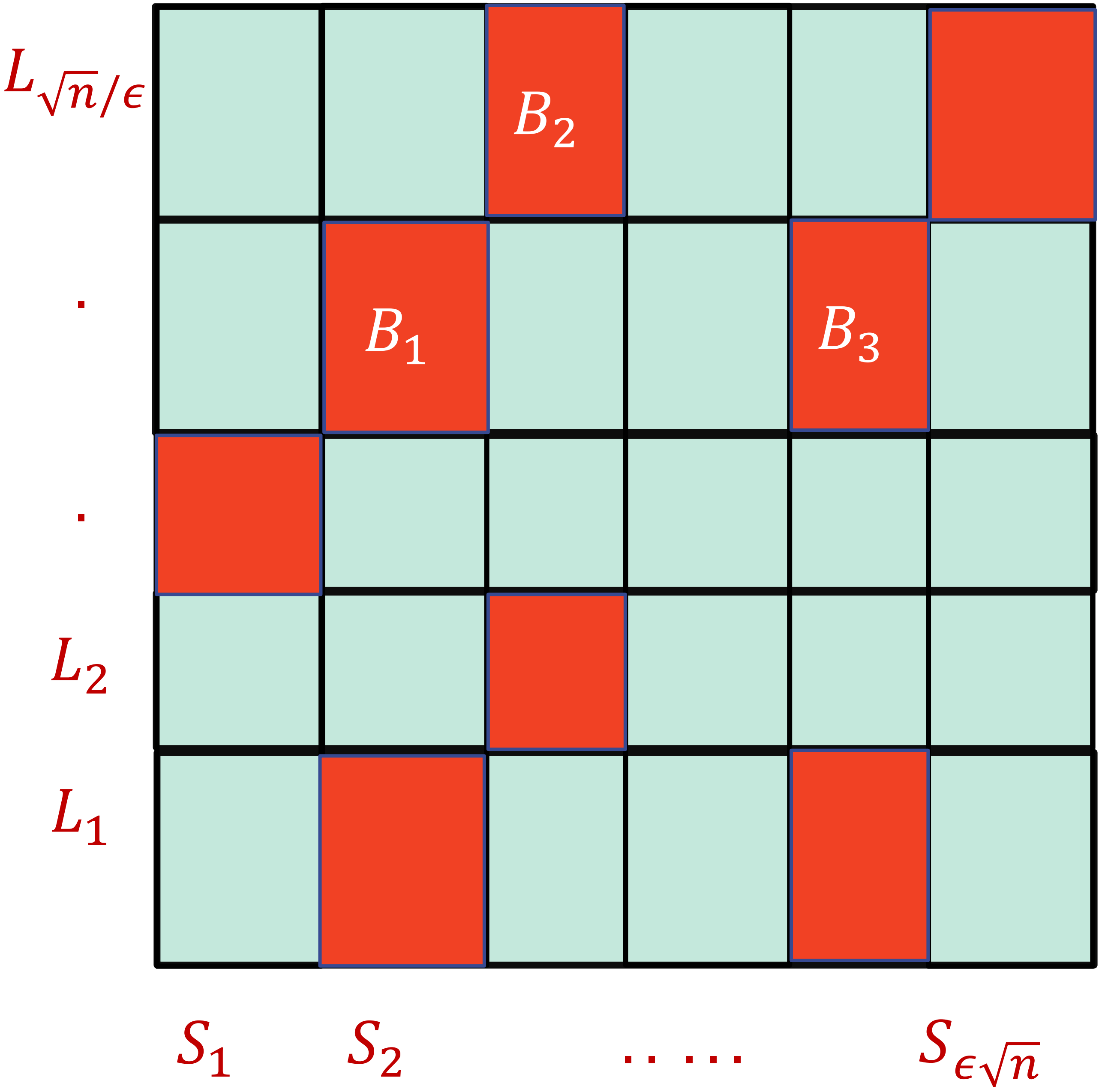
Poset, chains, and LIS

- $\langle P, \preccurlyeq \rangle$: Natural poset on dense boxes



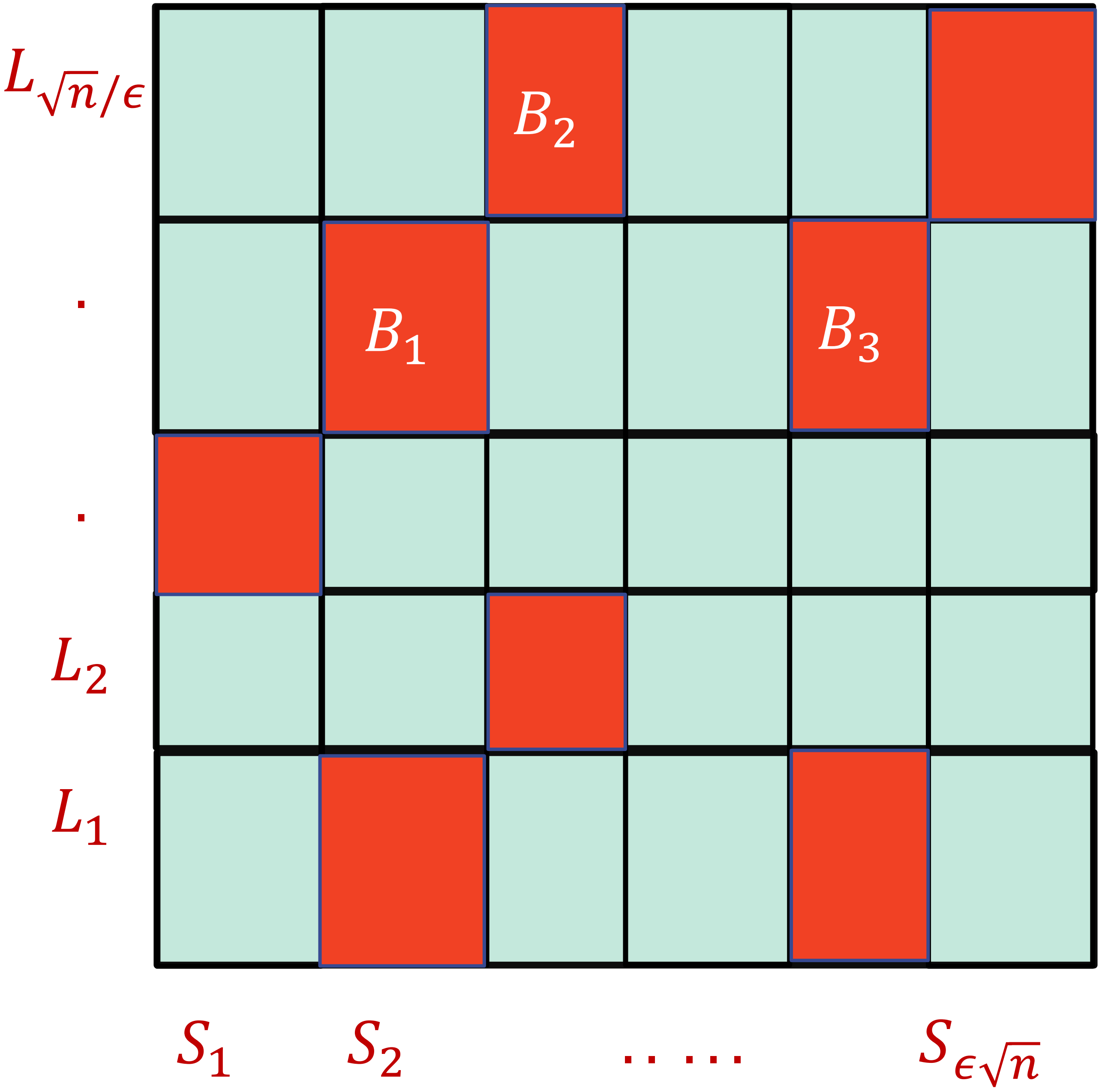
Poset, chains, and LIS

- $\langle P, \preccurlyeq \rangle$: Natural poset on dense boxes
- Set of boxes through which LIS passes is a chain in P



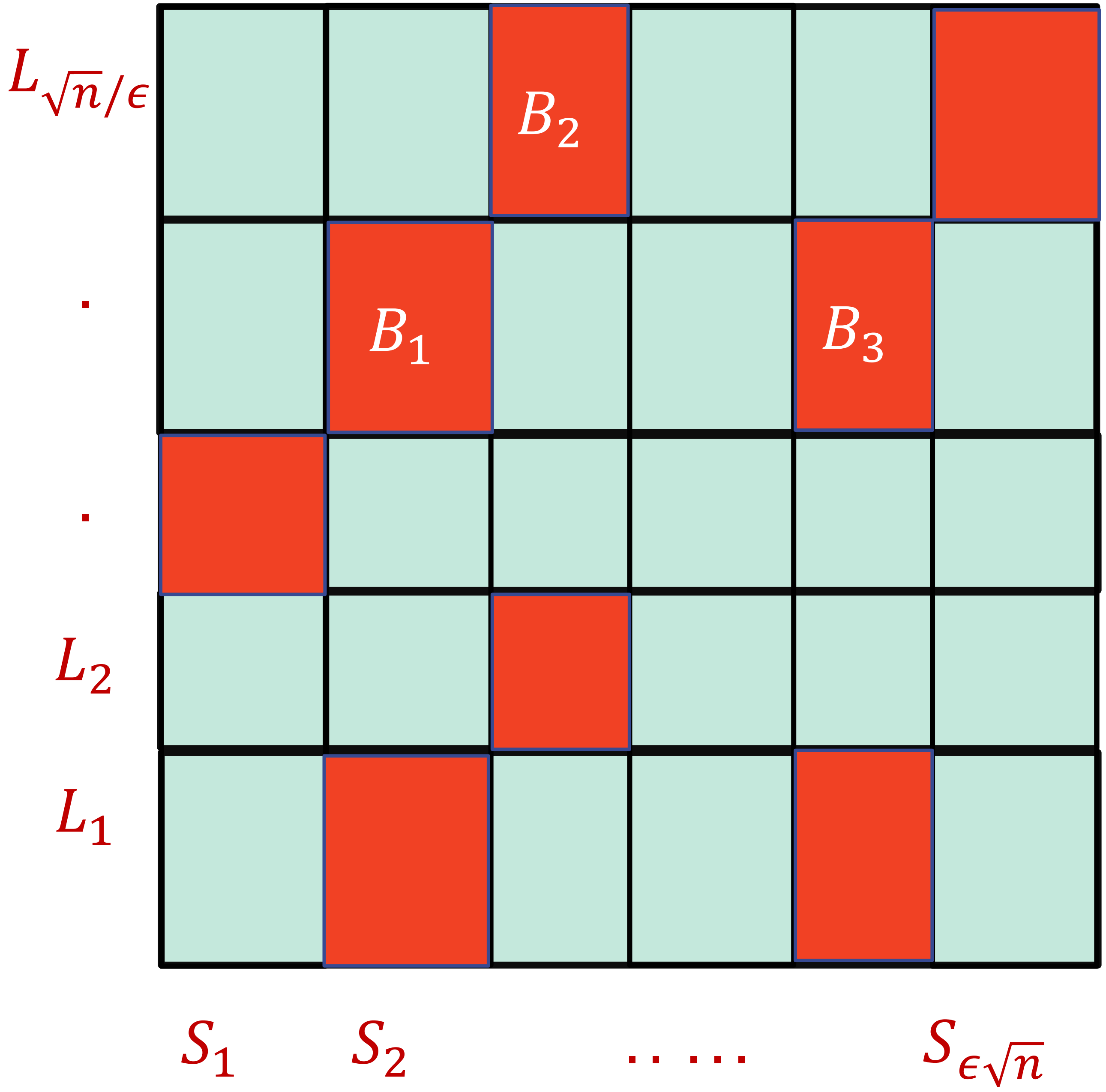
Poset, chains, and LIS

- $\langle P, \preccurlyeq \rangle$: Natural poset on dense boxes
- Set of boxes through which LIS passes is a chain in P
- Each chain in P corresponds to an increasing sequence



