Nearly Optimal Parallel Longest Increasing Subsequence

Nairen Cao, Shang-En Huang, Hsin-Hao Su

SPAA 2023

Longest increasing subsequence(LIS)

- Given a sequence of n numbers $A = (a_1, a_2, \dots, a_n)$, the goal is to find the longest subsequence from A such that its values are (strictly) increasing.

n = 6

4	**6**	**1**	**2**	**5**	**3**

LIS = 3

*	*	**1**	**2**	**5**	*

- LIS can be solved in $O(n \lg n)$ sequential time.

Previous Results

Reference	Total Work	Span	Notes
Nakashima and Fujiwara 2006	$O(n \lg n)$	$O\left(\frac{n \lg n}{p}\right)$ or $O(k^2 \lg n)$	Requires $p < n/k^2$.
Krusche and Tiskin 2009	$O(n \lg^2 n)$	$\tilde{O}(n^{\frac{2}{3}})$	
Shen, Wan, Gu, and Sun 2022	$O(n \lg^3 n)$	$O(k \lg^2 n)$	
Gu, Men, Shen, Sun, and Wan 2023	$O(n \lg k)$	$O(k \lg n)$	

p is the number of processors, k is the length of LIS.

Previous Results

Reference	Total Work	Span	Notes
Nakashima and Fujiwara 2006	$O(n \lg n)$	$O\left(\frac{n \lg n}{p}\right)$ or $O(k^2 \lg n)$	Requires $p < n/k^2$.
Krusche and Tiskin 2009	$O(n \lg^2 n)$	$\tilde{O}(n^{\frac{2}{3}})$	
Shen, Wan, Gu, and Sun 2022	$O(n \lg^3 n)$	$O(k \lg^2 n)$	
Gu, Men, Shen, Sun, and Wan 2023	$O(n \lg k)$	$O(k \lg n)$	

Can we achieve nearly linear work and nearly constant span?

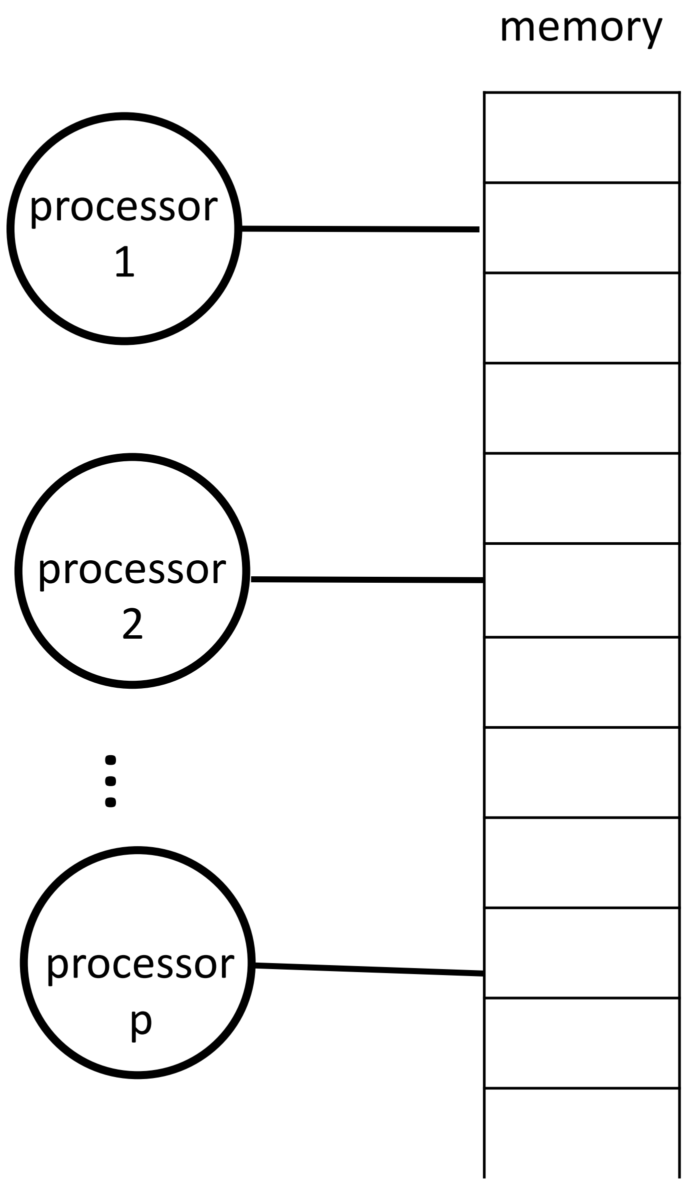
p is the number of processors, k is the length of LIS.

