Faster Min-Plus Product for Monotone Instances

Shucheng Chi, Ran Duan, Tianle Xie, **Tianyi Zhang**[1]

Tsinghua University Tel Aviv University

1. The fourth author is supported by funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 803118 UncertainENV)

Structured Min-Plus Problems

Min-Plus Product

- Given integral $n \times n$ matrices *A* and *B*, compute $(\min, +)$ product:

$$(A \star B)_{i,j} = \min_k \{A_{i,k} + B_{k,j}\}$$

- Min-Plus Product is equivalent to All-Pairs Shortest Paths [FM, 1971]
- Fastest runtime for $(\min, +)$ or APSP: $n^3/2^{\Omega(\sqrt{\log n})}$ [Williams, 2018]
- Hardness conjecture: $(\min, +)$ product requires $n^{3-o(1)}$ time

Structured Min-Plus Product

- If A and B have bounded entries $\in \{-W, \cdots, W, \infty\}$ then $A \star B$ can be computed in $\tilde{O}(Wn^{\omega})$ time [Alon et al., 1997]
- Inputs matrices with more structures:
  - Bounded-difference matrices [Bringmann et al., 2016]
  - Monotone matrices [Vassilevska Williams and Xu, 2020]

Bounded-Difference Min-Plus Product

- Matrix X has bounded-difference if:

$$|X_{i,j} - X_{i,j+1}|, |X_{i,j} - X_{i+1,j}| \leq 1$$

- Want to compute $A \star B$ when A and B both have bounded-difference
- Many applications in string problems:
  - Language edit distance, RNA folding [Bringmann et al., 2016]
  - Tree edit distance [Mao, 2021]
  - Dyck edit distance [Fried et al., 2022]

Monotone Min-Plus Product

- Matrix X is monotone if:

$$0 \leq X_{i,j} \leq X_{i,j+1} \leq O(n)$$

- Want to compute $A \star B$ when B is monotone
- Generalization of bounded-difference min-plus product [GPWX, 2021]
- Further application in graph problems:
  - Single-source replacement paths with negative weights [GPWX, 2021]

Min-Plus Convolution

- Given integral arrays *A* and *B* of length *n*, compute $(\min, +)$ convolution:
  $$(A \diamond B)_k = \min_i \{A_{k-i} + B_i\}$$
- Fastest runtime for $(\min, +)$ conv: $n^2/2^{\Omega(\sqrt{\log n})}$ [Williams, 2018]
- Hardness conjecture: $(\min, +)$ conv requires $n^{2-o(1)}$ time
  - Stronger than APSP or 3SUM conjecture

Monotone Min-Plus Convolution

- Array X is monotone if:

$$0 \leq X_i \leq X_{i+1} \leq O(n)$$

- Want to compute $A \diamond B$ when A and B both are monotone

- Applications:

  - Histogram indexing, necklace alignment [BCD+, 2006] [ACLL, 2014]

Runtime for Structured Min-Plus Instances

reference	bounded-difference min-plus product	monotone min-plus product	monotone min-plus convolution
baseline	n^3	n^3	n^2
Chan and Lewenstein 2015			$n^{1.859}$
Bringmann et al. 2016	$n^{2.824}$		
V. Williams and Xu 2020		$n^{(15+\omega)/6}$	
Gu et al. 2021		$n^{(12+\omega)/5}$	
Chi, Duan and Xie 2022	$n^{2+\omega/3}$		
new	$n^{(3+\omega)/2}$	$n^{(3+\omega)/2}$	$n^{1.5}$

FMM exponent $\omega < 2.373$ [Alman and Vassilevska Williams, 2021]

Runtime for Structured Min-Plus Instances

reference	bounded-difference min-plus product	monotone min-plus product	monotone min-plus convolution
baseline	n^3	n^3	n^2
Chan and Lewenstein 2015			$n^{1.859}$
Bringmann et al. 2016	$n^{2.824}$		
V. Williams and Xu 2020		$n^{(15+\omega)/6}$	
Gu et al. 2021		$n^{(12+\omega)/5}$	
Chi, Duan and Xie 2022	$n^{2+\omega/3}$		
new	$n^{(3+\omega)/2}$	$n^{(3+\omega)/2}$	$n^{1.5}$

FMM exponent $\omega < 2.373$ [Alman and Vassilevska Williams, 2021]

Monotone Min-Plus Product

with runtime $n^{2+\omega/3}$

Approximate Min-Plus

- For convenience, assume $\omega = 2$ for the rest
- Estimate $C = A \star B$ up to sub-linear additive errors
  A common step in previous works
- Rounding: $\tilde{A}_{i,j} = \lfloor A_{i,j}/n^{1/3} \rfloor$, $\tilde{B}_{i,j} = \lfloor B_{i,j}/n^{1/3} \rfloor$
  Compute: $\tilde{C} = \tilde{A} \star \tilde{B}$
- Approximation: $|\tilde{C}_{i,j} - C_{i,j}/n^{1/3}| = O(1)$
  Runtime: $\tilde{O}(n^{2+2/3})$

Quotient & Remainder

Basic idea: match quotients with $\tilde{C}$, then find minimum remainders

$$A_{i,k} = n^{1/3}\tilde{A}_{i,k} + \hat{A}_{i,k}$$

$$B_{k,j} = n^{1/3}\tilde{B}_{k,j} + \hat{B}_{k,j}$$

$$C_{i,j} = n^{1/3}\tilde{C}_{i,j} + ?$$

Quotient & Remainder

Basic idea: match quotients with $\tilde{C}$, then find minimum remainders

$$A_{i,k} = \boxed{n^{1/3}\tilde{A}_{i,k}} + \boxed{\hat{A}_{i,k}}$$

$$B_{k,j} = \boxed{n^{1/3}\tilde{B}_{k,j}} + \boxed{\hat{B}_{k,j}}$$

$$C_{i,j} = \boxed{n^{1/3}\tilde{C}_{i,j}} + \boxed{?}$$

quotients remainder

Quotient & Remainder

Basic idea: match quotients with $\tilde{C}$, then find minimum remainders

$$A_{i,k} = \boxed{n^{1/3}\tilde{A}_{i,k}} + \boxed{\hat{A}_{i,k}}$$

$$B_{k,j} = \boxed{n^{1/3}\tilde{B}_{k,j}} + \boxed{\hat{B}_{k,j}}$$

$$C_{i,j} = \boxed{n^{1/3}\tilde{C}_{i,j}} + \boxed{?}$$

quotients remainder

Only focus on $k \in [n]$ such that:

$$|\tilde{A}_{i,k} + \tilde{B}_{k,j} - \tilde{C}_{i,j}| = O(1)$$

and then minimize:

$$\hat{A}_{i,k} + \hat{B}_{k,j}$$

Quotient & Remainder

Basic idea: match quotients with $\tilde{C}$, then find minimum remainders

$$A_{i,k} = \boxed{n^{1/3}\tilde{A}_{i,k}} + \boxed{\hat{A}_{i,k}}$$

$$B_{k,j} = \boxed{n^{1/3}\tilde{B}_{k,j}} + \boxed{\hat{B}_{k,j}}$$

$$C_{i,j} = \boxed{n^{1/3}\tilde{C}_{i,j}} + \boxed{?}$$

quotients remainder

Using polynomials:

$$A_{i,k}(x, y) = x^{\hat{A}_{i,k}} \cdot y^{\tilde{A}_{i,k}}$$

$$B_{k,j}(x, y) = x^{\hat{B}_{k,j}} \cdot y^{\tilde{B}_{k,j}}$$

$$C_{i,j}(x, y) = \sum_{k=1}^{n} (A_{i,k} \cdot B_{k,j})(x, y)$$

Quotient & Remainder

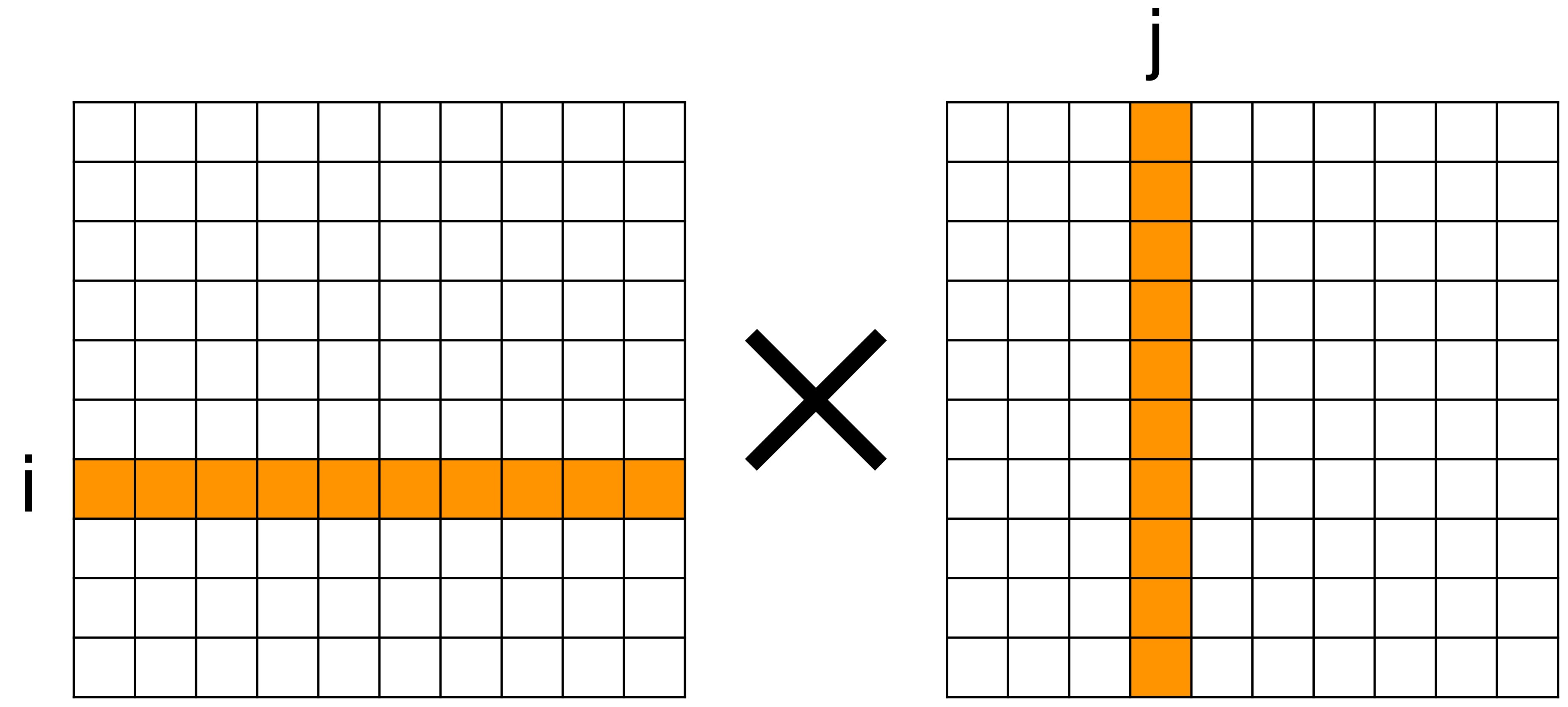
Basic idea: match quotients with $\tilde{C}$, then find minimum remainders

Using polynomials:

$$A_{i,k}(x, y) = x^{\hat{A}_{i,k}} \cdot y^{\tilde{A}_{i,k}}$$

$$B_{k,j}(x, y) = x^{\hat{B}_{k,j}} \cdot y^{\tilde{B}_{k,j}}$$

$$C_{i,j}(x, y) = \sum_{k=1}^{n} (A_{i,k} \cdot B_{k,j})(x, y)$$



$$C_{i,j}(x, y) = \sum_{k=1}^{n} (A_{i,k} \cdot B_{k,j})(x, y)$$

$$= y^{\tilde{C}_{i,j}} F_0(x) + y^{\tilde{C}_{i,j}+1} F_1(x) + y^{\tilde{C}_{i,j}+2} F_2(x) + \cdots\cdots$$