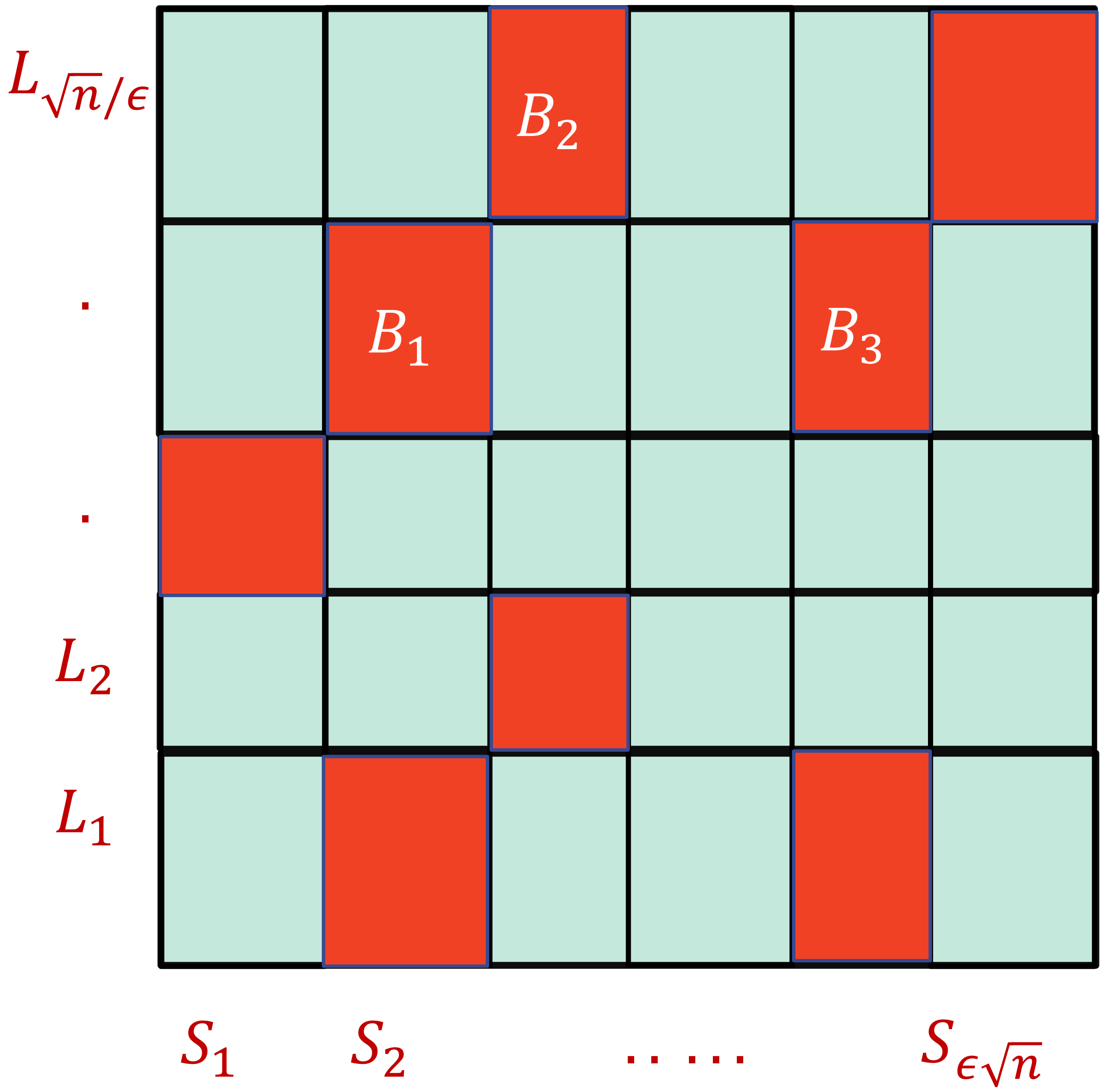
Poset, chains, and LIS

- **Strategy:** Estimate the lengths of each chain in P and output the max. value



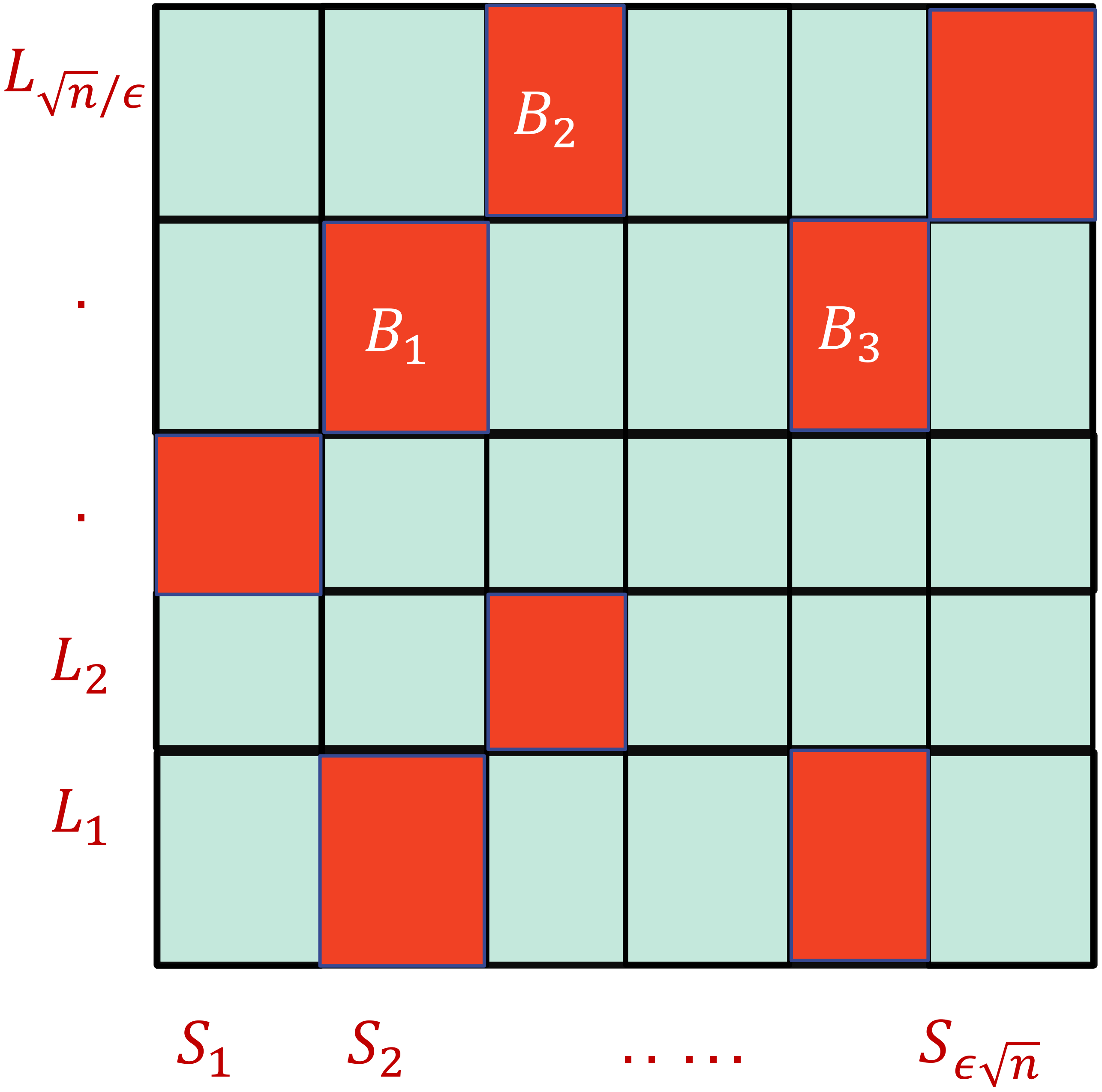
Poset, chains, and LIS

- **Strategy:** Estimate the lengths of each chain in P and output the max. value
- **Issue**: Too many chains!



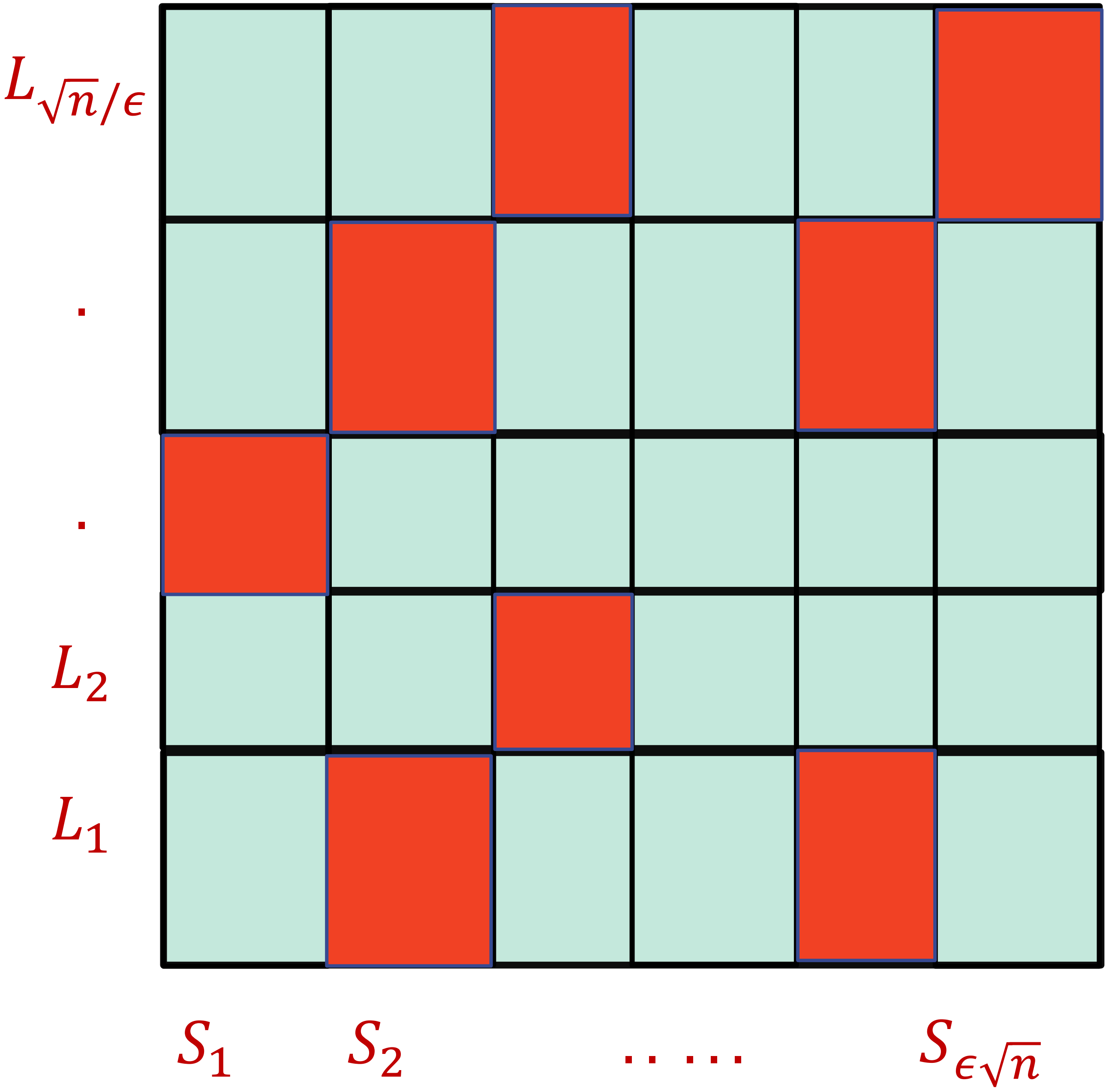
Poset, chains, and LIS

- **Strategy:** Estimate the lengths of each chain in P and output the max. value
- **Issue**: Too many chains!
- **Fix:** Reduce the number of chains by removing large antichains