Our Result

Reference	Total Work	Span	Notes
Nakashima and Fujiwara 2006	$O(n \lg n)$	$O\left(\frac{n \lg n}{p}\right)$ or $O(k^2 \lg n)$	Requires $p < n/k^2$.
Krusche and Tiskin 2009	$O(n \lg^2 n)$	$\tilde{O}(n^{\frac{2}{3}})$	
Shen, Wan, Gu, and Sun 2022	$O(n \lg^3 n)$	$O(k \lg^2 n)$	
Gu, Men, Shen, Sun, and Wan 2023	$O(n \lg k)$	$O(k \lg n)$	
Our result	$O(n \lg^2 n \lg \lg n)$	$O(\lg^4 n)$	Deterministic algorithm
Our result	$O(n \lg^2 n)$	$O(\lg^4 n)$	Randomized,with AC^0 operations

p is the number of processors, k is the length of LIS.

EREW PRAM Model



- Simultaneous Read/Write to any memory location by different processors is forbidden

Work and span

- The **work** is the **total** number of operations that all processors perform (running time if there is one processor).
- The **span** is the **longest** series of operations that have to be performed sequentially (running time if there are infinite processors).

Outline

- Implicit subunit-Monge matrix multiplication (ISMMM)
- Connection between LIS and ISMMM
- How to solve the ISMMM problem

Implicit subunit-Monge matrix: sub-permutation matrix

$j \rightarrow$, $i \downarrow$

	0	1	2	3	4	5	6
0		0	0	0	0	0	0
1		0	0	0	0	1	0
2		0	0	0	1	0	0
3		0	0	0	0	0	0
4		0	0	0	0	0	1
5		0	0	0	0	0	0
6							

Sub-permutation matrix contains at most
one element equals to 1 each row and column

Implicit subunit-Monge matrix: sub-permutation matrix

$j \rightarrow$

$i \downarrow$

	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	0	0	0	0	1	0
2	0	0	0	0	1	0	0
3	0	0	0	0	0	0	0
4	0	0	0	0	0	0	1
5	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0

Sub-permutation matrix contains
at most 1 each row and column
the 0-th column and last row are all 0

Implicit subunit-Monge matrix: sub-unit Monge matrix

$i \downarrow$ / $j \rightarrow$	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	0	0	0	0	1	0
2	0	0	0	0	1	0	0
3	0	0	0	0	0	0	0
4	0	0	0	0	0	0	1
5	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0

$M^{\Sigma}(1,5) = 2$

$i \downarrow$ / $j \rightarrow$	0	1	2	3	4	5	6
0	0	0	0	0	1	2	3
1	0	0	0	0	1	2	3
2	0	0	0	0	1	1	2
3	0	0	0	0	0	0	1
4	0	0	0	0	0	0	1
5	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0

Sub-permutation matrix contains at most 1 each row and column
the 0-th row and columns are all 0

Distribution matrix $M^{\Sigma}(i,j) = \sum_{\{\hat{\imath} \geq i, \hat{\jmath} \leq j\}} P(i,j)$,
If P is a sub-permutation matrix,
then M^{Σ} is a subunit-Monge matrix