Quotient & Remainder

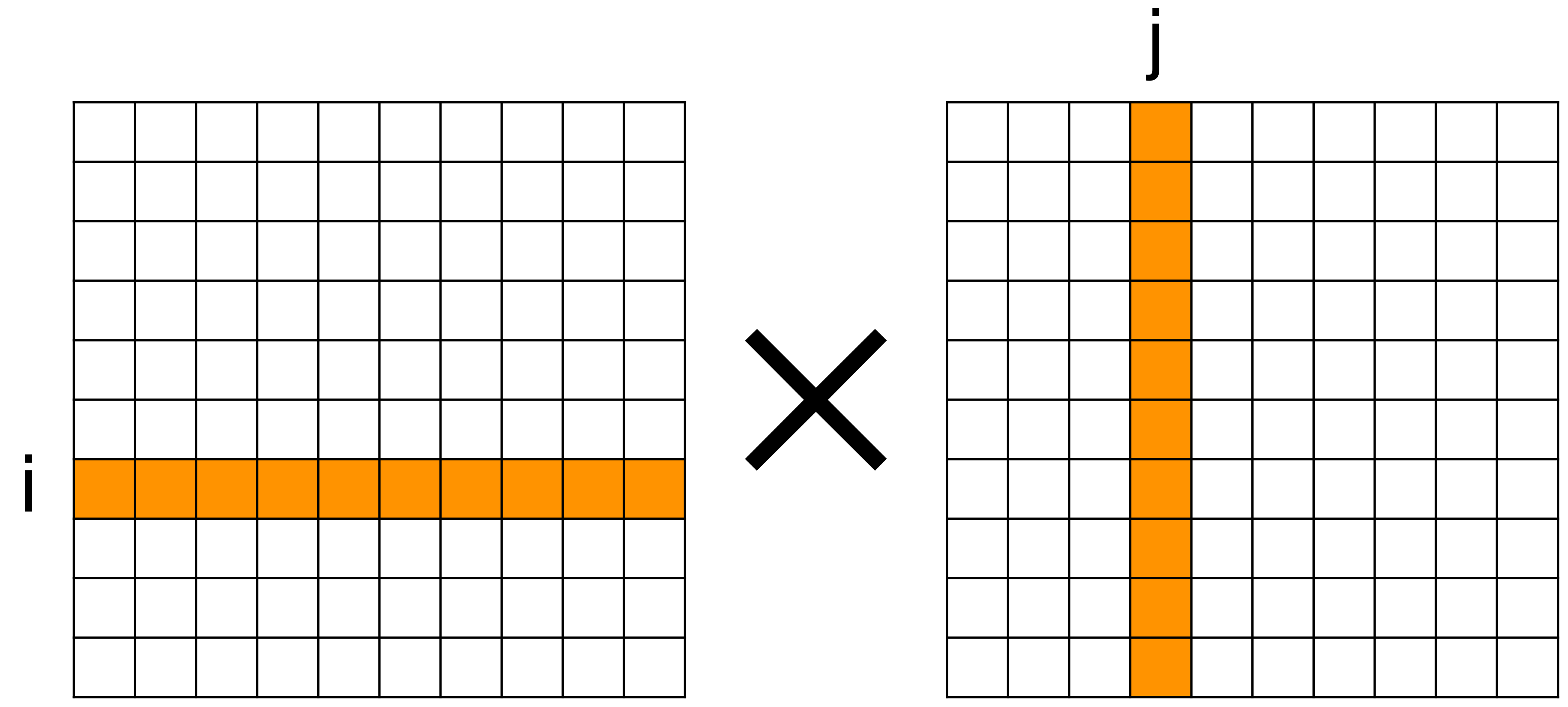
Basic idea: match quotients with $\tilde{C}$, then find minimum remainders

Only focus on $k \in [n]$ such that:

$$|\tilde{A}_{i,k} + \tilde{B}_{k,j} - \tilde{C}_{i,j}| = O(1)$$

and then minimize:

$$\hat{A}_{i,k} + \hat{B}_{k,j}$$



$$C_{i,j}(x,y) = \sum_{k=1}^{n} (A_{i,k} \cdot B_{k,j})(x,y)$$

$$= y^{\tilde{C}_{i,j}} F_0(x) + y^{\tilde{C}_{i,j}+1} F_1(x) + y^{\tilde{C}_{i,j}+2} F_2(x) + \cdots\cdots$$

Quotient & Remainder

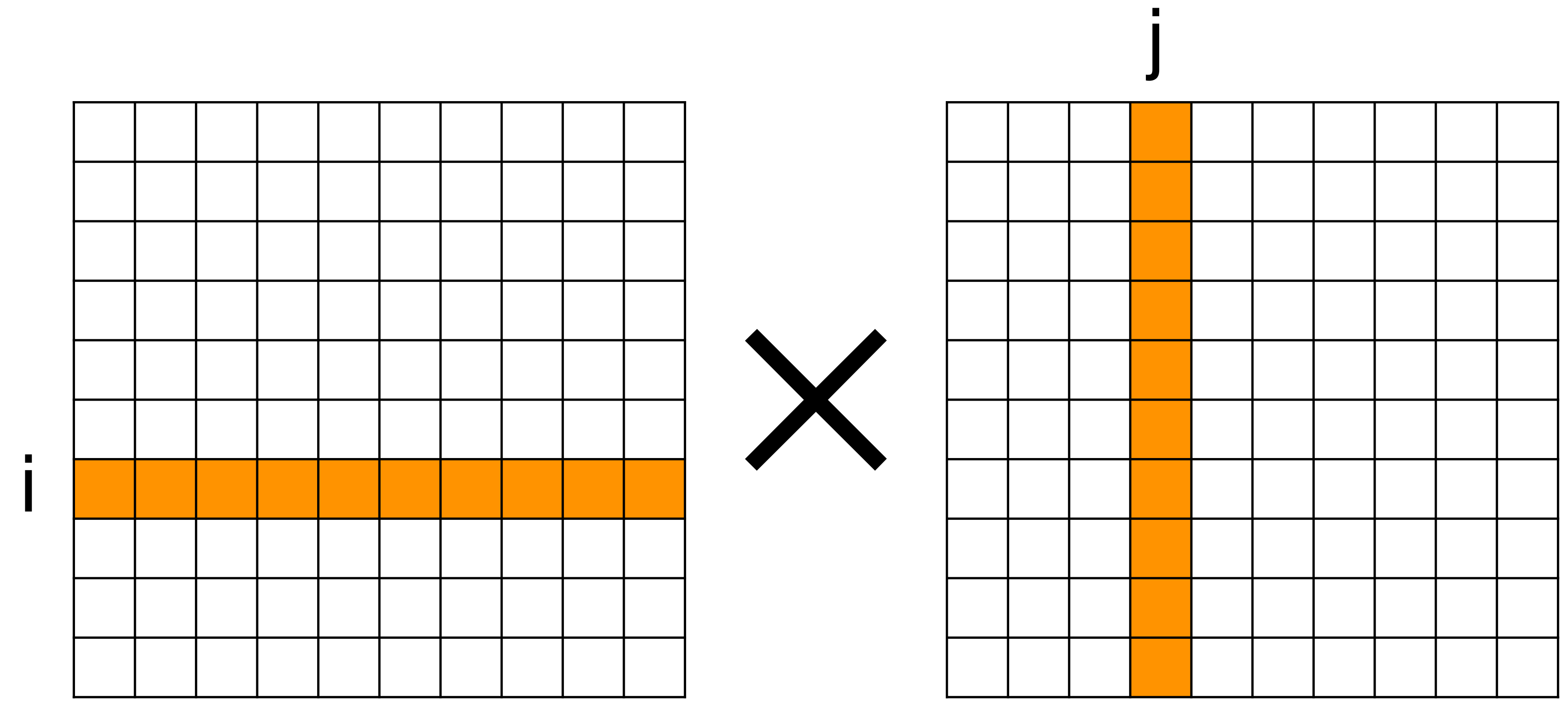
Basic idea: match quotients with $\tilde{C}$, then find minimum remainders

Only focus on $k \in [n]$ such that:

$$|\tilde{A}_{i,k} + \tilde{B}_{k,j} - \tilde{C}_{i,j}| = O(1)$$

and then minimize:

$$\hat{A}_{i,k} + \hat{B}_{k,j}$$



$$C_{i,j}(x,y) = \sum_{k=1}^{n} (A_{i,k} \cdot B_{k,j})(x,y)$$

$$= y^{\tilde{C}_{i,j}} F_0(x) + y^{\tilde{C}_{i,j}+1} F_1(x) + \cancel{y^{\tilde{C}_{i,j}+2} F_2(x) + \cdots\cdots}$$

Quotient & Remainder

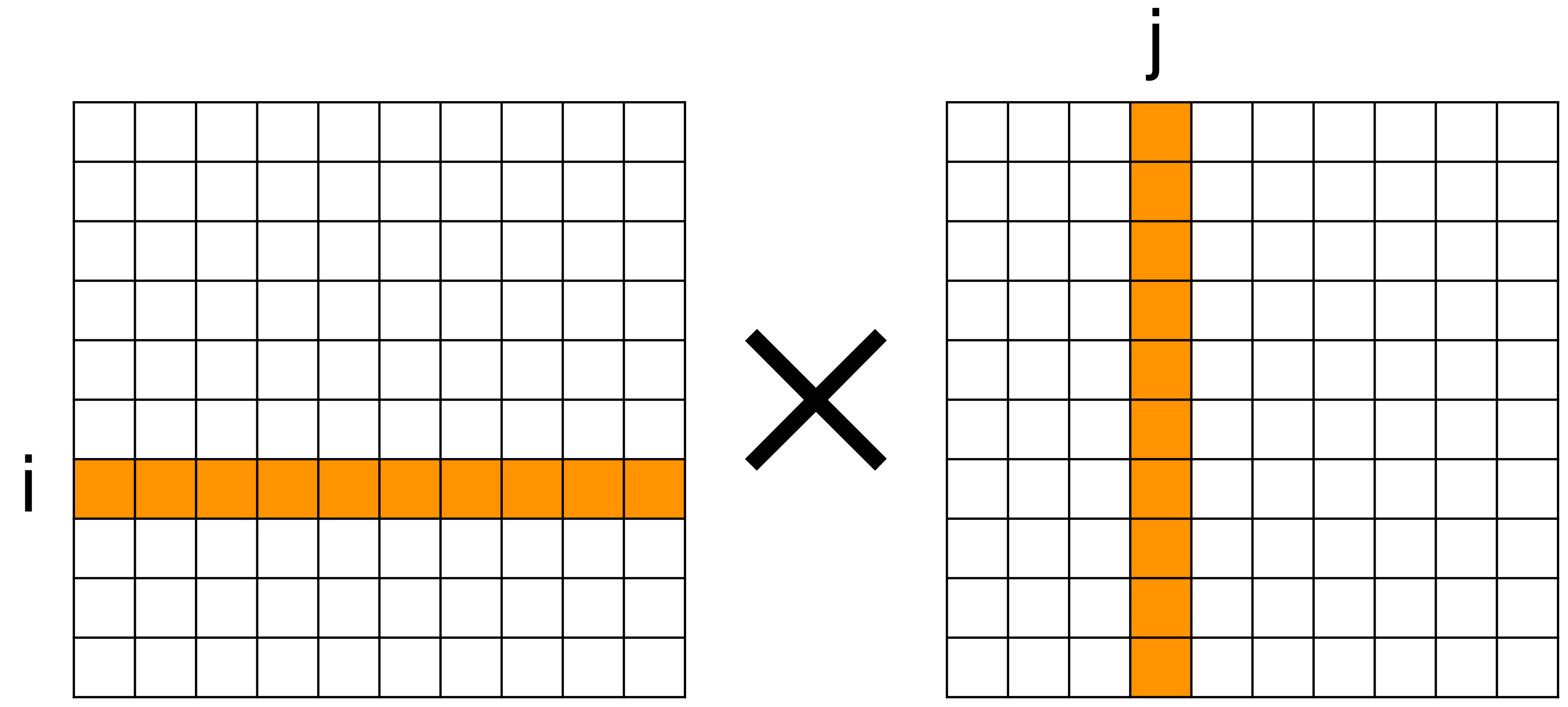
Basic idea: match quotients with $\tilde{C}$, then find minimum remainders