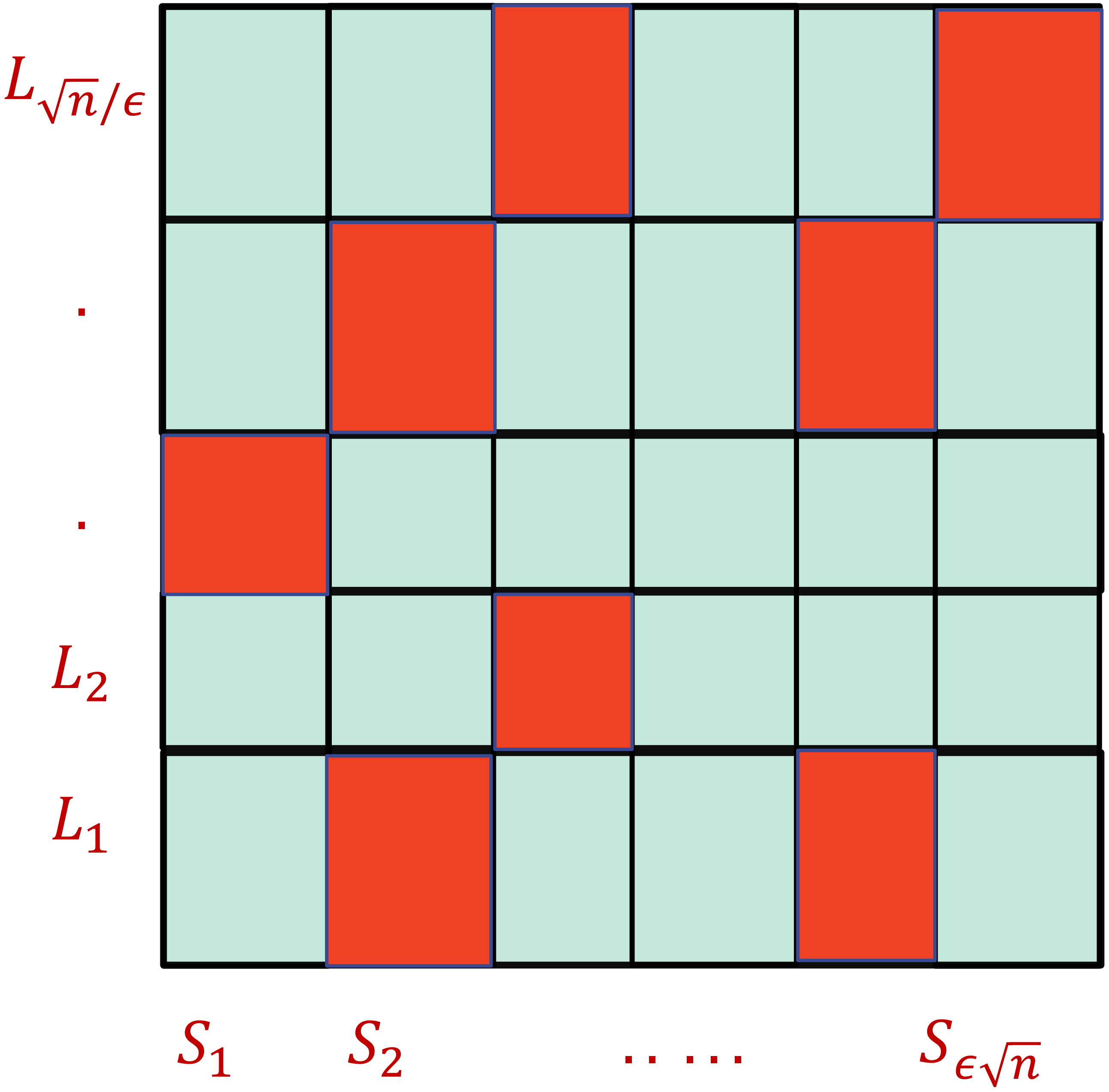
Chain reduction

- Set of cells through which LIS passes is a chain in P



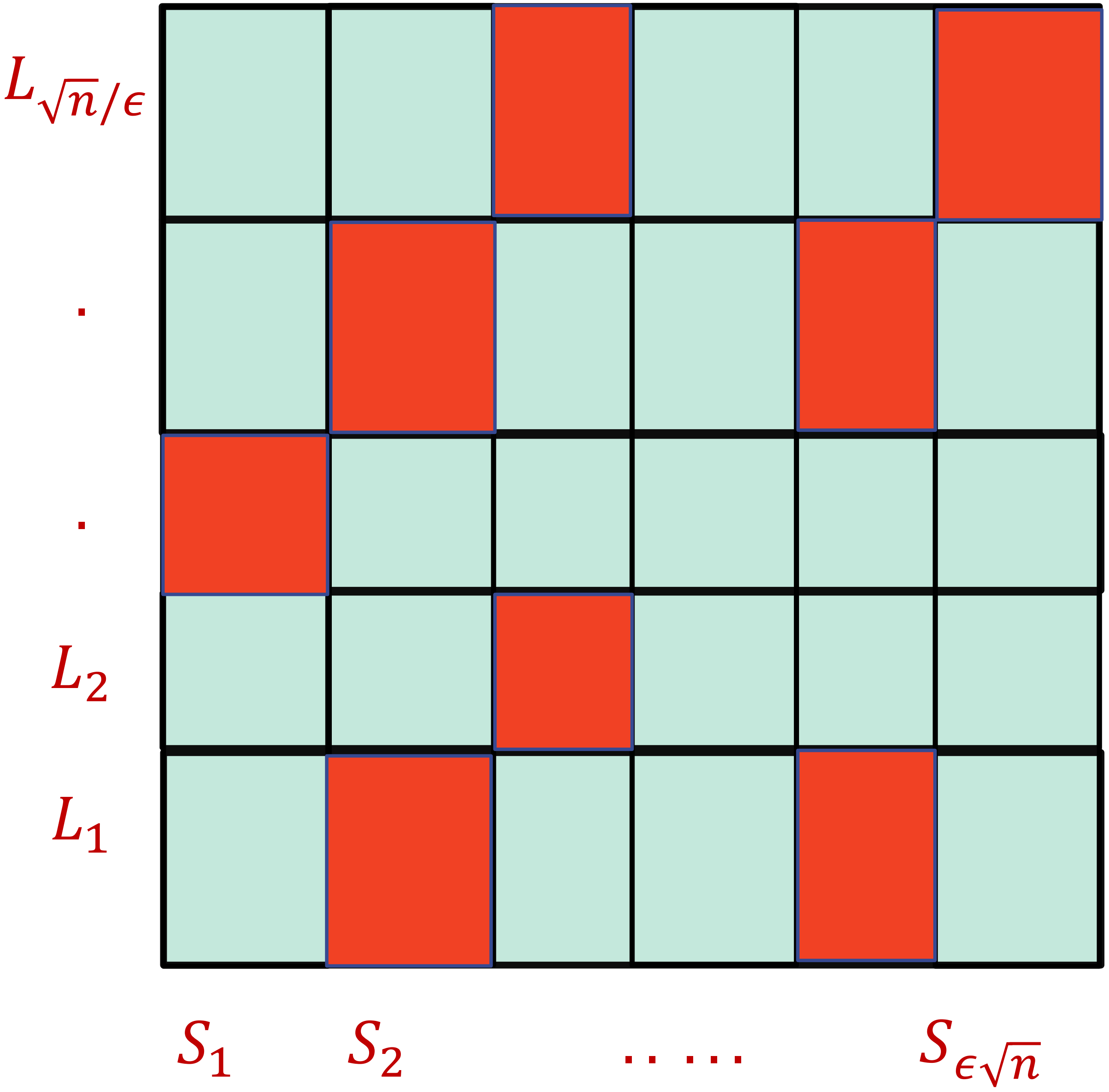
Chain reduction

- Set of cells through which LIS passes is a chain in P
- **Chain reduction:** Repeatedly remove antichains in P consisting of $\Theta(\frac{1}{\lambda})$ dense cells each



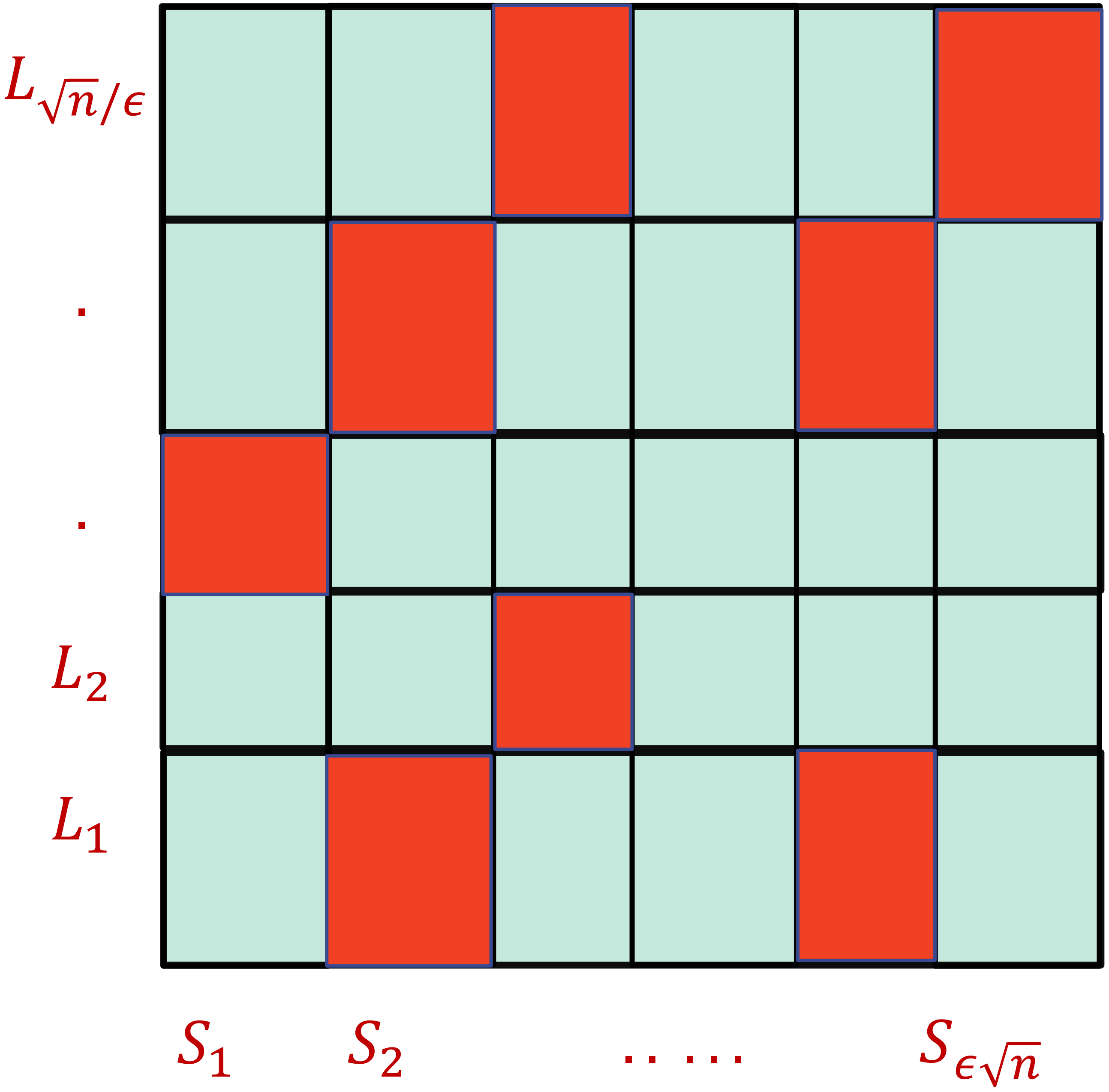
Chain reduction does not hurt much

- At most $\epsilon\sqrt{n}/\epsilon^3\lambda$ dense cells



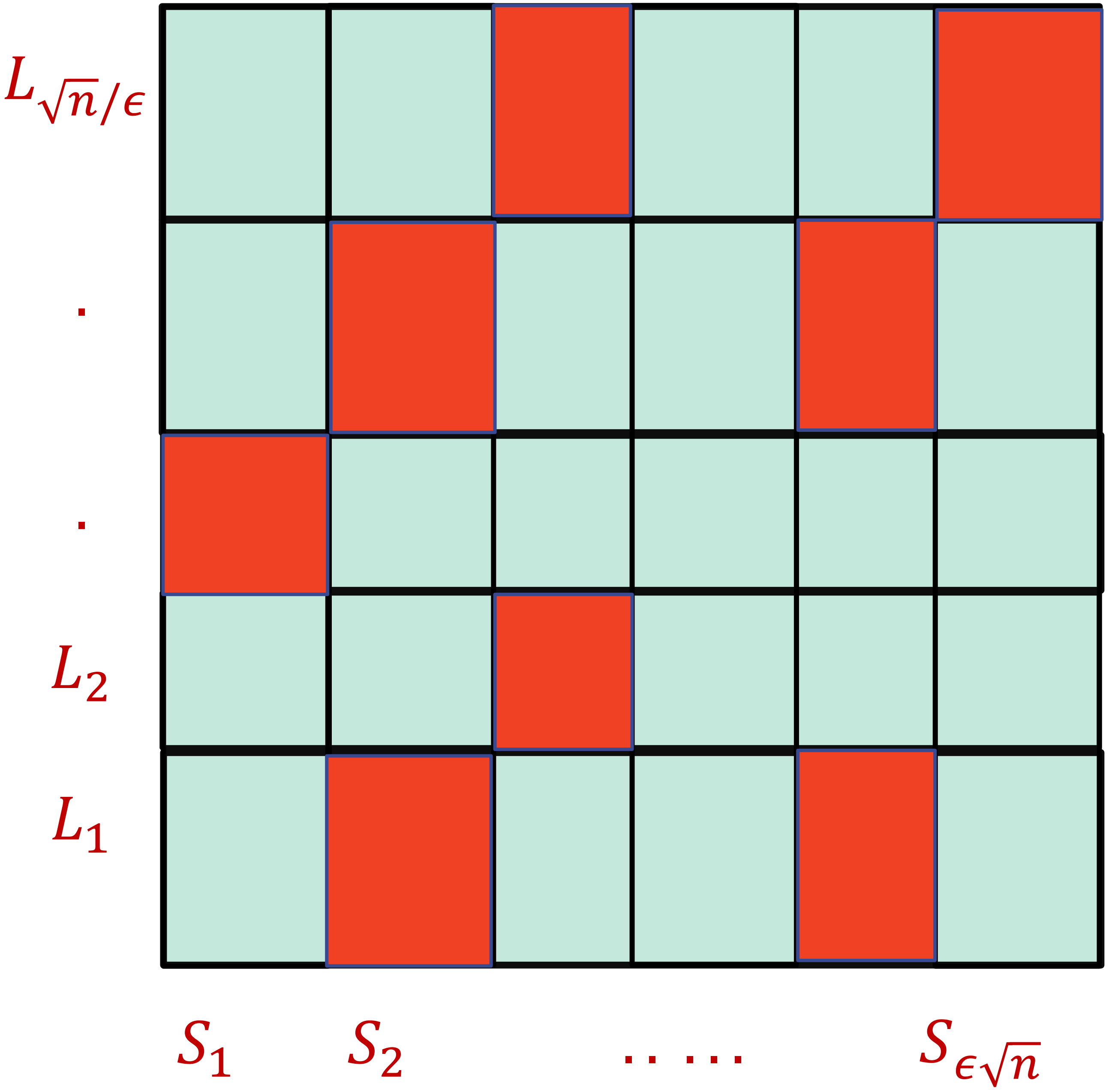
Chain reduction does not hurt much

- At most $\epsilon\sqrt{n}/\epsilon^3\lambda$ dense cells
- Antichain removal can be done at most $(\sqrt{n}/(\epsilon^2\,\lambda))/(1/\lambda) = \sqrt{n}/\epsilon^2$ times



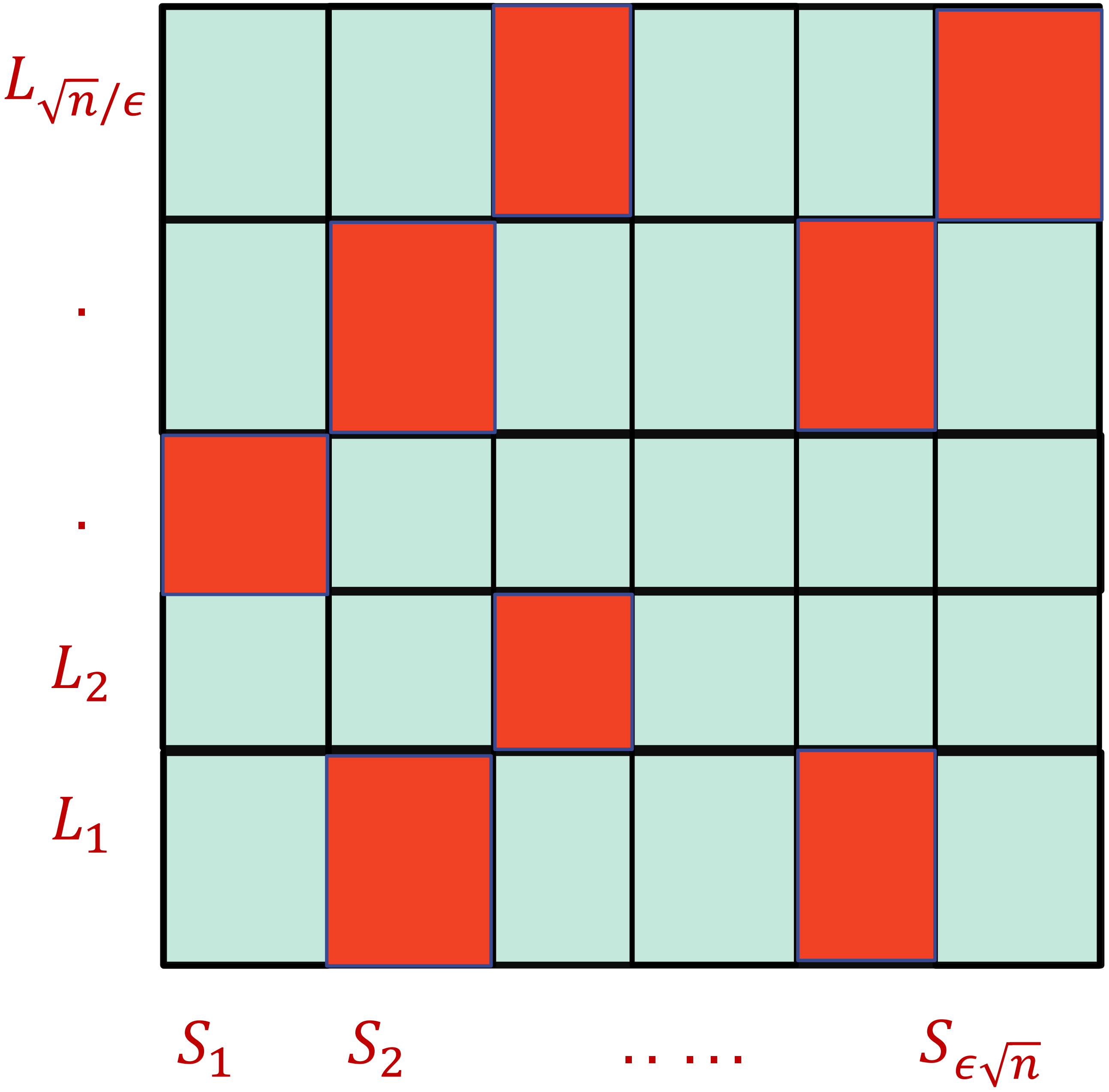
Chain reduction does not hurt much

- At most $\epsilon\sqrt{n}/\epsilon^3\lambda$ dense cells
- Antichain removal can be done at most $(\sqrt{n}/(\epsilon^2\,\lambda))/(1/\lambda) = \sqrt{n}/\epsilon^2$ times
- Each antichain removal hits one dense cell from an LIS



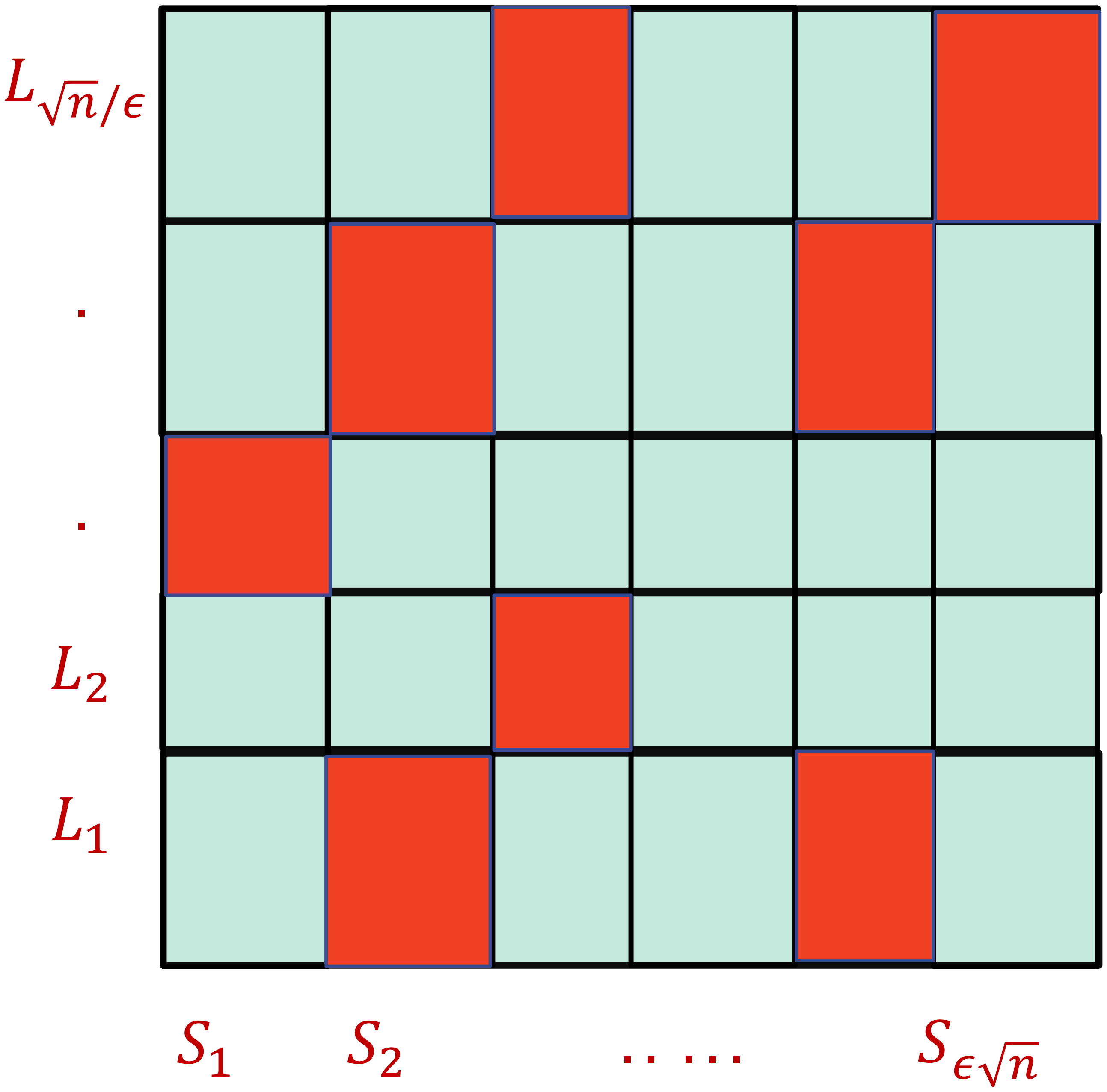
Chain reduction does not hurt much

- At most $\epsilon\sqrt{n}/\epsilon^3\lambda$ dense cells
- Antichain removal can be done at most $(\sqrt{n}/(\epsilon^2\,\lambda))/(1/\lambda) = \sqrt{n}/\epsilon^2$ times
- Each antichain removal hits one dense cell from an LIS
- Total loss to LIS at most $\epsilon\lambda n = \epsilon \cdot$LIS points



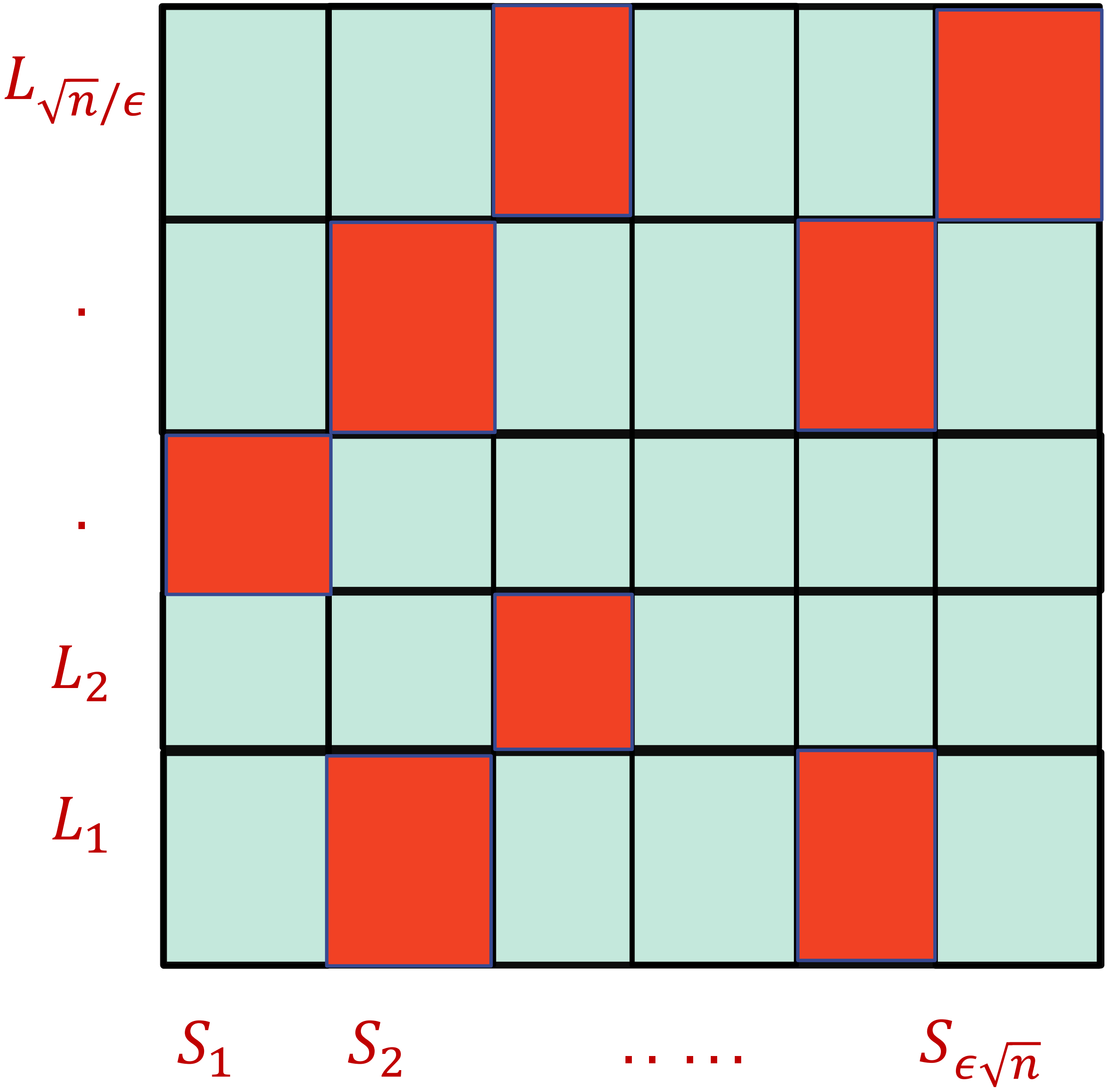
After chain reduction

- After chain reduction, P can be covered with at most $O(\frac{1}{\lambda})$ chains



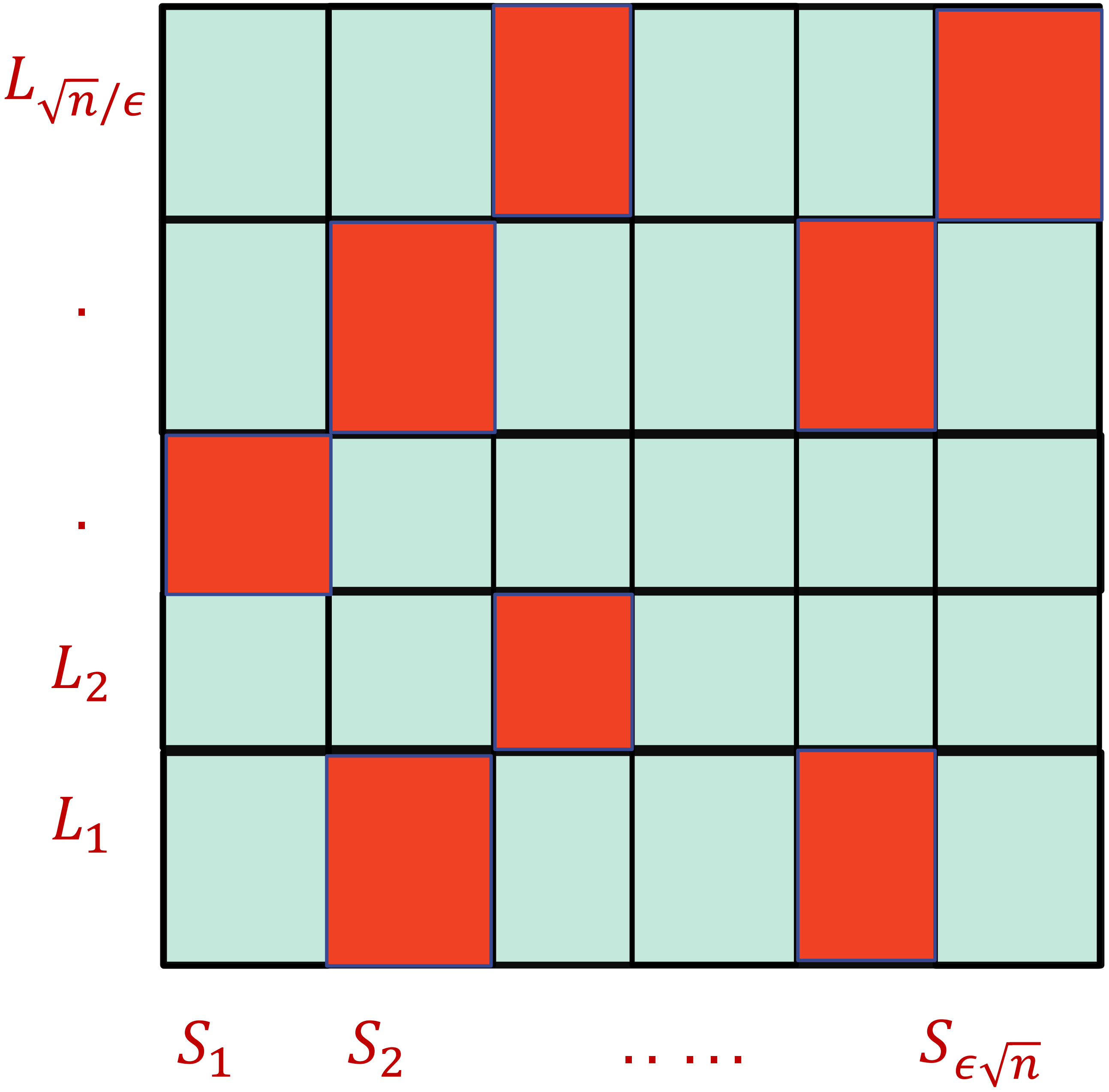
After chain reduction

- After chain reduction, P can be covered with at most $O(\frac{1}{\lambda})$ chains
- Estimate LIS in these chains