Only focus on $k \in [n]$ such that:

$$|\tilde{A}_{i,k} + \tilde{B}_{k,j} - \tilde{C}_{i,j}| = O(1)$$

and then minimize:

$$\hat{A}_{i,k} + \hat{B}_{k,j}$$



$$C_{i,j}(x,y) = \sum_{k=1}^{n} (A_{i,k} \cdot B_{k,j})(x,y)$$

$$= y^{\tilde{C}_{i,j}} F_0(x) + y^{\tilde{C}_{i,j}+1} F_1(x) + \cancel{y^{\tilde{C}_{i,j}+2} F_2(x) + \cdots\cdots}$$

$C_{i,j} = n^{1/3}(\tilde{C}_{i,j} + b) +$ **min-deg of** $F_b(x)$

Quotient & Remainder

Basic idea: match quotients with $\tilde{C}$, then find minimum remainders

Using polynomials:

$$A_{i,k}(x,y) = x^{\hat{A}_{i,k}} \cdot y^{\tilde{A}_{i,k}}$$

$$B_{k,j}(x,y) = x^{\hat{B}_{k,j}} \cdot y^{\tilde{B}_{k,j}}$$

$$C_{i,j}(x,y) = \sum_{k=1}^{n} (A_{i,k} \cdot B_{k,j})(x,y)$$

Runtime? No help

Multiplying polynomial matrices

$$C(x,y) = A(x,y) \cdot B(x,y)$$

takes time

$$n^{\omega=2} \cdot \underbrace{\deg_x}_{= n^{1/3}} \cdot \underbrace{\deg_y}_{= n^{2/3}} = n^3$$

Modulo on y-degrees

Key idea: apply modulo-p operations on degrees of y-variables

Using polynomials:

$$A_{i,k}(x,y) = x^{\hat{A}_{i,k}} \cdot y^{\tilde{A}_{i,k}}$$

$$B_{k,j}(x,y) = x^{\hat{B}_{k,j}} \cdot y^{\tilde{B}_{k,j}}$$

$$C_{i,j}(x,y) = \sum_{k=1}^{n} (A_{i,k} \cdot B_{k,j})(x,y)$$

Degree reduction by modulo:

$$A^{p}_{i,k}(x,y) = x^{\hat{A}_{i,k}} \cdot y^{\tilde{A}_{i,k} \mod p}$$

$$B^{p}_{k,j}(x,y) = x^{\hat{B}_{k,j}} \cdot y^{\tilde{B}_{k,j} \mod p}$$

$$C^{p}_{i,j}(x,y) = \sum_{k=1}^{n} (A^{p}_{i,k} \cdot B^{p}_{k,j})(x,y)$$

Modulo on y-degrees

Key idea: apply modulo-p operations on degrees of y-variables

Degree reduction by modulo:

$$A^{p}_{i,k}(x,y) = x^{\hat{A}_{i,k}} \cdot y^{\tilde{A}_{i,k} \mod p}$$

$$B^{p}_{k,j}(x,y) = x^{\hat{B}_{k,j}} \cdot y^{\tilde{B}_{k,j} \mod p}$$

$$C^{p}_{i,j}(x,y) = \sum_{k=1}^{n} (A^{p}_{i,k} \cdot B^{p}_{k,j})(x,y)$$

Runtime?

Multiplying polynomial matrices

$$C^{p}(x,y) = A^{p}(x,y) \cdot B^{p}(x,y)$$

takes time

$$n^{\omega=2} \cdot \underbrace{\deg_x}_{=\,n^{1/3}} \cdot p = n^{2+1/3} p$$

Modulo on y-degrees

Key idea: apply modulo-p operations on degrees of y-variables

Only focus on $k \in [n]$ s.t. $|\tilde{A}_{i,k} + \tilde{B}_{k,j} - \tilde{C}_{i,j}| = O(1)$

$$C_{i,j}(x, y) = y^{\tilde{C}_{i,j}} F_0(x) + y^{\tilde{C}_{i,j}+1} F_1(x) + y^{\tilde{C}_{i,j}+2} F_2(x) + \cdots + y^{\tilde{C}_{i,j}+p} F_p(x) + y^{\tilde{C}_{i,j}+p+1} F_p(x) + \cdots\cdots$$

Taking modulo-p adds many erroneous x-monomials

$$C^p_{i,j}(x, y) = y^{\tilde{C}_{i,j} \mod p}[F_0(x) + F_p(x)] + y^{\tilde{C}_{i,j}+1 \mod p}[F_1(x) + F_{1+p}(x)] + y^{\tilde{C}_{i,j}+2 \mod p} F_2(x) + \cdots\cdots$$

In other words, many x-monomials are hashed to the same y-bucket

Bounding total errors

The number of erroneous x-terms:

If $p \in [n^{1/3}, 2n^{1/3}]$ is a random prime, then total #errors $= \tilde{O}(n^{3-1/3})$

Proof:

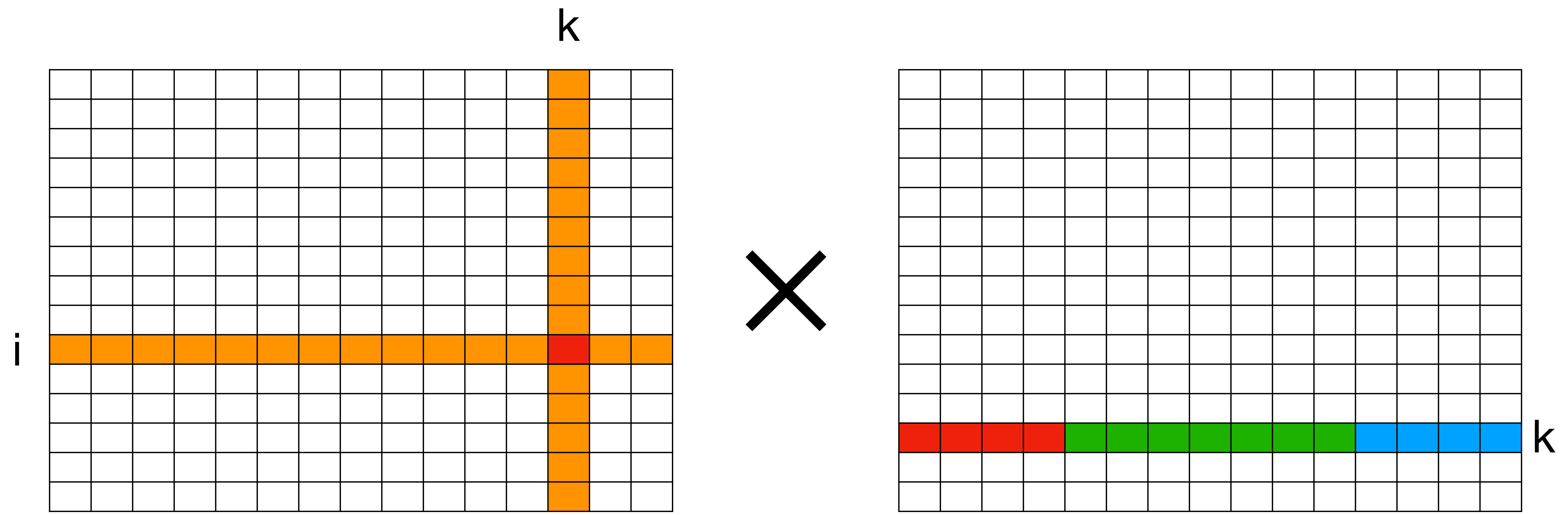
- Fix any $i, j, k \in [n]$ s.t. $|\tilde{A}_{i,k} + \tilde{B}_{k,j} - \tilde{C}_{i,j}| \neq O(1)$
  What is the probability that $\tilde{A}_{i,k} + \tilde{B}_{k,j} - \tilde{C}_{i,j} \equiv O(1) \mod p$
- As $p \in [n^{1/3}, 2n^{1/3}]$ is random, the probability is $\tilde{O}(n^{-1/3})$

Finding all error monomials

Goal:
Fix a prime $p \in [n^{1/3}, 2n^{1/3}]$, find all $(i, j, k) \in [n]^3$ such that:

$$|\tilde{A}_{i,k} + \tilde{B}_{k,j} - \tilde{C}_{i,j}| \neq O(1)$$

$$\tilde{A}_{i,k} + \tilde{B}_{k,j} - \tilde{C}_{i,j} \equiv O(1) \mod p$$

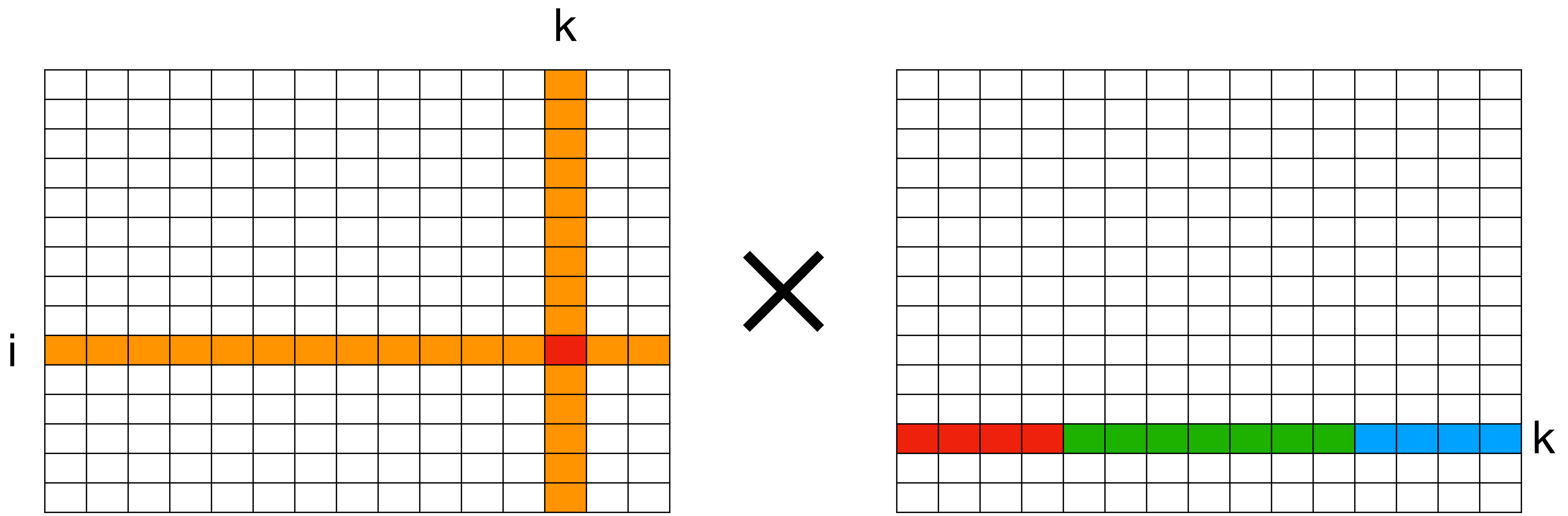


Finding all error monomials

Goal:
Fix a prime
$p \in [n^{1/3}, 2n^{1/3}]$, find all
$(i, j, k) \in [n]^3$ such that:

$$|\tilde{A}_{i,k} + \tilde{B}_{k,j} - \tilde{C}_{i,j}| \neq O(1)$$

$$\tilde{A}_{i,k} + \tilde{B}_{k,j} - \tilde{C}_{i,j} \equiv O(1) \mod p$$



Divide each row into intervals

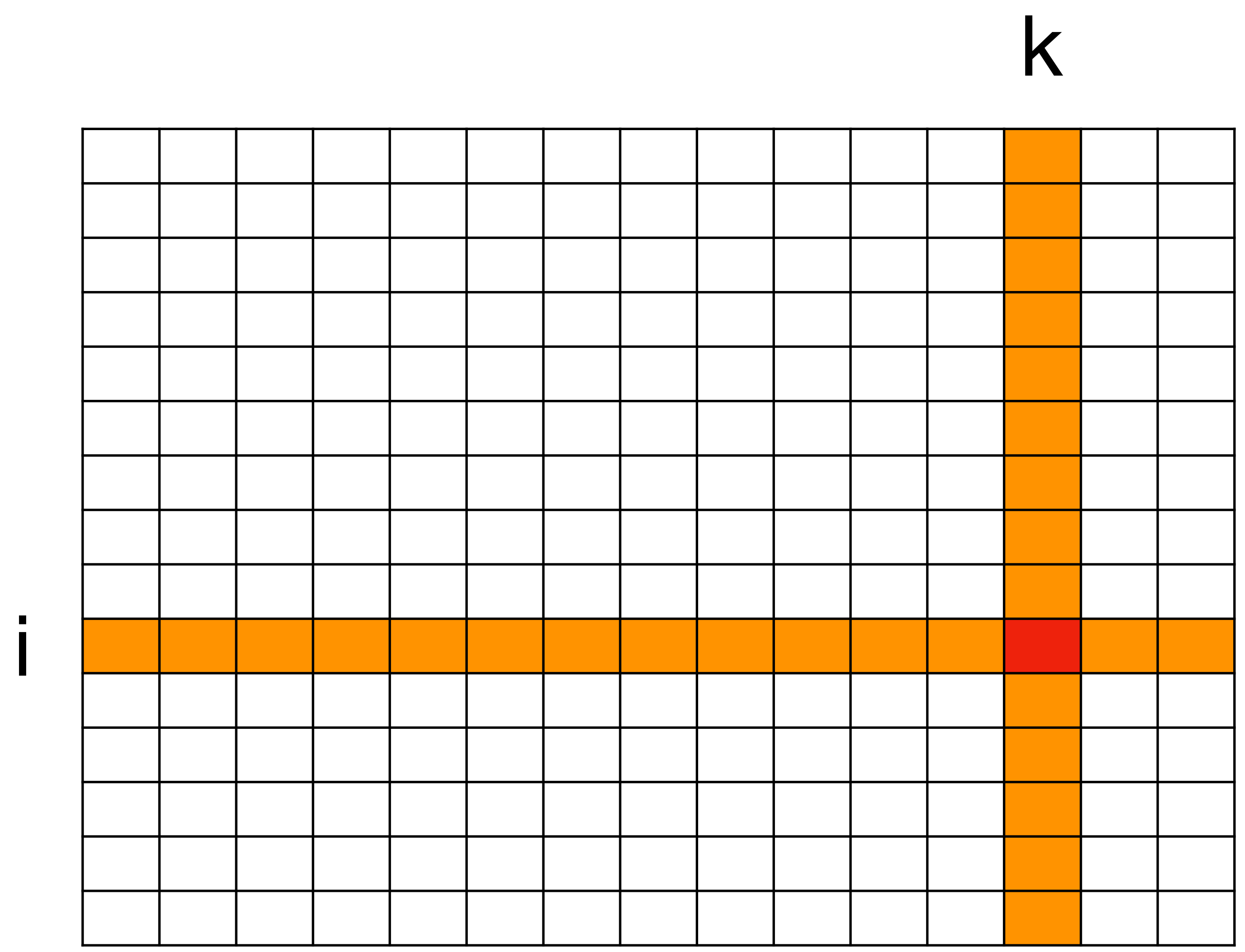
Entries in the each interval are the same

Finding all error monomials

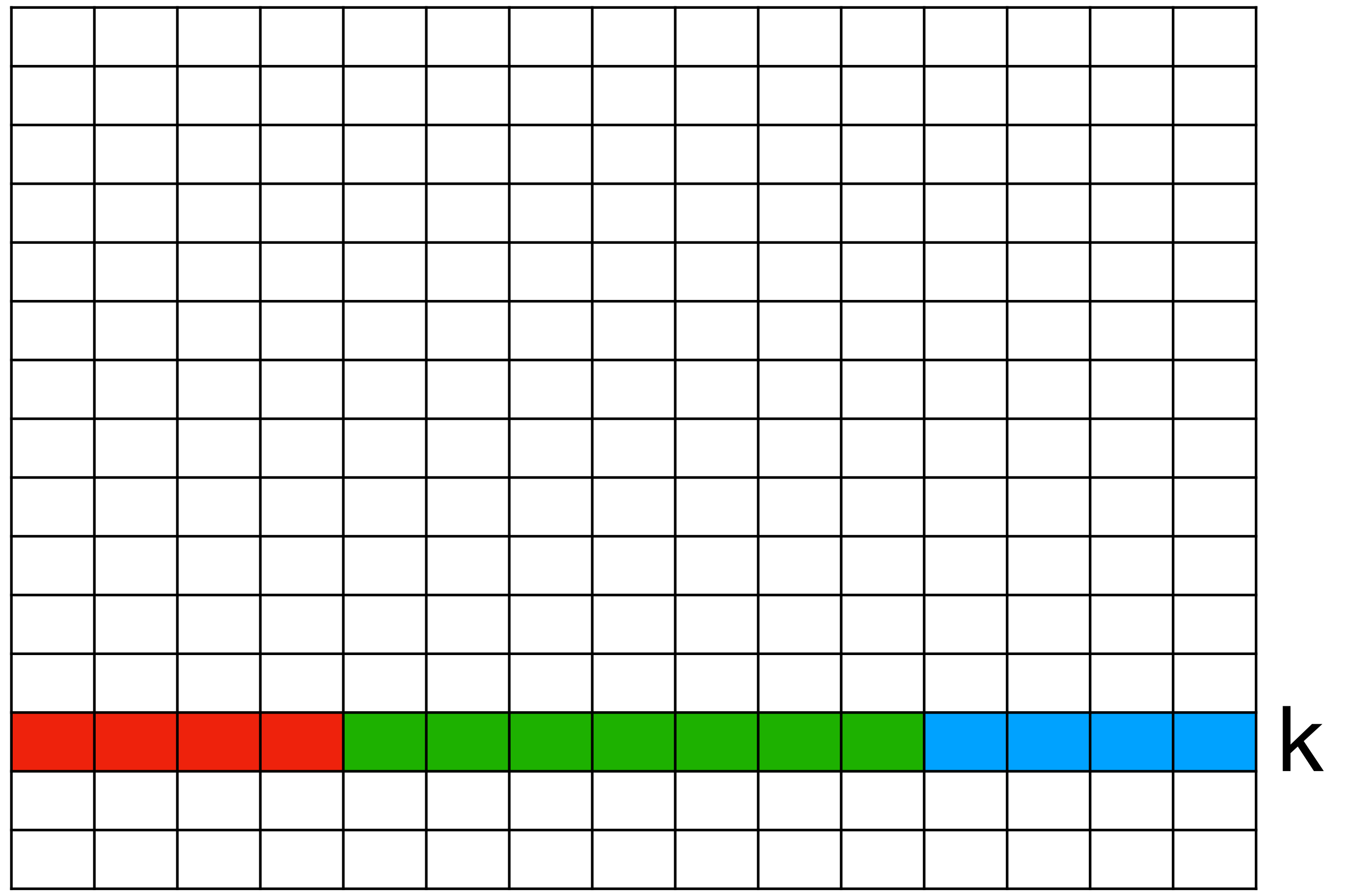
Goal:
Fix a prime
$p \in [n^{1/3}, 2n^{1/3}]$, find all $(i,j,k) \in [n]^3$ such that:

$$|\tilde{A}_{i,k} + \tilde{B}_{k,j} - \tilde{C}_{i,j}| \neq O(1)$$

$$\tilde{A}_{i,k} + \tilde{B}_{k,j} - \tilde{C}_{i,j} \equiv O(1) \mod p$$



$\times$



Divide each row into intervals

Entries in the each interval are the same

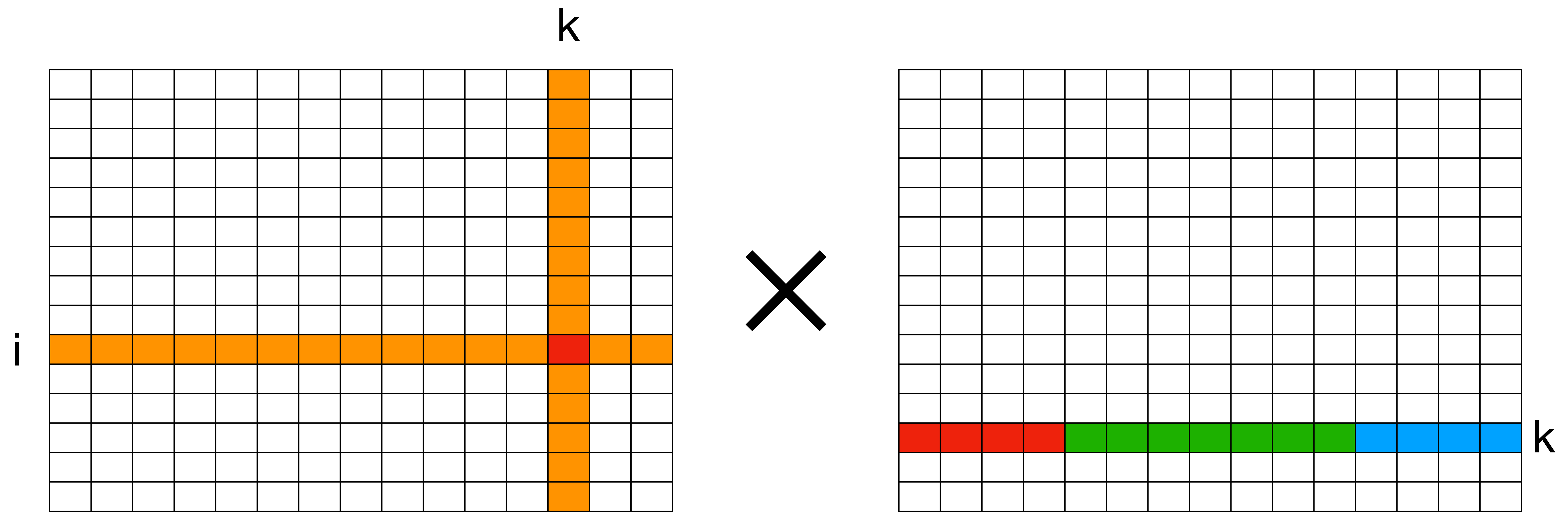
Since $\tilde{B}$ is monotone, there are at most $n^{2/3}$ intervals for each row

Finding all error monomials

Goal:
Fix a prime $p \in [n^{1/3}, 2n^{1/3}]$, find all $(i, j, k) \in [n]^3$ such that:

$$|\tilde{A}_{i,k} + \tilde{B}_{k,j} - \tilde{C}_{i,j}| \neq O(1)$$

$$\tilde{A}_{i,k} + \tilde{B}_{k,j} - \tilde{C}_{i,j} \equiv O(1) \mod p$$



Algorithm:
- Fix any $i, k \in [n]$ and any interval of k-th row of B
- List all j in the interval such that
$$\tilde{C}_{i,j} \neq \tilde{A}_{i,k} + \tilde{B}_{k,j} - O(1)$$
$$\tilde{C}_{i,j} \equiv \tilde{A}_{i,k} + \tilde{B}_{k,j} - O(1) \mod p$$
- Total runtime = $\tilde{O}(n^{2+2/3})$

Divide each row into **intervals**

Entries in the each interval are the same

Since $\tilde{B}$ is monotone, there are at most $n^{2/3}$ intervals for each row

Monotone Min-Plus Product

with runtime $n^{(3+\omega)/2}$

Recursion: first attempt

<u>Basic idea:</u>

Instead of brute-force, compute the approx-matrix $\tilde{C} = \tilde{A} \star \tilde{B}$ by recursion

Recursion: first attempt

Basic idea:

Instead of brute-force, compute the approx-matrix $\tilde{C} = \tilde{A} \star \tilde{B}$ by recursion

- Previously: $A_{i,k} = n^{1/3}\tilde{A}_{i,k} + \hat{A}_{i,k}$, $B_{k,j} = n^{1/3}\tilde{B}_{k,j} + \hat{B}_{k,j}$

$$\tilde{C} = \tilde{A} \star \tilde{B}, \text{ runtime} = n^{2+2/3}$$

$$C^p(x,y) = A^p(x,y) \cdot B^p(x,y), \text{ runtime} = n^{\omega=2} \cdot \deg_x \cdot p = n^{2+2/3}$$