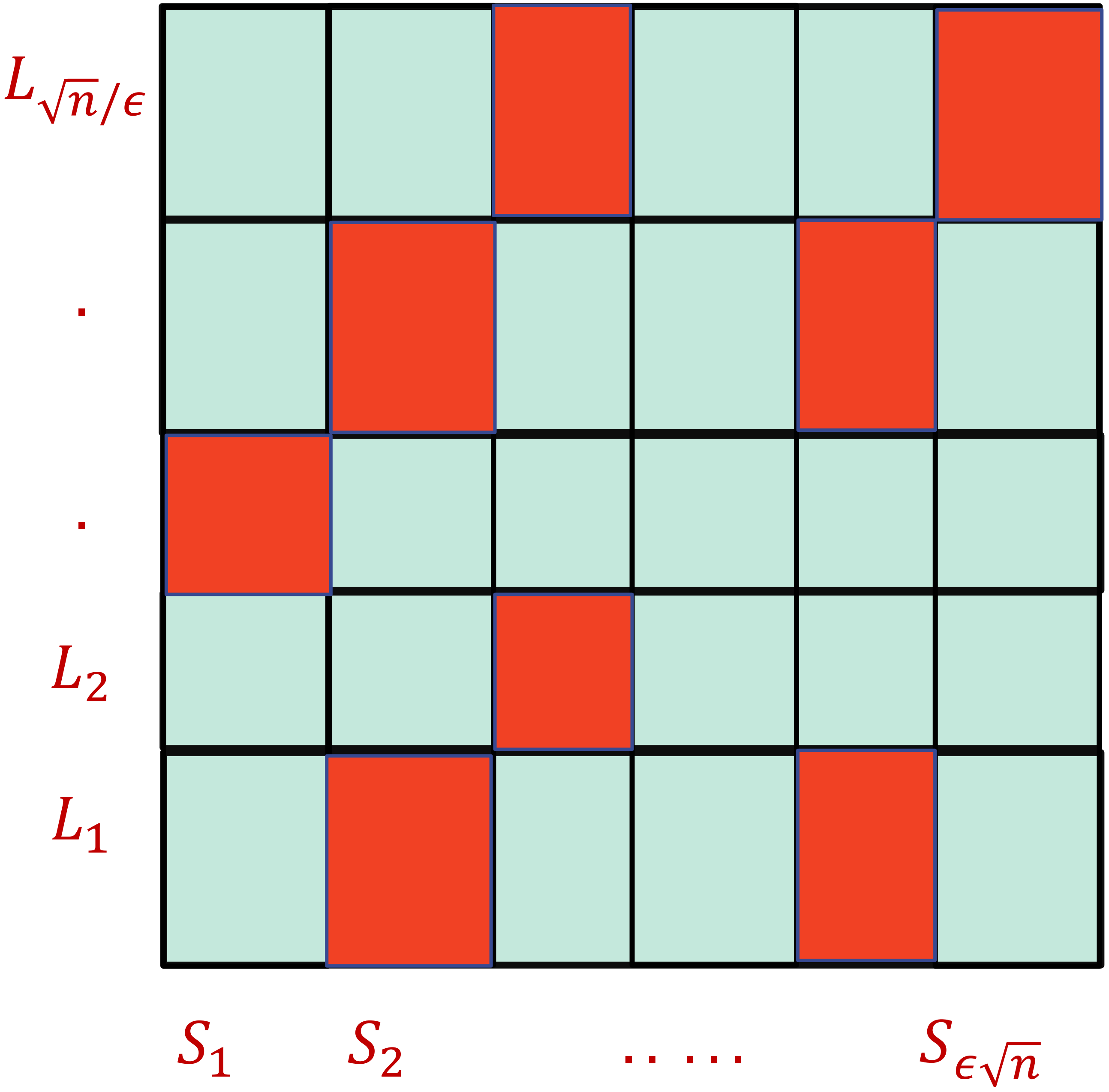
After chain reduction

- After chain reduction, P can be covered with at most $O(\frac{1}{\lambda})$ chains
- Estimate LIS in these chains
- Output the max. value estimated

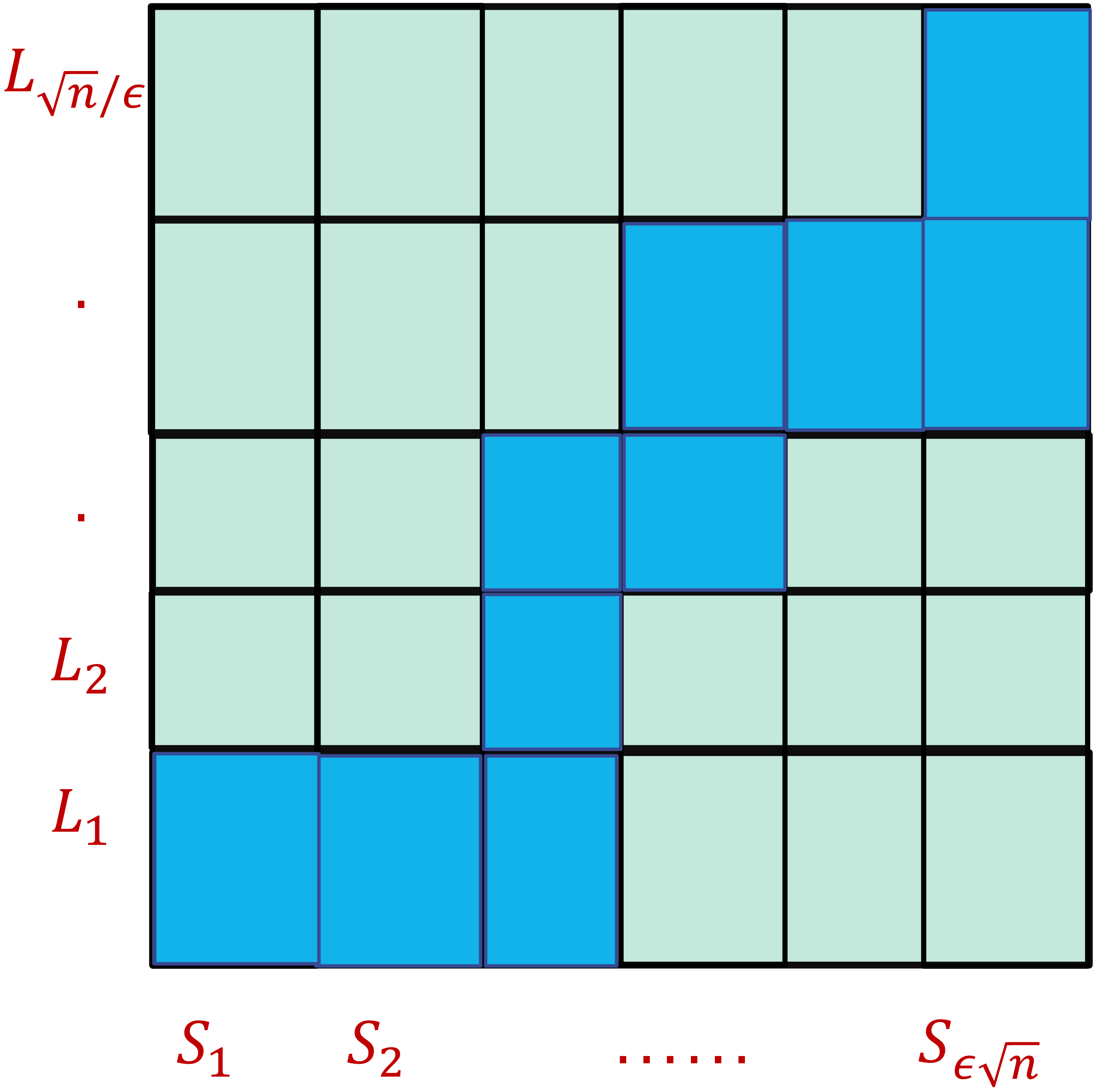


After chain reduction

- After chain reduction, P can be covered with at most $O(\frac{1}{\lambda})$ chains
- Estimate LIS in these chains
- Output the max. value estimated
- Loss to LIS incurred by a factor of λ