Recursion: first attempt

Basic idea:

Instead of brute-force, compute the approx-matrix $\tilde{C} = \tilde{A} \star \tilde{B}$ by recursion

- Previously: $A_{i,k} = n^{1/3}\tilde{A}_{i,k} + \hat{A}_{i,k}\ ,\ \ B_{k,j} = n^{1/3}\tilde{B}_{k,j} + \hat{B}_{k,j}$

  $\tilde{C} = \tilde{A} \star \tilde{B}$, runtime = $n^{2+2/3}$

  $C^p(x,y) = A^p(x,y) \cdot B^p(x,y)$, runtime = $n^{\omega=2} \cdot \deg_x \cdot p = n^{2+2/3}$

- Recursion: $A_{i,k} = 2\tilde{A}_{i,k} + \hat{A}_{i,k}\ ,\ \ B_{k,j} = 2\tilde{B}_{k,j} + \hat{B}_{k,j}$

  $\tilde{C} = \tilde{A} \star \tilde{B}$, by recursion

  $C^p(x,y) = A^p(x,y) \cdot B^p(x,y)$, runtime = $n^{\omega=2} \cdot \underbrace{\deg_x}_{=\,O(1)} \cdot p = n^2 p$

Recursion: first attempt

Basic idea:

Instead of brute-force, compute the approx-matrix $\tilde{C} = \tilde{A} \star \tilde{B}$ by recursion

- Previously: $A_{i,k} = n^{1/3}\tilde{A}_{i,k} + \hat{A}_{i,k}$, $B_{k,j} = n^{1/3}\tilde{B}_{k,j} + \hat{B}_{k,j}$

  $\tilde{C} = \tilde{A} \star \tilde{B}$, runtime = $n^{2+2/3}$

  $C^p(x,y) = A^p(x,y) \cdot B^p(x,y)$, runtime = $n^{\omega=2} \cdot \deg_x \cdot p = n^{2+2/3}$

- Recursion: $A_{i,k} = 2\tilde{A}_{i,k} + \hat{A}_{i,k}$, $B_{k,j} = 2\tilde{B}_{k,j} + \hat{B}_{k,j}$

  $\tilde{C} = \tilde{A} \star \tilde{B}$, by recursion

  $C^p(x,y) = A^p(x,y) \cdot B^p(x,y)$, runtime = $n^{\omega=2} \cdot \underset{\substack{/\!/ \\ O(1)}}{\deg_x} \cdot p = n^2 p$

- Hopefully, we can choose $p = \Theta(n^{1/2})$, so total time = $n^{2.5}$

Recursion: first attempt

Basic idea:

Instead of brute-force, compute the approx-matrix $\tilde{C} = \tilde{A} \star \tilde{B}$ by recursion

Recursion: first attempt

<u>**Basic idea:**</u>

Instead of brute-force, compute the approx-matrix $\tilde{C} = \tilde{A} \star \tilde{B}$ by recursion

- Recursion:

$$\lfloor \frac{A_{i,k}}{n^{1/2}} \rfloor, \lfloor \frac{2A_{i,k}}{n^{1/2}} \rfloor, \cdots \lfloor \frac{B_{k,j}}{n^{1/2}} \rfloor, \lfloor \frac{2B_{k,j}}{n^{1/2}} \rfloor, \cdots$$

$$A^{(l)}_{i,k}(x,y) = x^{\lfloor A_{i,k}/2^l \rfloor - 2\lfloor A_{i,k}/2^{l+1} \rfloor} \cdot y^{\lfloor A_{i,k}/2^{l+1} \rfloor} \mod p$$

$$B^{(l)}_{k,j}(x,y) = x^{\lfloor B_{k,j}/2^l \rfloor - 2\lfloor B_{k,j}/2^{l+1} \rfloor} \cdot y^{\lfloor B_{k,j}/2^{l+1} \rfloor} \mod p$$

Compute $C^{(l)}(x,y) = A^{(l)}(x,y) \cdot B^{(l)}(x,y)$

$$\lfloor \frac{C_{i,j}}{n^{1/2}} \rfloor, \lfloor \frac{2C_{i,j}}{n^{1/2}} \rfloor, \lfloor \frac{4C_{i,j}}{n^{1/2}} \rfloor, \lfloor \frac{8C_{i,j}}{n^{1/2}} \rfloor, \cdots\cdots$$

Recursion: first attempt

Basic idea:

Instead of brute-force, compute the approx-matrix $\tilde{C} = \tilde{A} \star \tilde{B}$ by recursion

- Recursion:

$$\lfloor \frac{A_{i,k}}{n^{1/2}} \rfloor, \lfloor \frac{2A_{i,k}}{n^{1/2}} \rfloor, \cdots \lfloor \frac{B_{k,j}}{n^{1/2}} \rfloor, \lfloor \frac{2B_{k,j}}{n^{1/2}} \rfloor, \cdots$$

$$A_{i,k}^{(l)}(x,y) = x^{\lfloor A_{i,k}/2^l \rfloor - 2\lfloor A_{i,k}/2^{l+1} \rfloor} \cdot y^{\lfloor A_{i,k}/2^{l+1} \rfloor} \mod p$$

$$B_{k,j}^{(l)}(x,y) = x^{\lfloor B_{k,j}/2^l \rfloor - 2\lfloor B_{k,j}/2^{l+1} \rfloor} \cdot y^{\lfloor B_{k,j}/2^{l+1} \rfloor} \mod p$$

Compute $C^{(l)}(x,y) = A^{(l)}(x,y) \cdot B^{(l)}(x,y)$

$$\lfloor \frac{C_{i,j}}{n^{1/2}} \rfloor, \lfloor \frac{2C_{i,j}}{n^{1/2}} \rfloor, \lfloor \frac{4C_{i,j}}{n^{1/2}} \rfloor, \lfloor \frac{8C_{i,j}}{n^{1/2}} \rfloor, \cdots\cdots$$

- Error terms:

$$A_{i,k}^{(l)}(x,y) = x^{\lfloor A_{i,k}/2^l \rfloor - 2\lfloor A_{i,k}/2^{l+1} \rfloor} \cdot y^{\lfloor A_{i,k}/2^{l+1} \rfloor} \mod p$$

$$B_{k,j}^{(l)}(x,y) = x^{\lfloor B_{k,j}/2^l \rfloor - 2\lfloor B_{k,j}/2^{l+1} \rfloor} \cdot y^{\lfloor B_{k,j}/2^{l+1} \rfloor} \mod p$$

Compute $C^{(l)}(x,y) = A^{(l)}(x,y) \cdot B^{(l)}(x,y)$

find triples $(i,j,k) \in [n]^3$ s.t.

$$\lfloor A_{i,k}/2^l \rfloor + \lfloor B_{k,j}/2^l \rfloor - \lfloor C_{i,j}/2^l \rfloor \neq O(1)$$

$$\lfloor A_{i,k}/2^l \rfloor + \lfloor B_{k,j}/2^l \rfloor - \lfloor C_{i,j}/2^l \rfloor \equiv O(1) \mod p$$

Inductively maintain all such triples

Recursion: first attempt

Basic idea:

Instead of brute-force, compute the approx-matrix $\tilde{C} = \tilde{A} \star \tilde{B}$ by recursion

- Error terms:

$$A^{(l)}_{i,k}(x,y) = x^{\lfloor A_{i,k}/2^l \rfloor - 2\lfloor A_{i,k}/2^{l+1} \rfloor} \cdot y^{\lfloor A_{i,k}/2^{l+1} \rfloor} \mod p$$

$$B^{(l)}_{k,j}(x,y) = x^{\lfloor B_{k,j}/2^l \rfloor - 2\lfloor B_{k,j}/2^{l+1} \rfloor} \cdot y^{\lfloor B_{k,j}/2^{l+1} \rfloor} \mod p$$

Compute $C^{(l)}(x,y) = A^{(l)}(x,y) \cdot B^{(l)}(x,y)$

find triples $(i,j,k) \in [n]^3$ s.t.

$$\lfloor A_{i,k}/2^l \rfloor + \lfloor B_{k,j}/2^l \rfloor - \lfloor C_{i,j}/2^l \rfloor \neq O(1)$$

$$\lfloor A_{i,k}/2^l \rfloor + \lfloor B_{k,j}/2^l \rfloor - \lfloor C_{i,j}/2^l \rfloor \equiv O(1) \mod p$$

Inductively maintain all such triples

- **Issues with induction:**

- Want to find $(i,j,k) \in [n]^3$

$$\lfloor A_{i,k}/2^l \rfloor + \lfloor B_{k,j}/2^l \rfloor - \lfloor C_{i,j}/2^l \rfloor \equiv 1 \mod p$$

Need to know triples:

$$\lfloor A_{i,k}/2^{l+1} \rfloor + \lfloor B_{k,j}/2^{l+1} \rfloor - \lfloor C_{i,j}/2^{l+1} \rfloor \equiv 2^{-1} \mod p$$

But only have triples:

$$\lfloor A_{i,k}/2^{l+1} \rfloor + \lfloor B_{k,j}/2^{l+1} \rfloor - \lfloor C_{i,j}/2^{l+1} \rfloor \equiv O(1) \mod p$$

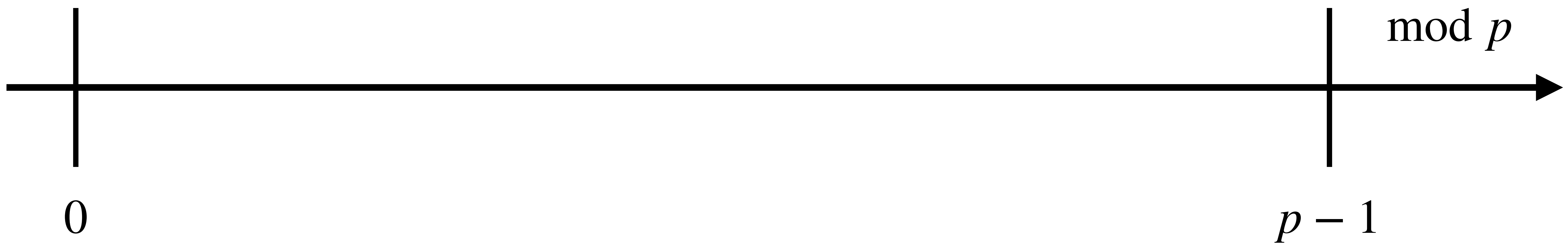
Recursion: first attempt

<u>Basic idea:</u>

Instead of brute-force, compute the approx-matrix $\tilde{C} = \tilde{A} \star \tilde{B}$ by recursion

<u>Maintaining erroneous terms during recursions:</u>

- **<u>Issue:</u>**
  division by 2 under modulo-p is not necessarily shrinking, could jump around
  $O(1)$ and 2^{-1} are far away under modulo-p



$$\Delta_l = \lfloor A_{i,k}/2^l \rfloor + \lfloor B_{k,j}/2^l \rfloor - \lfloor C_{i,j}/2^l \rfloor$$

Recursion: first attempt

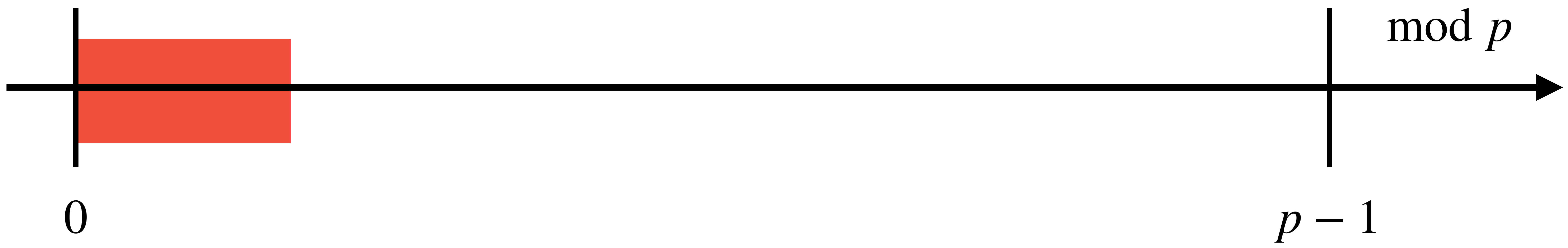
<u>Basic idea:</u>

Instead of brute-force, compute the approx-matrix $\tilde{C} = \tilde{A} \star \tilde{B}$ by recursion

<u>Maintaining erroneous terms during recursions:</u>

- **<u>Issue:</u>**
  division by 2 under modulo-p is not necessarily shrinking, could jump around
  $O(1)$ and 2^{-1} are far away under modulo-p

Hopefull all Δ_l
are **in this region**



$$\Delta_l = \lfloor A_{i,k}/2^l \rfloor + \lfloor B_{k,j}/2^l \rfloor - \lfloor C_{i,j}/2^l \rfloor$$

Recursion: first attempt

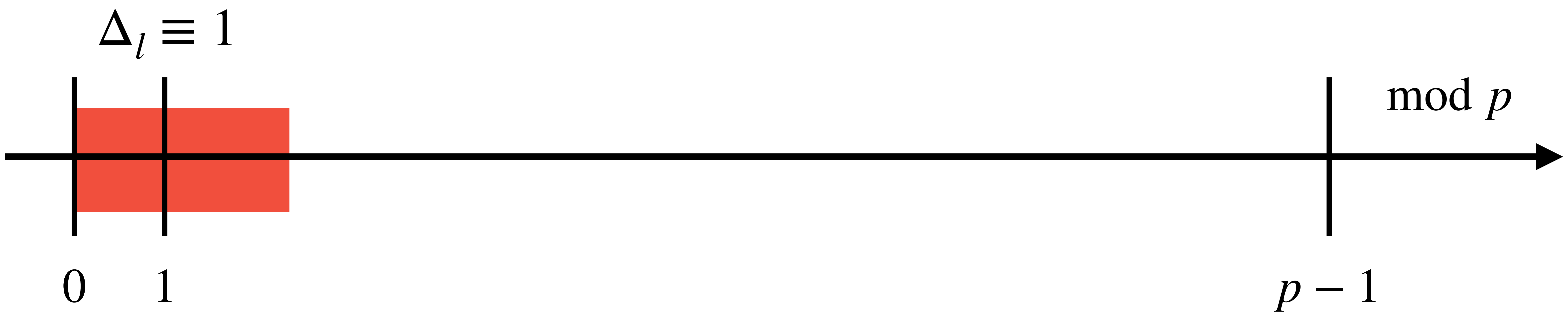
<u>Basic idea:</u>

Instead of brute-force, compute the approx-matrix $\tilde{C} = \tilde{A} \star \tilde{B}$ by recursion

<u>Maintaining erroneous terms during recursions:</u>

- **<u>Issue:</u>**
  division by 2 under modulo-p is not necessarily shrinking, could jump around
  $O(1)$ and 2^{-1} are far away under modulo-p

Hopefull all Δ_l are **in this region**



$$\Delta_l = \lfloor A_{i,k}/2^l \rfloor + \lfloor B_{k,j}/2^l \rfloor - \lfloor C_{i,j}/2^l \rfloor$$

Recursion: first attempt

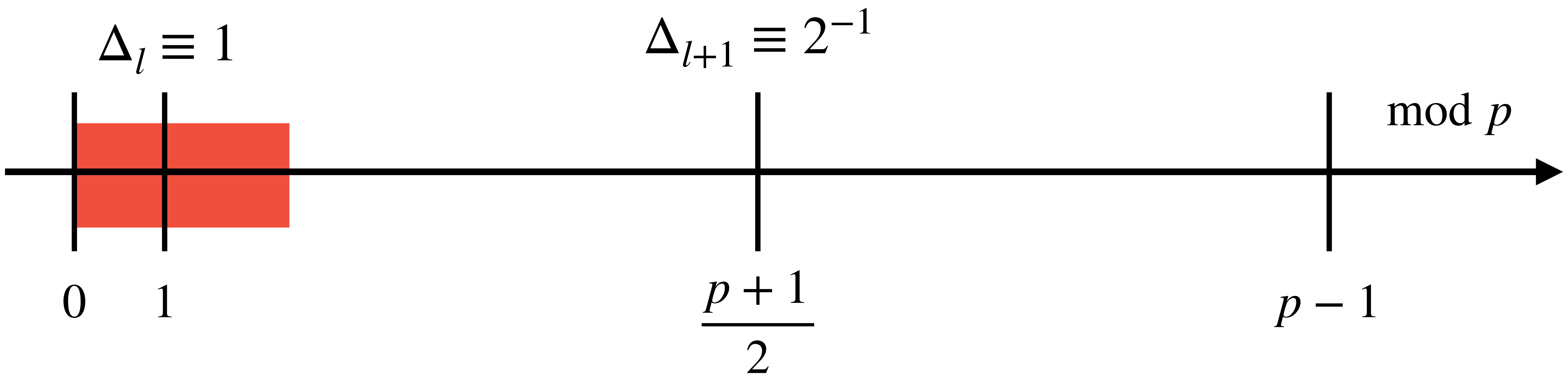
Basic idea:

Instead of brute-force, compute the approx-matrix $\tilde{C} = \tilde{A} \star \tilde{B}$ by recursion

Maintaining erroneous terms during recursions:

- **Issue:**
  division by 2 under modulo-p is not necessarily shrinking, could jump around
  $O(1)$ and 2^{-1} are far away under modulo-p

Hopefull all Δ_l are **in this region**



$$\Delta_l = \lfloor A_{i,k}/2^l \rfloor + \lfloor B_{k,j}/2^l \rfloor - \lfloor C_{i,j}/2^l \rfloor$$

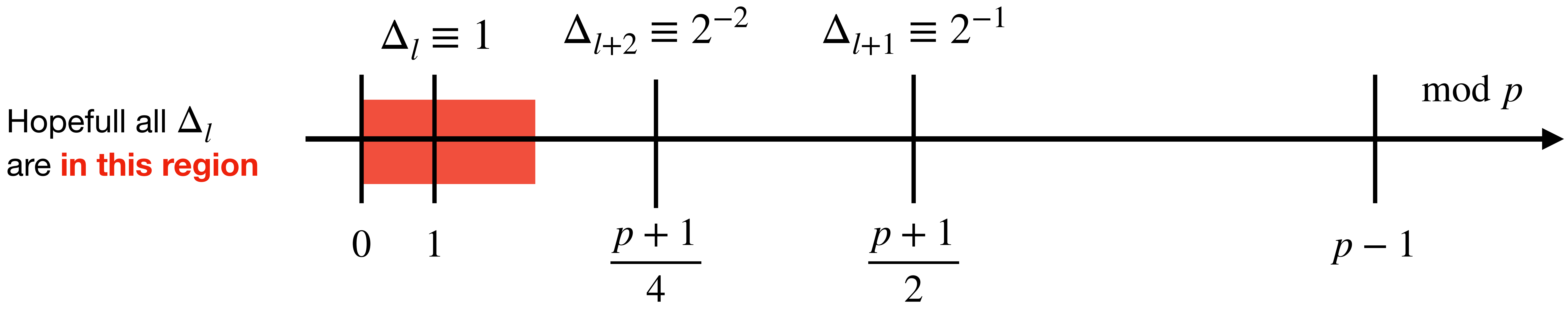
Recursion: first attempt

Basic idea:

Instead of brute-force, compute the approx-matrix $\tilde{C} = \tilde{A} \star \tilde{B}$ by recursion

Maintaining erroneous terms during recursions:

- **Issue:**
  division by 2 under modulo-p is not necessarily shrinking, could jump around
  $O(1)$ and 2^{-1} are far away under modulo-p



$$\Delta_l = \lfloor A_{i,k}/2^l \rfloor + \lfloor B_{k,j}/2^l \rfloor - \lfloor C_{i,j}/2^l \rfloor$$

Recursion: first attempt

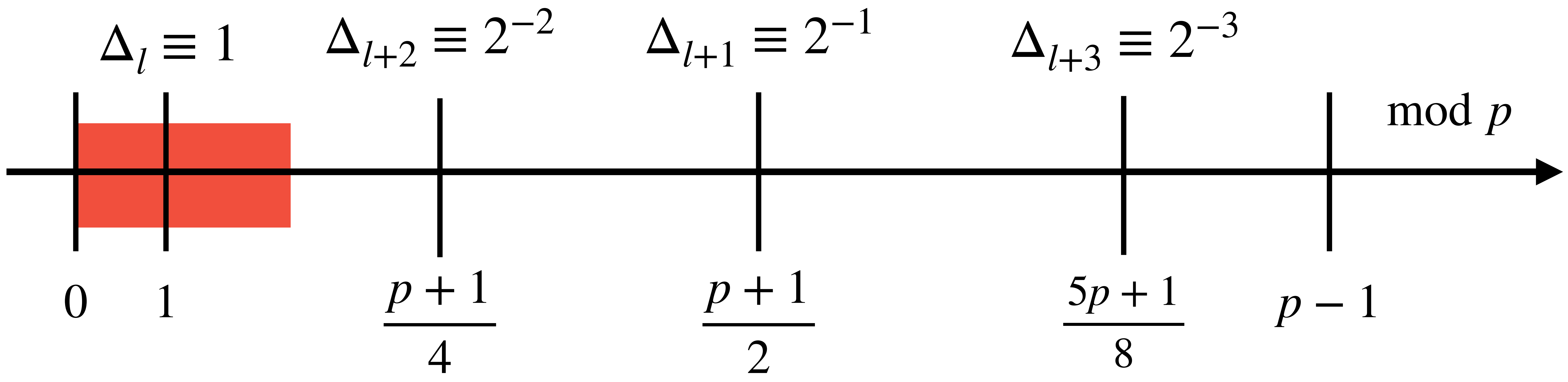
<u>Basic idea:</u>

Instead of brute-force, compute the approx-matrix $\tilde{C} = \tilde{A} \star \tilde{B}$ by recursion

<u>Maintaining erroneous terms during recursions:</u>

- **<u>Issue:</u>**
  division by 2 under modulo-p is not necessarily shrinking, could jump around
  $O(1)$ and 2^{-1} are far away under modulo-p

Hopefull all Δ_l are **in this region**



$$\Delta_l = \lfloor A_{i,k}/2^l \rfloor + \lfloor B_{k,j}/2^l \rfloor - \lfloor C_{i,j}/2^l \rfloor$$

Recursion: scaling residues

Key idea:

Instead of scaling entry values, only **scale the residues** modulo-p

Recursion: scaling residues

Key idea:

Instead of scaling entry values, only **scale the residues** modulo-p

- Previously: $\lfloor\frac{A_{i,k}}{n^{1/2}}\rfloor, \lfloor\frac{2A_{i,k}}{n^{1/2}}\rfloor, \lfloor\frac{4A_{i,k}}{n^{1/2}}\rfloor, \lfloor\frac{8A_{i,k}}{n^{1/2}}\rfloor, \ldots\ldots, \lfloor\frac{B_{k,j}}{n^{1/2}}\rfloor, \lfloor\frac{2B_{k,j}}{n^{1/2}}\rfloor, \lfloor\frac{4B_{k,j}}{n^{1/2}}\rfloor, \lfloor\frac{8B_{k,j}}{n^{1/2}}\rfloor, \ldots\ldots$

$$A^{(l)}_{i,k}(x,y) = x^{\{0,1\}} \cdot y^{\lfloor A_{i,k}/2^l \rfloor} \mod p \qquad B^{(l)}_{k,j}(x,y) = x^{\{0,1\}} \cdot y^{\lfloor B_{k,j}/2^l \rfloor} \mod p$$

Recursion: scaling residues

Key idea:

Instead of scaling entry values, only **scale the residues** modulo-p

- Previously: $\lfloor \frac{A_{i,k}}{n^{1/2}} \rfloor, \lfloor \frac{2A_{i,k}}{n^{1/2}} \rfloor, \lfloor \frac{4A_{i,k}}{n^{1/2}} \rfloor, \lfloor \frac{8A_{i,k}}{n^{1/2}} \rfloor, \ldots\ldots, \lfloor \frac{B_{k,j}}{n^{1/2}} \rfloor, \lfloor \frac{2B_{k,j}}{n^{1/2}} \rfloor, \lfloor \frac{4B_{k,j}}{n^{1/2}} \rfloor, \lfloor \frac{8B_{k,j}}{n^{1/2}} \rfloor, \ldots\ldots$

$$A^{(l)}_{i,k}(x,y) = x^{\{0,1\}} \cdot y^{\lfloor A_{i,k}/2^l \rfloor} \mod p \qquad B^{(l)}_{k,j}(x,y) = x^{\{0,1\}} \cdot y^{\lfloor B_{k,j}/2^l \rfloor} \mod p$$

- Now: $A_{i,k} = p\tilde{A}_{i,k} + R_{i,k} \qquad B_{k,j} = p\tilde{B}_{k,j} + S_{k,j}$

$$\lfloor \frac{R_{i,k}}{n^{1/2}} \rfloor, \lfloor \frac{2R_{i,k}}{n^{1/2}} \rfloor, \lfloor \frac{4R_{i,k}}{n^{1/2}} \rfloor, \lfloor \frac{8R_{i,k}}{n^{1/2}} \rfloor, \ldots\ldots, \lfloor \frac{S_{k,j}}{n^{1/2}} \rfloor, \lfloor \frac{2S_{k,j}}{n^{1/2}} \rfloor, \lfloor \frac{4S_{k,j}}{n^{1/2}} \rfloor, \lfloor \frac{8S_{k,j}}{n^{1/2}} \rfloor, \ldots\ldots$$

$$A^{(l)}_{i,k}(x,y) = x^{\lfloor R_{i,k}/2^l \rfloor - 2\lfloor R_{i,k}/2^{l+1} \rfloor} \cdot y^{\lfloor R_{i,k}/2^{l+1} \rfloor} \qquad B^{(l)}_{k,j}(x,y) = x^{\lfloor S_{k,j}/2^l \rfloor - 2\lfloor S_{k,j}/2^{l+1} \rfloor} \cdot y^{\lfloor S_{k,j}/2^{l+1} \rfloor}$$

Recursion: scaling residues

Key idea:
Instead of scaling entry values, only **scale the residues** modulo-p

- Now: $A_{i,k} = p\tilde{A}_{i,k} + R_{i,k}$ $\quad B_{k,j} = p\tilde{B}_{k,j} + S_{k,j}$

$$A_{i,k}^{(l)}(x,y) = x^{\lfloor R_{i,k}/2^l \rfloor - 2\lfloor R_{i,k}/2^{l+1} \rfloor} \cdot y^{\lfloor R_{i,k}/2^{l+1} \rfloor} \qquad B_{k,j}^{(l)}(x,y) = x^{\lfloor S_{k,j}/2^l \rfloor - 2\lfloor S_{k,j}/2^{l+1} \rfloor} \cdot y^{\lfloor S_{k,j}/2^{l+1} \rfloor}$$

- Recursion: $C_{i,j} = p\tilde{C}_{i,j} + T_{i,j}$

$$\lfloor T_{i,j}/2^l \rfloor = \min_{1 \le k \le n} \{ \lfloor R_{i,k}/2^l \rfloor + \lfloor S_{k,j}/2^l \rfloor \mid |\tilde{A}_{i,k} + \tilde{B}_{k,j} - \tilde{C}_{i,j}| = O(1) \} \pm O(1)$$

Compute $C^{(l)}(x,y) = A^{(l)}(x,y) \cdot B^{(l)}(x,y)$, and then some extra work

Recursion: scaling residues

Key idea:

Instead of scaling entry values, only **scale the residues** modulo-p

- Error terms:

$$A^{(l)}_{i,k}(x,y) = x^{\lfloor R_{i,k}/2^l \rfloor - 2\lfloor R_{i,k}/2^{l+1} \rfloor} \cdot y^{\lfloor R_{i,k}/2^{l+1} \rfloor}$$

$$B^{(l)}_{k,j}(x,y) = x^{\lfloor S_{k,j}/2^l \rfloor - 2\lfloor S_{k,j}/2^{l+1} \rfloor} \cdot y^{\lfloor S_{k,j}/2^{l+1} \rfloor}$$

Compute $C^{(l)}(x,y) = A^{(l)}(x,y) \cdot B^{(l)}(x,y)$

find triples $(i,j,k) \in [n]^3$ s.t.

$\lfloor T_{i,j}/2^l \rfloor = \lfloor R_{i,k}/2^l \rfloor + \lfloor S_{k,j}/2^l \rfloor \pm O(1)$

$|\tilde{A}_{i,k} + \tilde{B}_{k,j} - \tilde{C}_{i,j}| \neq O(1)$

Inductively maintain all such triples

Recursion: scaling residues

Key idea:

Instead of scaling entry values, only **scale the residues** modulo-p

- Error terms:

$$A^{(l)}_{i,k}(x,y) = x^{\lfloor R_{i,k}/2^l \rfloor - 2\lfloor R_{i,k}/2^{l+1} \rfloor} \cdot y^{\lfloor R_{i,k}/2^{l+1} \rfloor}$$

$$B^{(l)}_{k,j}(x,y) = x^{\lfloor S_{k,j}/2^l \rfloor - 2\lfloor S_{k,j}/2^{l+1} \rfloor} \cdot y^{\lfloor S_{k,j}/2^{l+1} \rfloor}$$

Compute $C^{(l)}(x,y) = A^{(l)}(x,y) \cdot B^{(l)}(x,y)$

find triples $(i,j,k) \in [n]^3$ s.t.

$\lfloor T_{i,j}/2^l \rfloor = \lfloor R_{i,k}/2^l \rfloor + \lfloor S_{k,j}/2^l \rfloor \pm O(1)$

$|\tilde{A}_{i,k} + \tilde{B}_{k,j} - \tilde{C}_{i,j}| \neq O(1)$

Inductively maintain all such triples

- **Previous issues fixed:**

- Want to find $(i,j,k) \in [n]^3$

$\lfloor T_{i,j}/2^l \rfloor = \lfloor R_{i,k}/2^l \rfloor + \lfloor S_{k,j}/2^l \rfloor \pm O(1)$

Need to know triples:

$\lfloor T_{i,j}/2^{l+1} \rfloor = \lfloor R_{i,k}/2^{l+1} \rfloor + \lfloor S_{k,j}/2^{l+1} \rfloor \pm \frac{O(1)}{2}$

Already have maintained:

$\lfloor T_{i,j}/2^{l+1} \rfloor = \lfloor R_{i,k}/2^{l+1} \rfloor + \lfloor S_{k,j}/2^{l+1} \rfloor \pm O(1)$

Recursion: finding error terms

<u>Basic idea:</u>

Use the fact that $p \in [n^{1/2}, 2n^{1/2}]$ is a uniformly random prime

<u>Goal:</u>

- At each recursion level, find all $(i, j, k) \in [n]^3$ such that:
- $\lfloor T_{i,j}/2^l \rfloor = \lfloor R_{i,k}/2^l \rfloor + \lfloor S_{k,j}/2^l \rfloor \pm O(1)$
  $|\tilde{A}_{i,k} + \tilde{B}_{k,j} - \tilde{C}_{i,j}| \neq O(1)$

Recursion: finding error terms

Basic idea:

Use the fact that $p \in [n^{1/2}, 2n^{1/2}]$ is a uniformly random prime

Goal:

- At each recursion level, find all $(i, j, k) \in [n]^3$ such that:
- $\lfloor T_{i,j}/2^l \rfloor = \lfloor R_{i,k}/2^l \rfloor + \lfloor S_{k,j}/2^l \rfloor \pm O(1)$
  $|\tilde{A}_{i,k} + \tilde{B}_{k,j} - \tilde{C}_{i,j}| \neq O(1)$

Sanity check:

- Even without recursion (l=0), the probability that $T_{i,j} = R_{i,k} + S_{k,j}$ is $\tilde{O}(n^{-1/2})$
- Need recursion to locate all k's

Recursion: finding error terms

Basic idea:

Use the fact that $p \in [n^{1/2}, 2n^{1/2}]$ is a uniformly random prime

Goal:

- At each recursion level, find all $(i, j, k) \in [n]^3$ such that:
- $\lfloor T_{i,j}/2^l \rfloor = \lfloor R_{i,k}/2^l \rfloor + \lfloor S_{k,j}/2^l \rfloor \pm O(1)$
  $|\tilde{A}_{i,k} + \tilde{B}_{k,j} - \tilde{C}_{i,j}| \neq O(1)$

Total number of error terms:

- $\lfloor T_{i,j}/2^l \rfloor = \lfloor R_{i,k}/2^l \rfloor + \lfloor S_{k,j}/2^l \rfloor \pm O(1)$
  or
  $|A_{i,k} + B_{k,j} - C_{i,j}| \equiv O(2^l) \mod p$
- Probability is at most $\tilde{O}(2^l \cdot n^{-1/2})$
  Total #errors = $\tilde{O}(2^l \cdot n^3/p)$

Recursion: finding error terms

Basic idea:

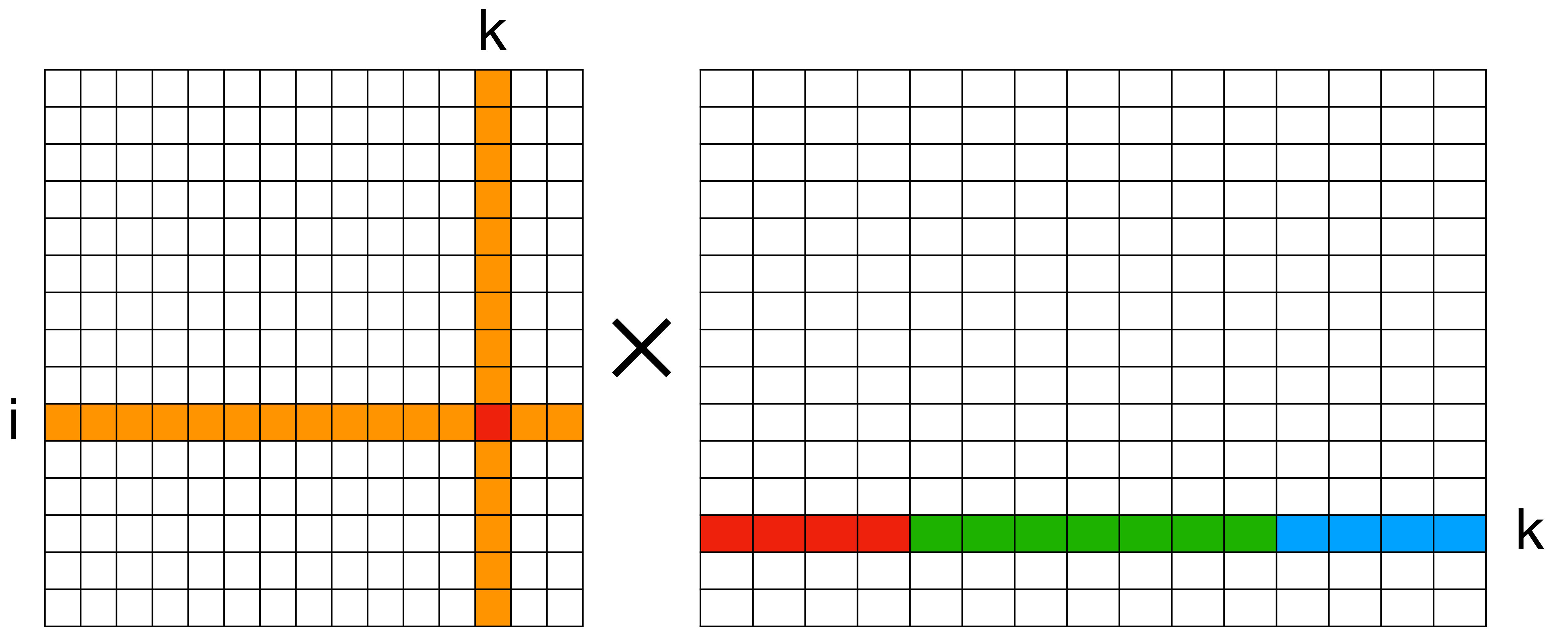
Use the fact that $p \in [n^{1/2}, 2n^{1/2}]$ is a uniformly random prime