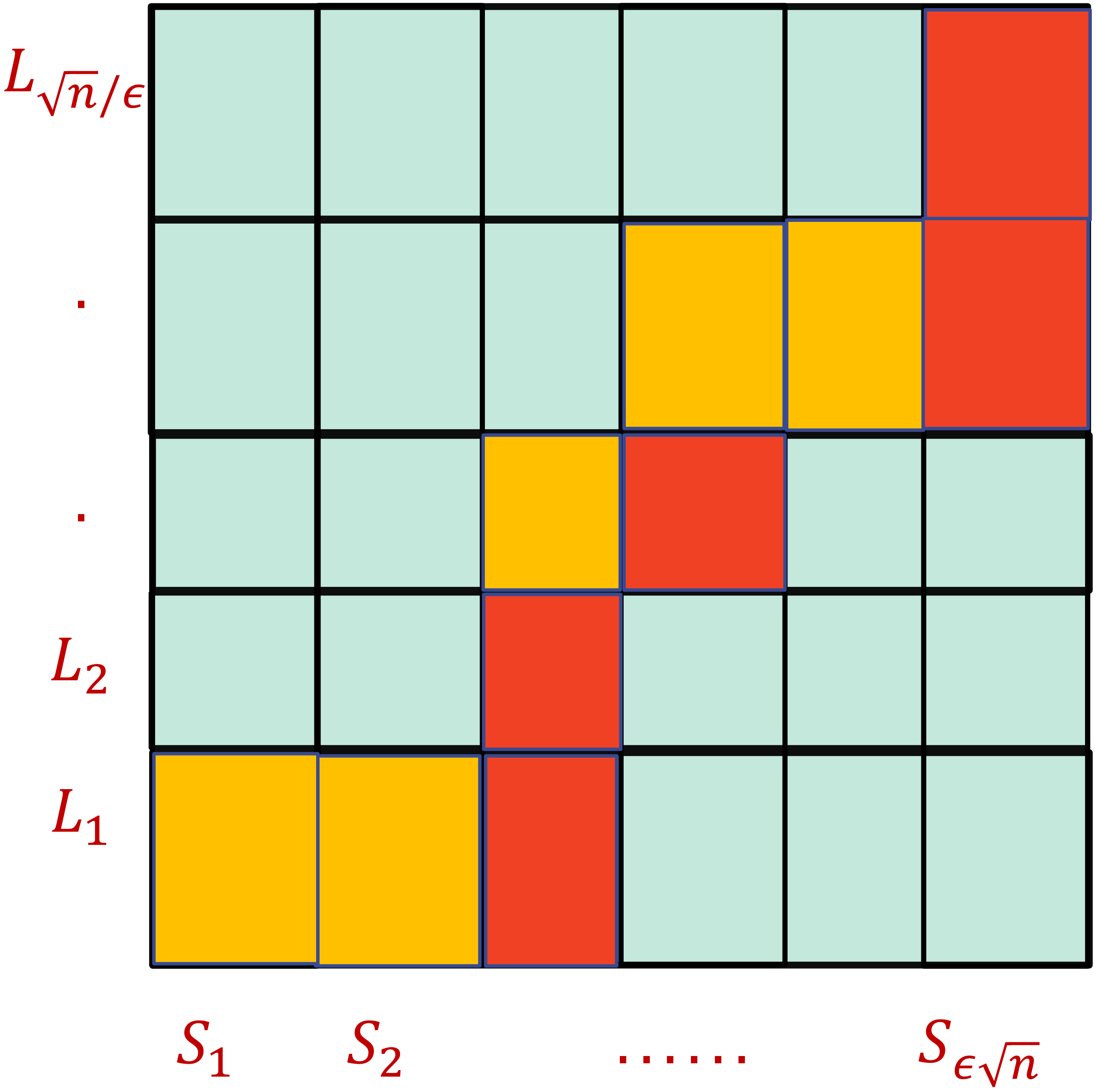


Estimating LIS in one chain



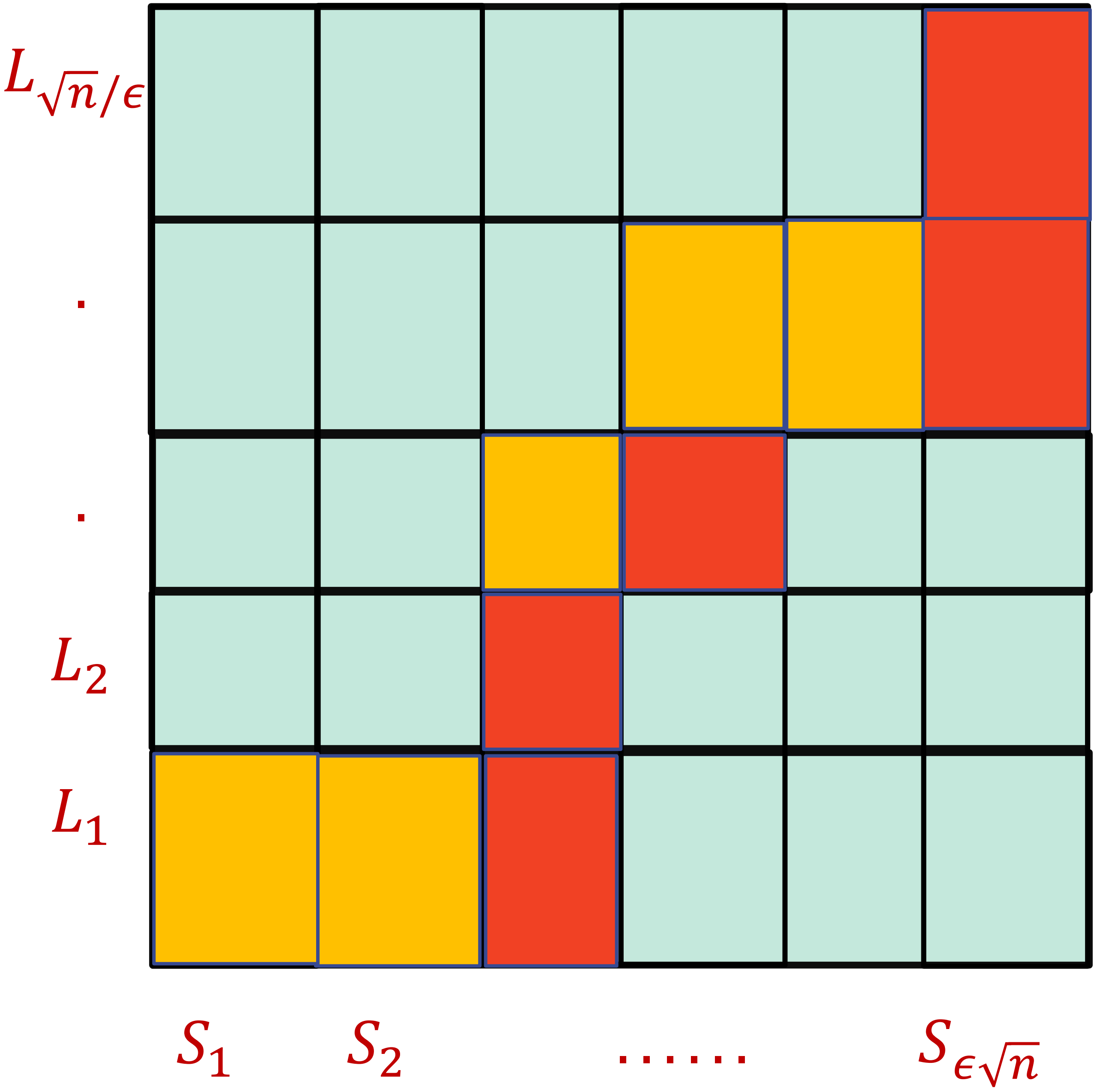
Estimating LIS in one chain

- Partition into horizontal and vertical chains



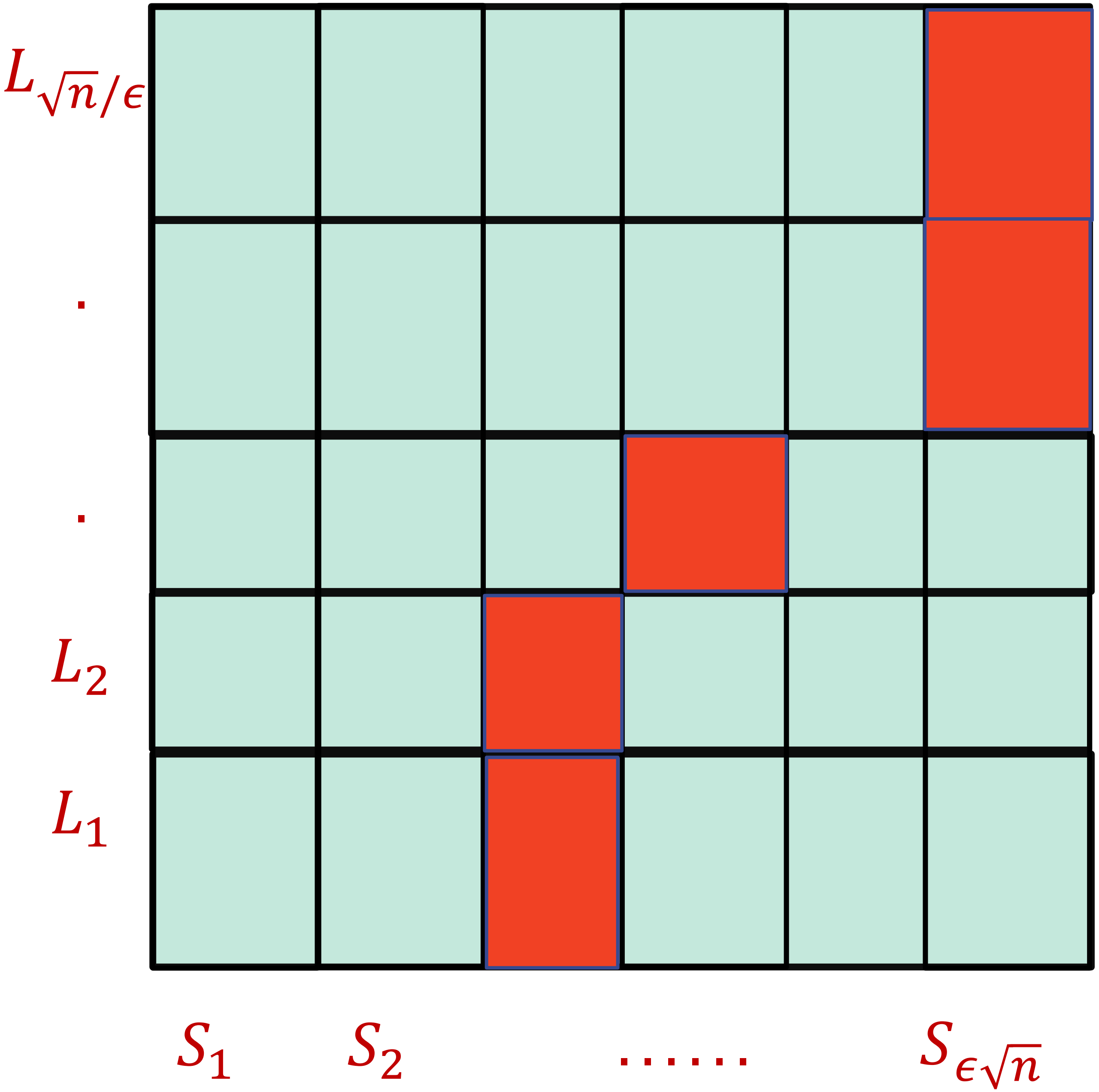
Estimating LIS in one chain

- Partition into horizontal and vertical chains
- Estimate LIS separately for vertical and horizontal chains



Vertical chains

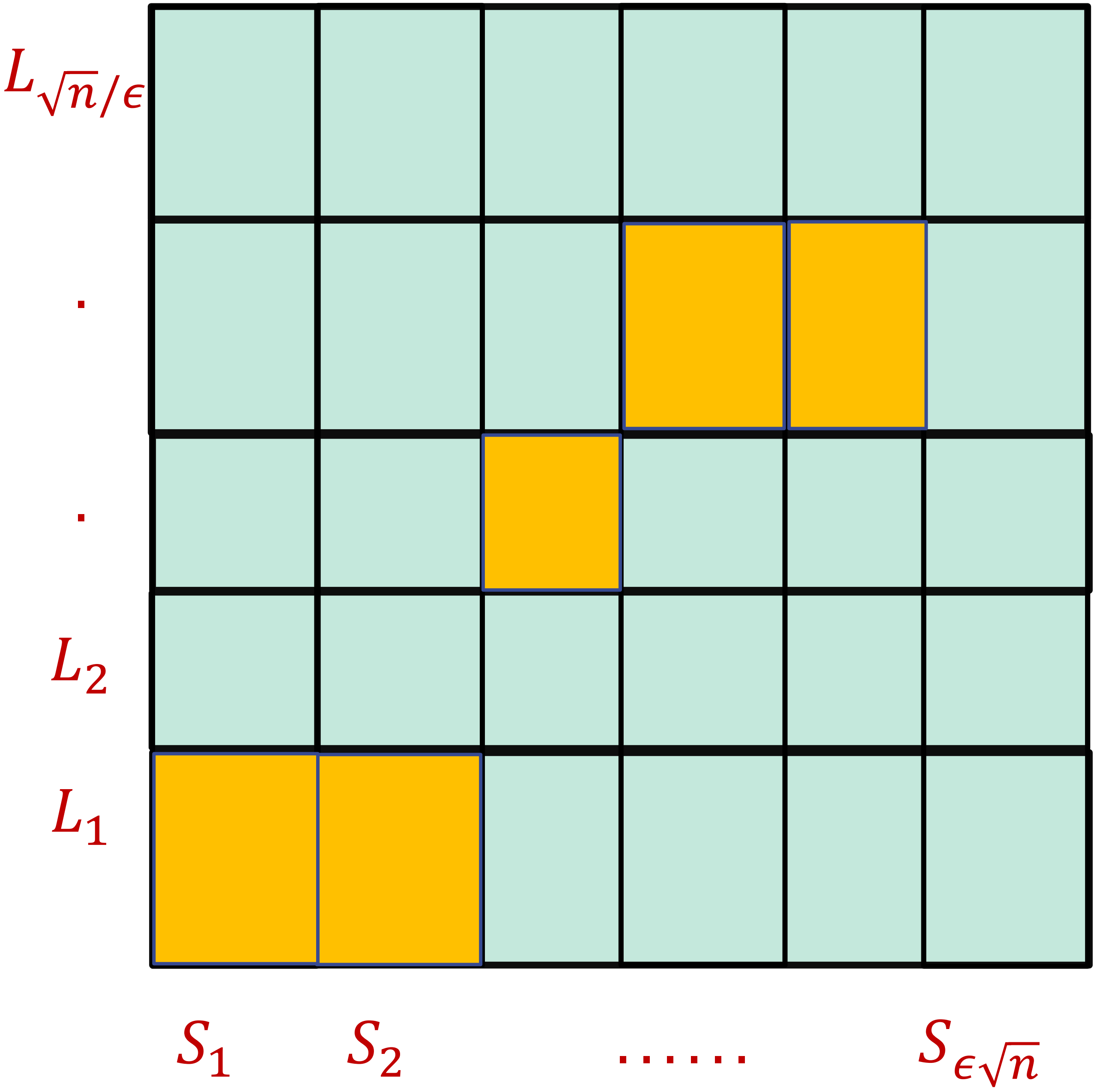
- For each vertical block, can evaluate the LIS by making $\sqrt{n}$ queries
- Sample a constant number of vertical blocks and evaluate LIS in the sampled blocks to estimate LIS in the vertical chain
- Above step can be made nonadaptive



Horizontal chains

Let $\ell = \epsilon/\lambda^2$.

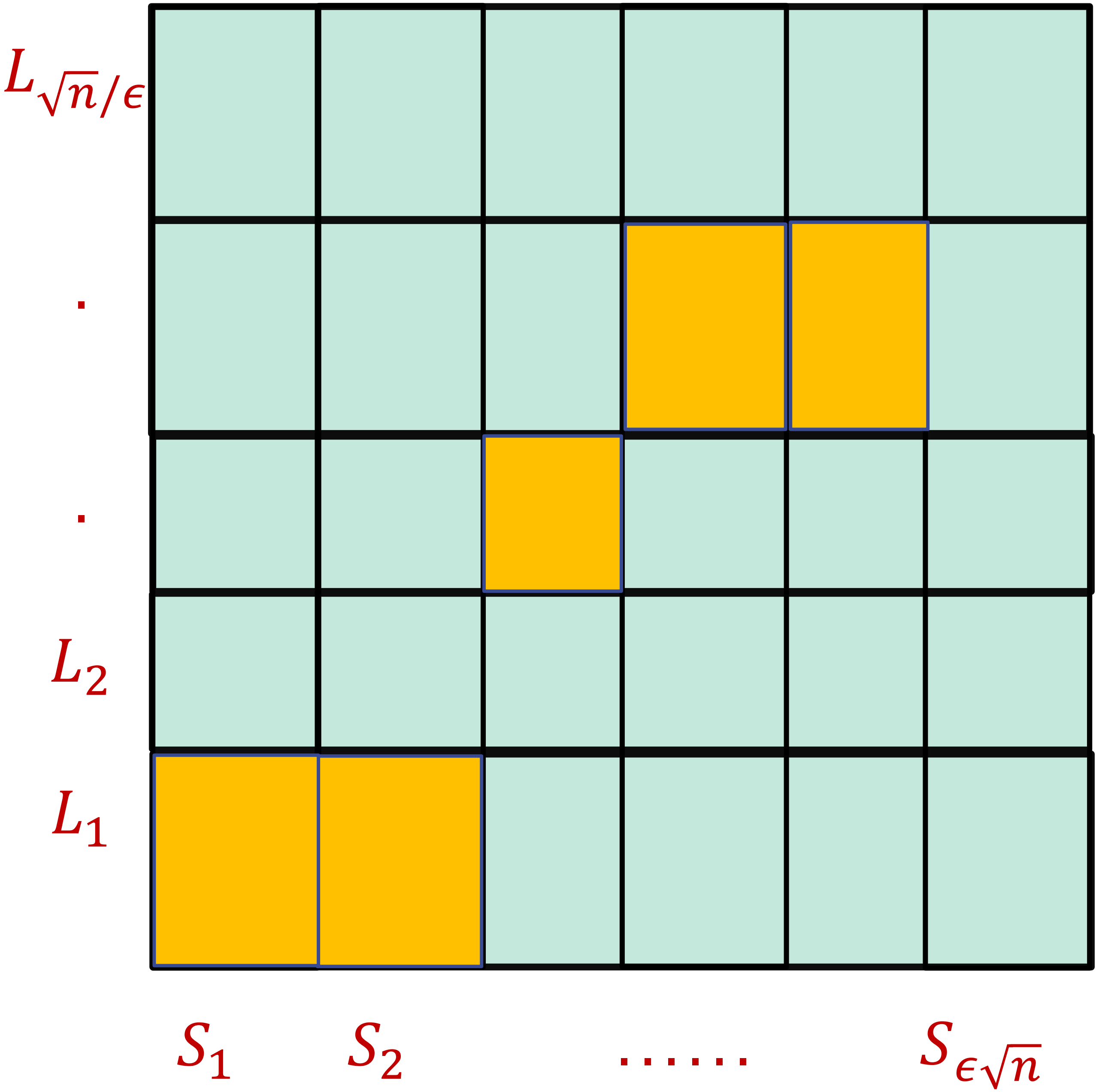
- It is "okay to delete" all horizontal blocks with more than ℓ boxes.



Horizontal chains

Let $\ell = \epsilon/\lambda^2$.

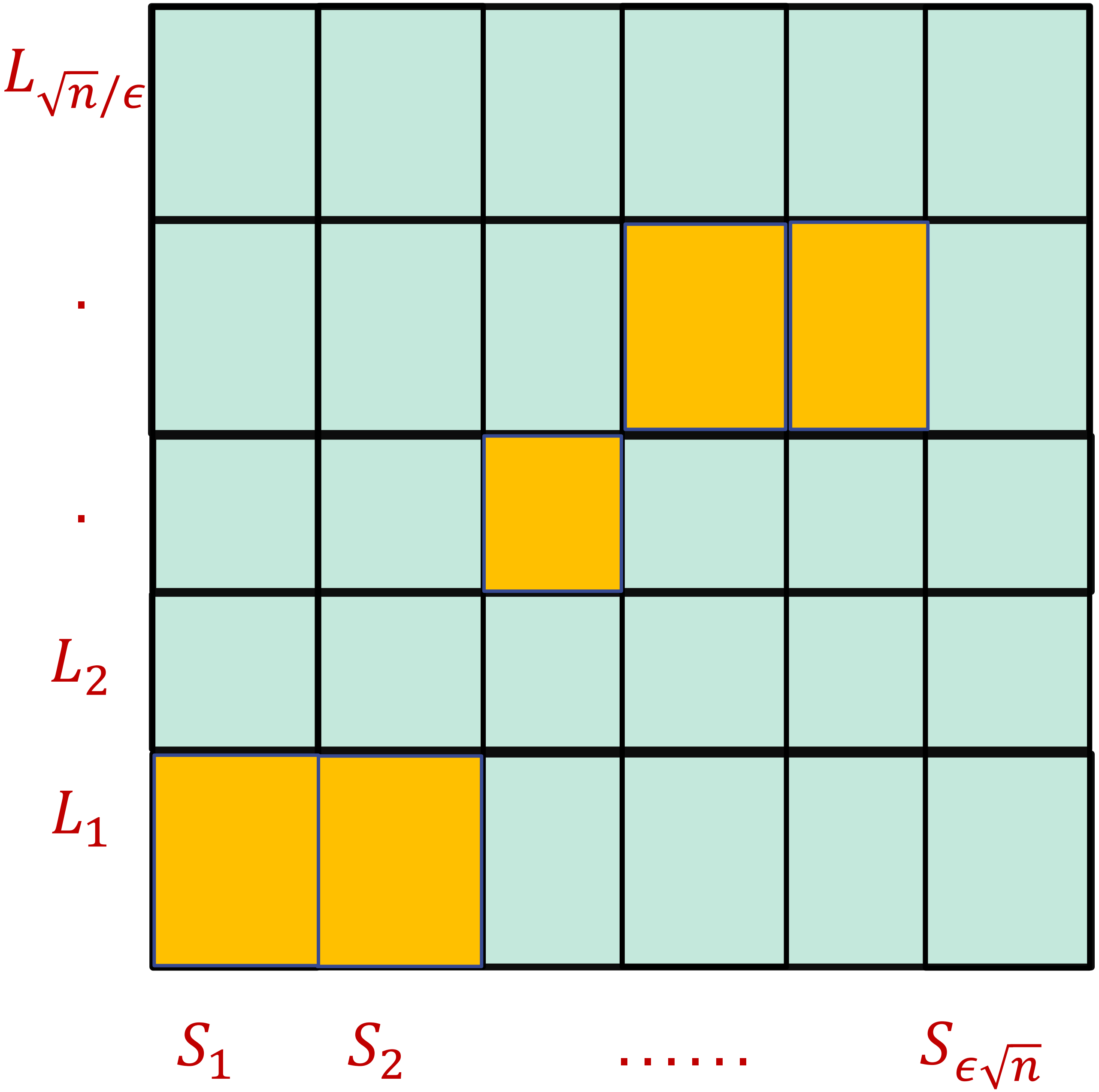
- It is "okay to delete" all horizontal blocks with more than ℓ boxes.
- Sample a constant number of remaining horizontal blocks, exactly compute the LIS in each



Horizontal chains

Let $\ell = \epsilon/\lambda^2$.

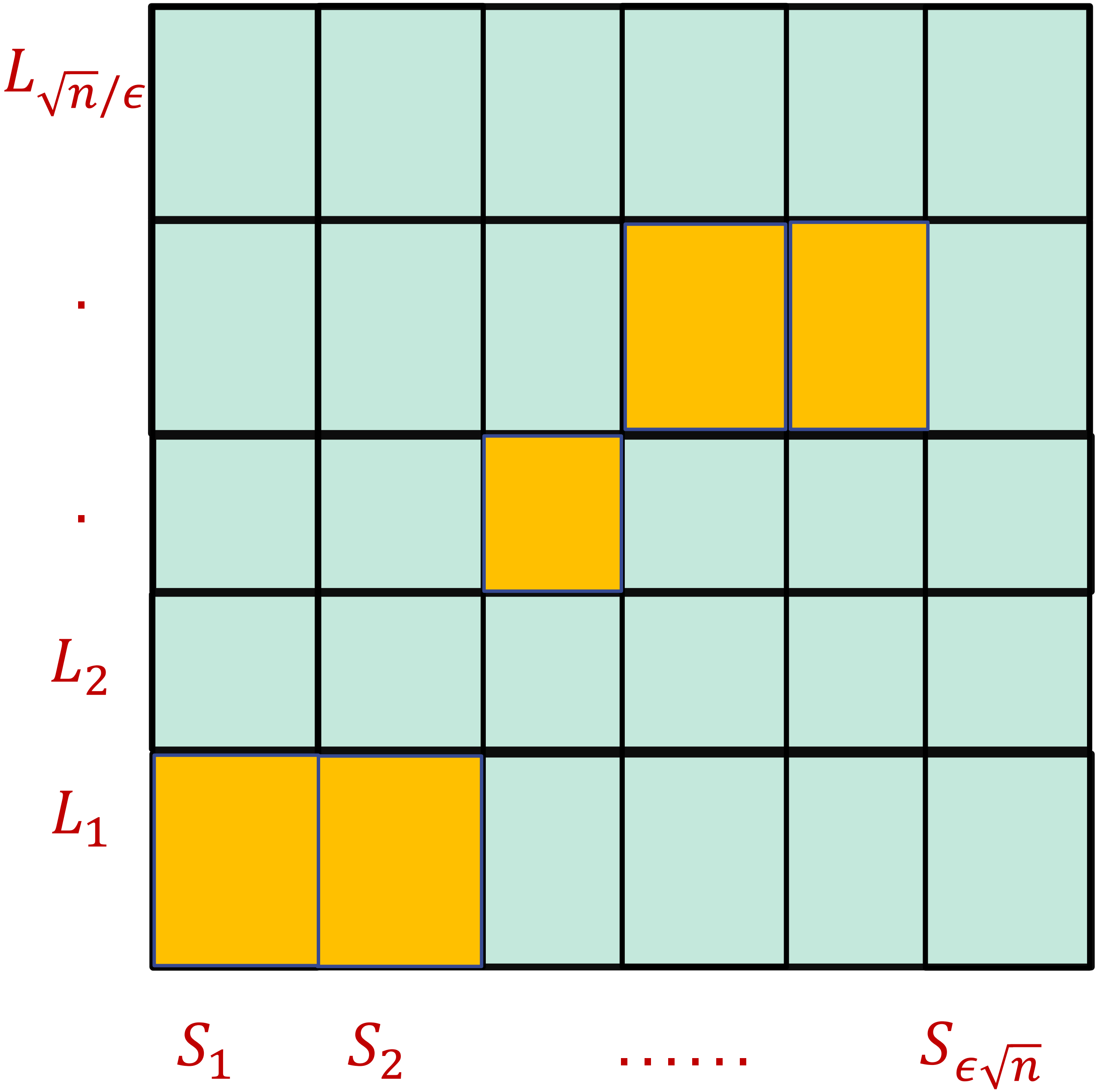
- It is "okay to delete" all horizontal blocks with more than ℓ boxes.
- Sample a constant number of remaining horizontal blocks, exactly compute the LIS in each by querying $O(\frac{\sqrt{n}}{\lambda^2})$ points per block,



Horizontal chains

Let $\ell = \epsilon/\lambda^2$.

- It is "okay to delete" all horizontal blocks with more than ℓ boxes.
- Sample a constant number of remaining horizontal blocks, exactly compute the LIS in each by querying $O(\frac{\sqrt{n}}{\lambda^2})$ points per block, and estimate the LIS length in the chain



Analysis idea

- **Query Complexity:** Layering, Tagging of boxes, Finer layering per vertical stripe, and estimating the length of horizontal and vertical chains overall take $\tilde{\Theta}(\sqrt{n} \cdot \text{poly}(\frac{1}{\lambda}))$ queries

Analysis idea

- **Query Complexity:** Layering, Tagging of boxes, Finer layering per vertical stripe, and estimating the length of horizontal and vertical chains overall take $\tilde{\Theta}(\sqrt{n} \cdot \mathrm{poly}(\frac{1}{\lambda}))$ queries

- **Approximation ratio:** Significant loss incurred is by restricting attention to $\Theta(\frac{1}{\lambda})$ chain of dense cells, by a factor of $\Omega(\lambda)$

LIS estimation algorithm

Let $r \leq n$ denote the number of distinct values in the input array

$\Omega(\lambda)$ multiplicative approximation, where $\lambda = \text{LIS}/n$	$O(\sqrt{r} \cdot \text{poly}(\frac{1}{\lambda}))$ **nonadaptive** queries

Summary of our contributions

- First nontrivial lower bound on the query complexity of LIS estimation algorithms
- Nonadaptive LIS estimation algorithms with better approximation guarantees than state of the art algorithms while providing the same running time
- Parameterizing the complexity of sublinear-time algorithms for LIS estimation
- Separation between erasure-resilient and tolerant testing models for the natural property of sortedness

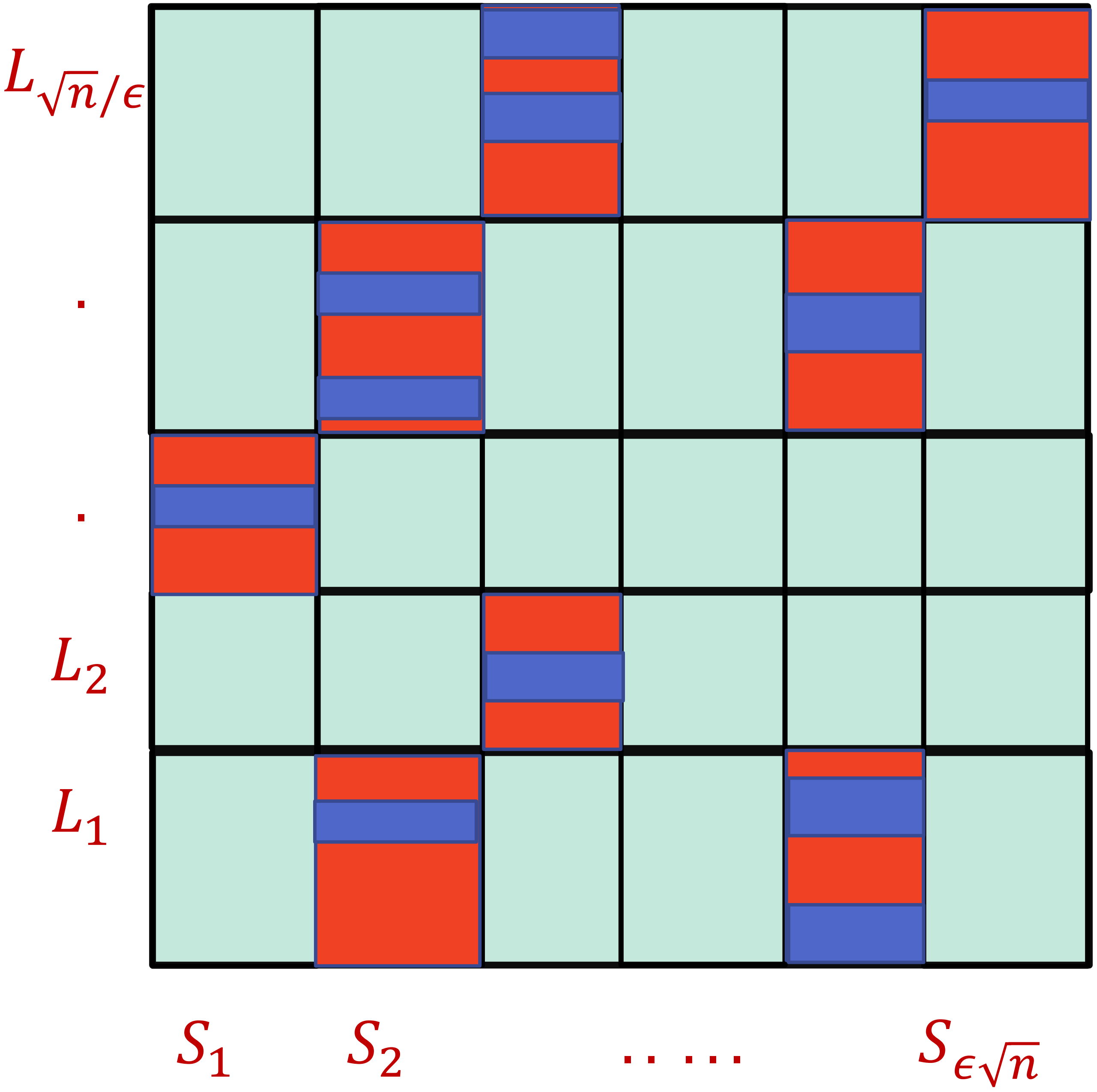
Open problems

- Good lower bounds for adaptive or nonadaptive LIS estimation tasks, with either approximation guarantees
- Improving the algorithm of Saks and Seshadhri in terms of:
  - Query complexity
  - Adaptivity

Thank you!