Total number of error terms:

- $\lfloor T_{i,j}/2^l \rfloor = \lfloor R_{i,k}/2^l \rfloor + \lfloor S_{k,j}/2^l \rfloor \pm O(1)$
  or
  $|A_{i,k} + B_{k,j} - C_{i,j}| \equiv O(2^l) \mod p$

- Probability is at most $\tilde{O}(2^l \cdot n^{-1/2})$
  Total #errors = $\tilde{O}(2^l \cdot n^3/p)$

Extra factor of 2^l



Key observation:
Since B is monotone, we can partition each row $\{\lfloor S_{k,j}/2^l \rfloor\}_{1 \le j \le n}$ into $O(n/2^l)$ **intervals**, such that entries in the each interval are the same. Group error terms as **index-interval triples** $(i, k, [a, b])$

Recursion: finding error terms

Index-interval triples:

Use the fact that $p \in [n^{1/2}, 2n^{1/2}]$ is a uniformly random prime

Total #Index-interval triples:

- $|A_{i,k} + B_{k,j} - C_{i,j}| \equiv O(2^l) \mod p$
- Probability is at most $\tilde{O}(2^l \cdot n^{-1/2})$
  Total #error terms = $\tilde{O}(2^l \cdot n^3/p)$
- Partition each B-row into $O(n/2^l)$ intervals
- Total #index-interval triples = $\tilde{O}(n^3/p)$

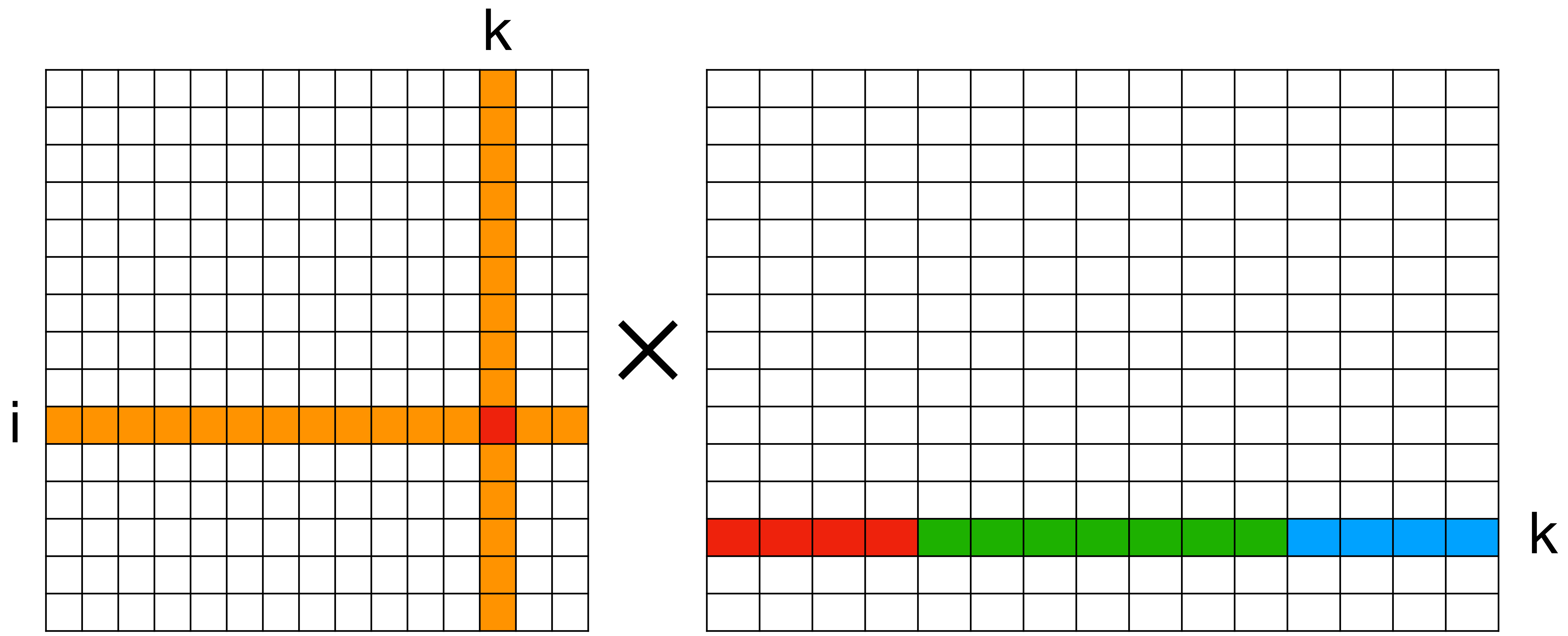
Recursion: finding error terms

Index-interval triples:

Use the fact that $p \in [n^{1/2}, 2n^{1/2}]$ is a uniformly random prime

Total #Index-interval triples:

- $|A_{i,k} + B_{k,j} - C_{i,j}| \equiv O(2^l) \mod p$
- Probability is at most $\tilde{O}(2^l \cdot n^{-1/2})$
  Total #error terms = $\tilde{O}(2^l \cdot n^3/p)$
- Partition each B-row into $O(n/2^l)$ intervals
- Total #index-interval triples = $\tilde{O}(n^3/p)$



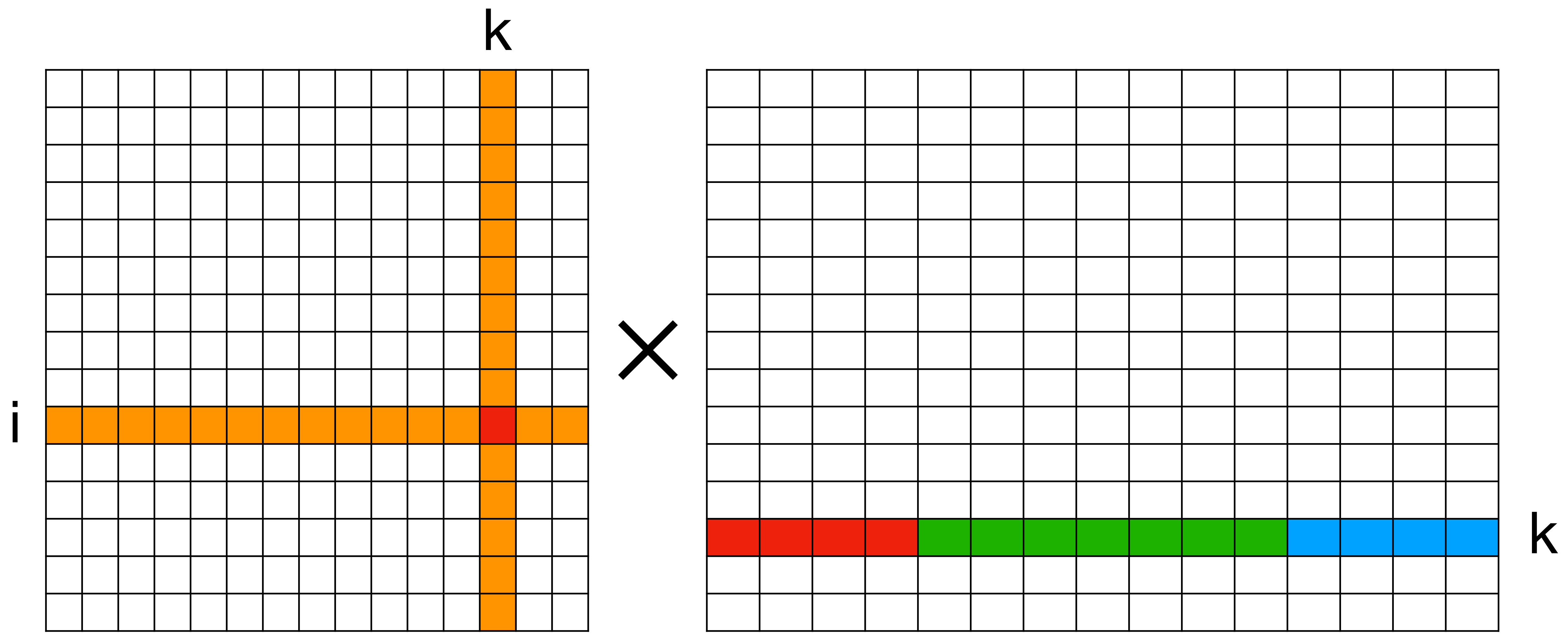
Recursion: finding error terms

Index-interval triples:

Use the fact that $p \in [n^{1/2}, 2n^{1/2}]$ is a uniformly random prime

Total #Index-interval triples:

- $|A_{i,k} + B_{k,j} - C_{i,j}| \equiv O(2^l) \mod p$
- Probability is at most $\tilde{O}(2^l \cdot n^{-1/2})$
  Total #error terms = $\tilde{O}(2^l \cdot n^3/p)$
- Partition each B-row into $O(n/2^l)$ intervals
- Total #index-interval triples = $\tilde{O}(n^3/p)$



Subtracting error terms:

For any $(i, k, [a, b])$, use segment-tree data structures

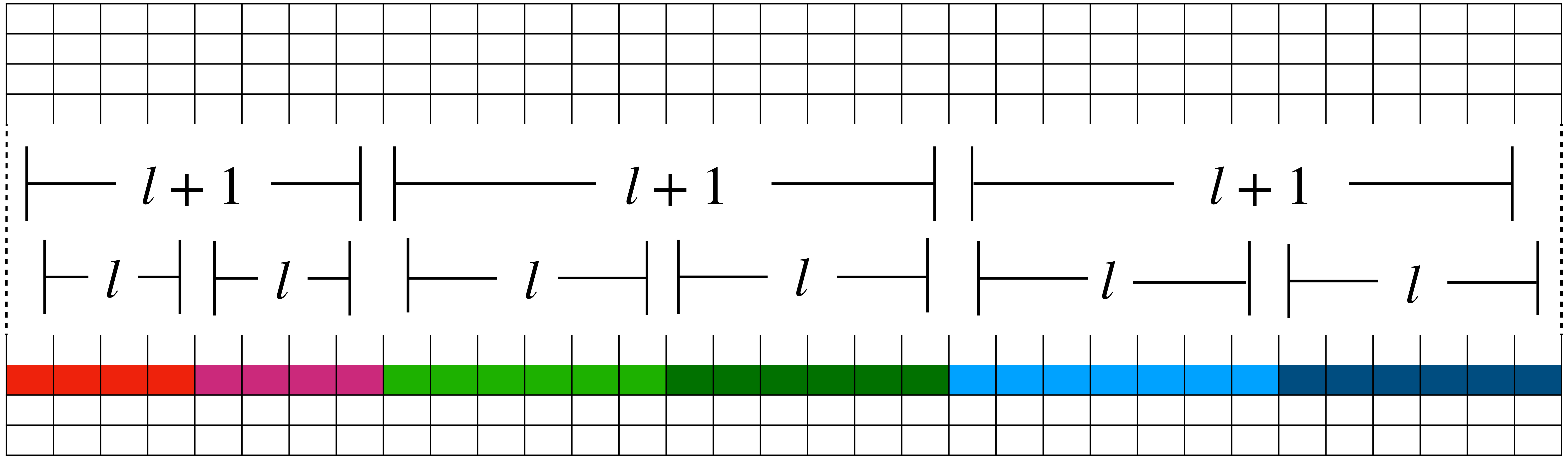
$$C^{(l)}_{i,j}(x, y) \leftarrow C^{(l)}_{i,j}(x, y) - A^{(l)}_{i,k}(x, y) \cdot B^{(l)}_{k,j}(x, y)$$

simultaneously for all $j \in [a, b]$, as all $B^{(l)}_{k,j}$ are equal

Recursion: finding error terms

Index-interval triples:

Use the fact that $p \in [n^{1/2}, 2n^{1/2}]$ is a uniformly random prime



Maintaining index-interval triples recursively:

- Triples $\{(i, k, [a, b])\}$ on the l-level is a **refinement** of triples on the $(l+1)$-level

Thanks for listening