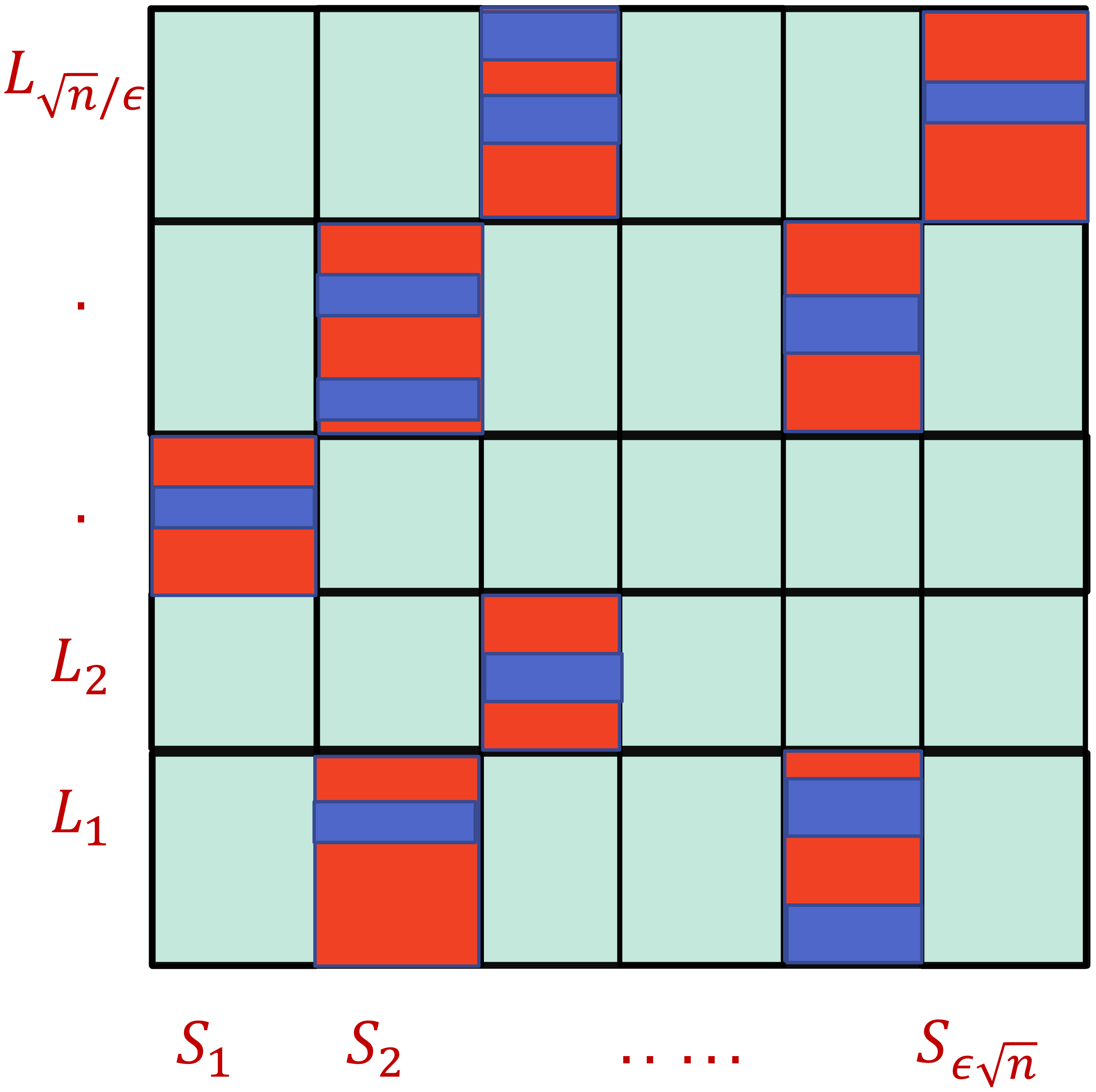
Finer layering

- Subdivide each dense box into cells of containing roughly β fraction of points in its stripe



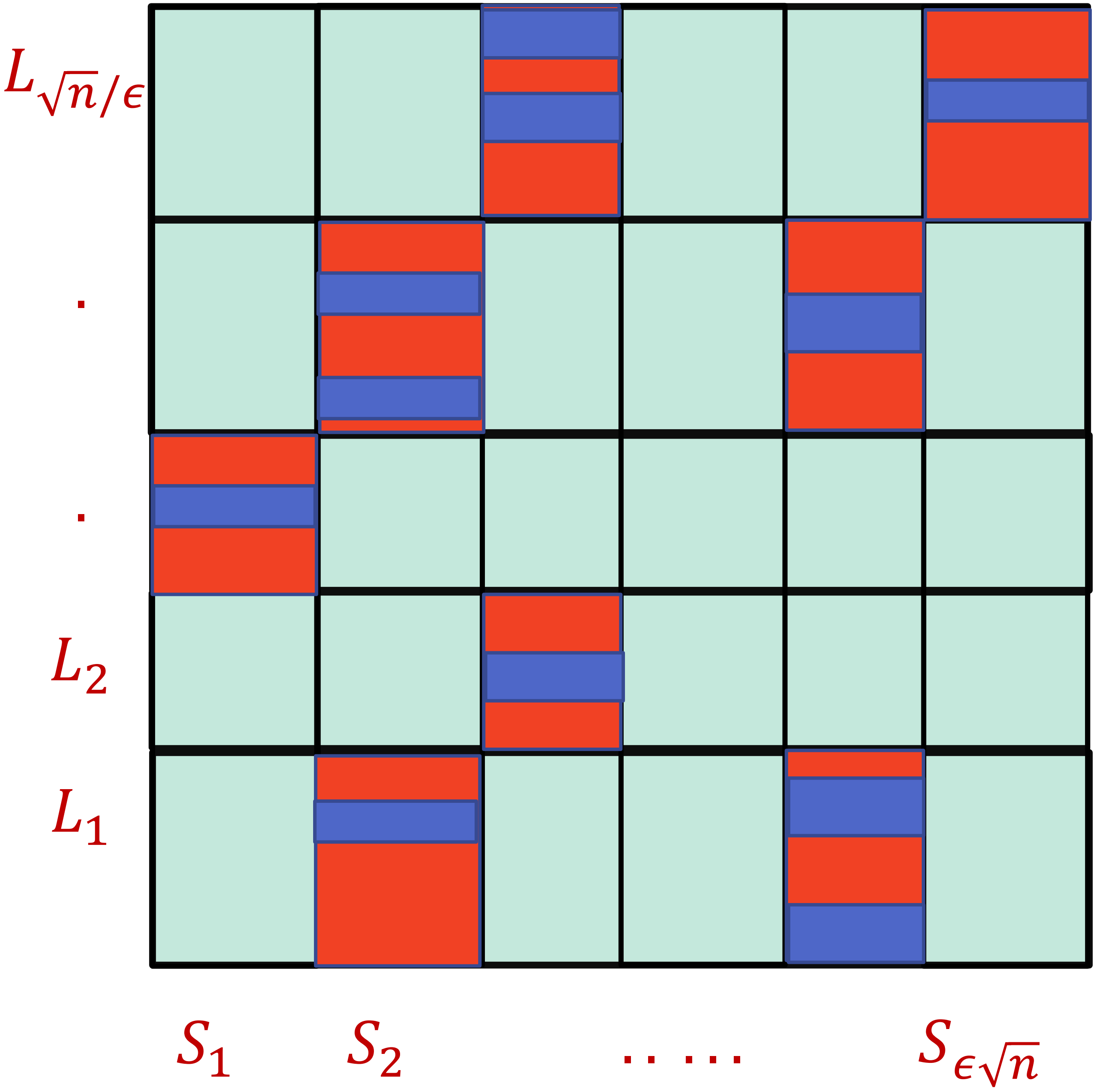
Finer layering

- Subdivide each dense box into cells of containing roughly β fraction of points in its stripe
  by making $\widetilde{\Theta}(\frac{1}{\beta})$ queries from each stripe.



Finer layering

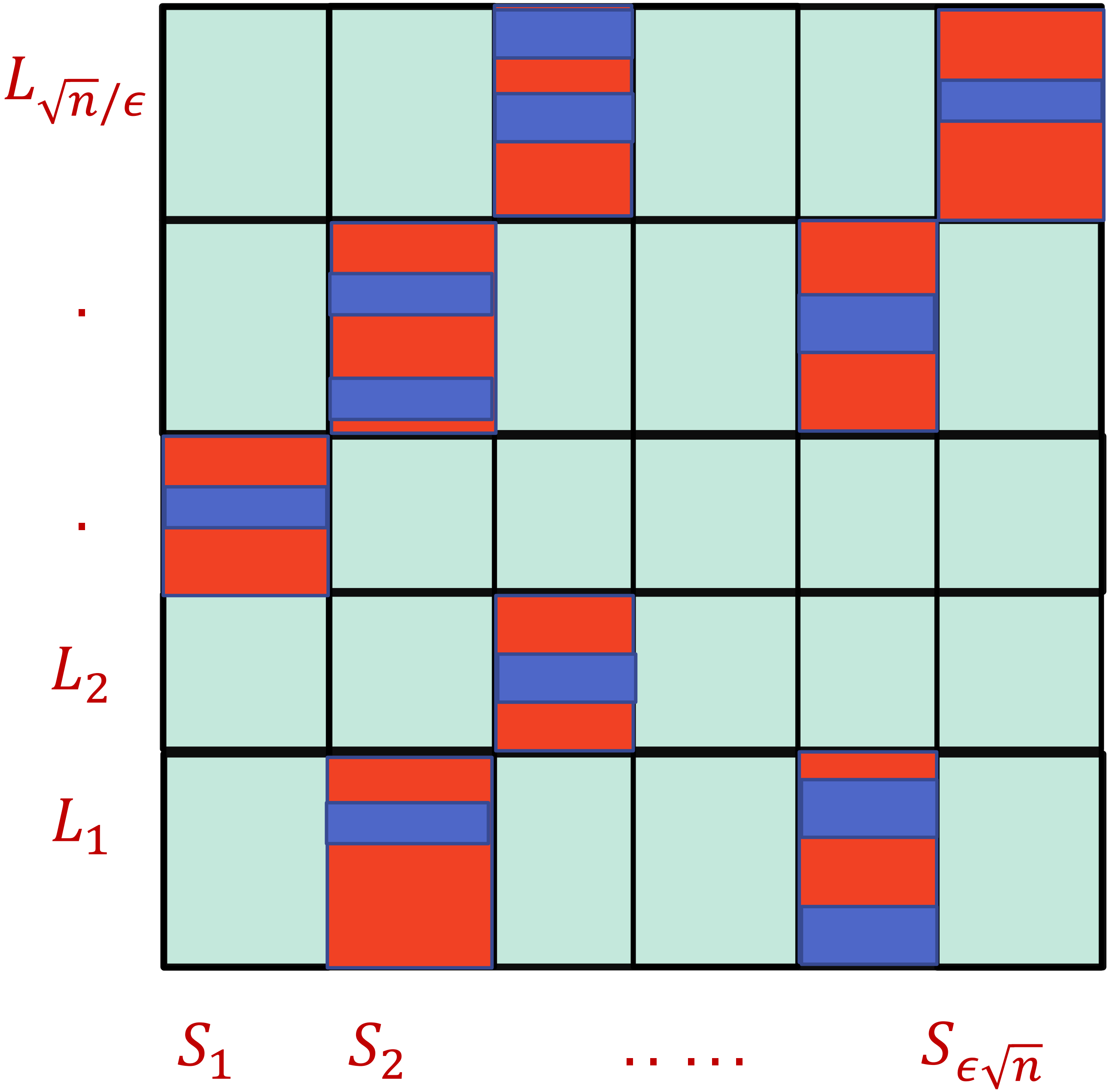
- Subdivide each dense box into cells of containing roughly β fraction of points in its stripe
  by making $\tilde{\Theta}(\frac{1}{\beta})$
  queries from each stripe.
- Overall $\tilde{\Theta}(\sqrt{n})$ queries



Two posets

$\langle P, \preccurlyeq \rangle$: Natural poset on dense boxes

$\langle P^\star, \preccurlyeq^\star \rangle$: Poset on dense cells



Two posets

$\langle P, \preccurlyeq \rangle$: Natural poset on dense boxes

$\langle P^{\star}, \preccurlyeq^{\star} \rangle$: Poset on dense cells

$$B_1 \preccurlyeq B_2$$

$$C_1 \preccurlyeq^{\star} C_2 \preccurlyeq^{\star} C_3$$

$$C_2 \preccurlyeq^{\star} C_4$$