Implicit subunit-Monge matrix multiplication

$i \downarrow$ \ $j \rightarrow$	0	1	2	3
0	0	1	0	0
1	0	0	0	0
2	0	0	1	0
3	0	0	0	0

$\boxdot$

$i \downarrow$ \ $j \rightarrow$	0	1	2	3
0	0	0	1	0
1	0	0	0	0
2	0	1	0	0
3	0	0	0	0

We have two sub-permutation matrices

Implicit subunit-Monge matrix multiplication operator

Implicit subunit-Monge matrix multiplication

$i \downarrow$ / $j \rightarrow$	0	1	2	3
0	0	1	0	0
1	0	0	0	0
2	0	0	1	0
3	0	0	0	0

$\boxdot$

$i \downarrow$ / $j \rightarrow$	0	1	2	3
0	0	0	1	0
1	0	0	0	0
2	0	1	0	0
3	0	0	0	0

Compute the distribution matrix

$i \downarrow$ / $j \rightarrow$	0	1	2	3
0	0	1	2	2
1	0	0	1	1
2	0	0	1	1
3	0	0	0	0

$i \downarrow$ / $j \rightarrow$	0	1	2	3
0	0	1	2	2
1	0	1	1	1
2	0	1	1	1
3	0	0	0	0

Implicit subunit-Monge matrix multiplication

$j \rightarrow$, $i \downarrow$

i \ j	0	1	2	3
0	0	1	0	0
1	0	0	0	0
2	0	0	1	0
3	0	0	0	0

$\boxdot$

$j \rightarrow$, $i \downarrow$

i \ j	0	1	2	3
0	0	0	1	0
1	0	0	0	0
2	0	1	0	0
3	0	0	0	0

Compute the distribution matrix

$j \rightarrow$, $i \downarrow$

i \ j	0	1	2	3
0	0	1	2	2
1	0	0	1	1
2	0	0	1	1
3	0	0	0	0

(min, +) product

$j \rightarrow$, $i \downarrow$

i \ j	0	1	2	3
0	0	1	2	2
1	0	1	1	1
2	0	1	1	1
3	0	0	0	0

$=$

Subunit-Monge matrix

$j \rightarrow$, $i \downarrow$

i \ j	0	1	2	3
0	0	1	2	2
1	0	1	1	1
2	0	1	1	1
3	0	0	0	0

Implicit subunit-Monge matrix multiplication

$i \downarrow$ \ $j \rightarrow$	0	1	2	3
0	0	1	0	0
1	0	0	0	0
2	0	0	1	0
3	0	0	0	0

$\boxdot$

$i \downarrow$ \ $j \rightarrow$	0	1	2	3
0	0	0	1	0
1	0	0	0	0
2	0	1	0	0
3	0	0	0	0

$=$

$i \downarrow$ \ $j \rightarrow$	0	1	2	3
0	0	0	1	0
1	0	0	0	0
2	0	1	0	0
3	0	0	0	0

Compute the distribution matrix

Recover the sub-permutation matrix

$i \downarrow$ \ $j \rightarrow$	0	1	2	3
0	0	1	2	2
1	0	0	1	1
2	0	0	1	1
3	0	0	0	0

(min, +) product

$i \downarrow$ \ $j \rightarrow$	0	1	2	3
0	0	1	2	2
1	0	1	1	1
2	0	1	1	1
3	0	0	0	0

$=$

$i \downarrow$ \ $j \rightarrow$	0	1	2	3
0	0	1	2	2
1	0	1	1	1
2	0	1	1	1
3	0	0	0	0

Implicit subunit-Monge matrix multiplication

$i \downarrow$ / $j \rightarrow$	0	1	2	3
0	0	1	0	0
1	0	0	0	0
2	0	0	1	0
3	0	0	0	0

$\boxdot$

$i \downarrow$ / $j \rightarrow$	0	1	2	3
0	0	0	1	0
1	0	0	0	0
2	0	1	0	0
3	0	0	0	0

$=$

$i \downarrow$ / $j \rightarrow$	0	1	2	3
0	0	0	1	0
1	0	0	0	0
2	0	1	0	0
3	0	0	0	0

The input and output contains at most $O(n)$ non-zero terms,
Can we compute the output fast in the parallel setting?

Connection between LIS and ISMMM

Theorem: If one can solve the ISMMM problem in $O\big(W(n)\big)$ work and $O\big(S(n)\big)$ span. Then, one can compute an LIS in $O(W(n)\lg n)$ work and $O(S(n)\lg n)$ span.

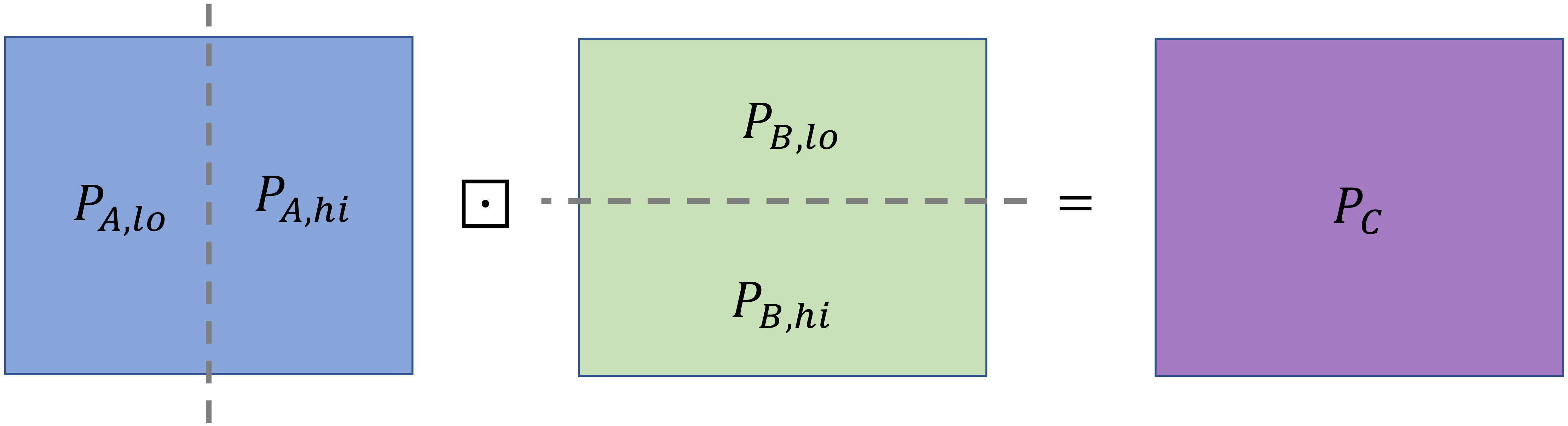
Connection between LIS and ISMMM

Theorem: If one can solve the ISMMM problem in $O\big(W(n)\big)$ work and $O\big(S(n)\big)$ span. Then, one can compute an LIS in $O(W(n)\lg n)$ work and $O(S(n)\lg n)$ span.

There is a parallel algorithm solving the ISMMM problem in $O(nlgn)$ work and $O(\lg^3 n)$ span.

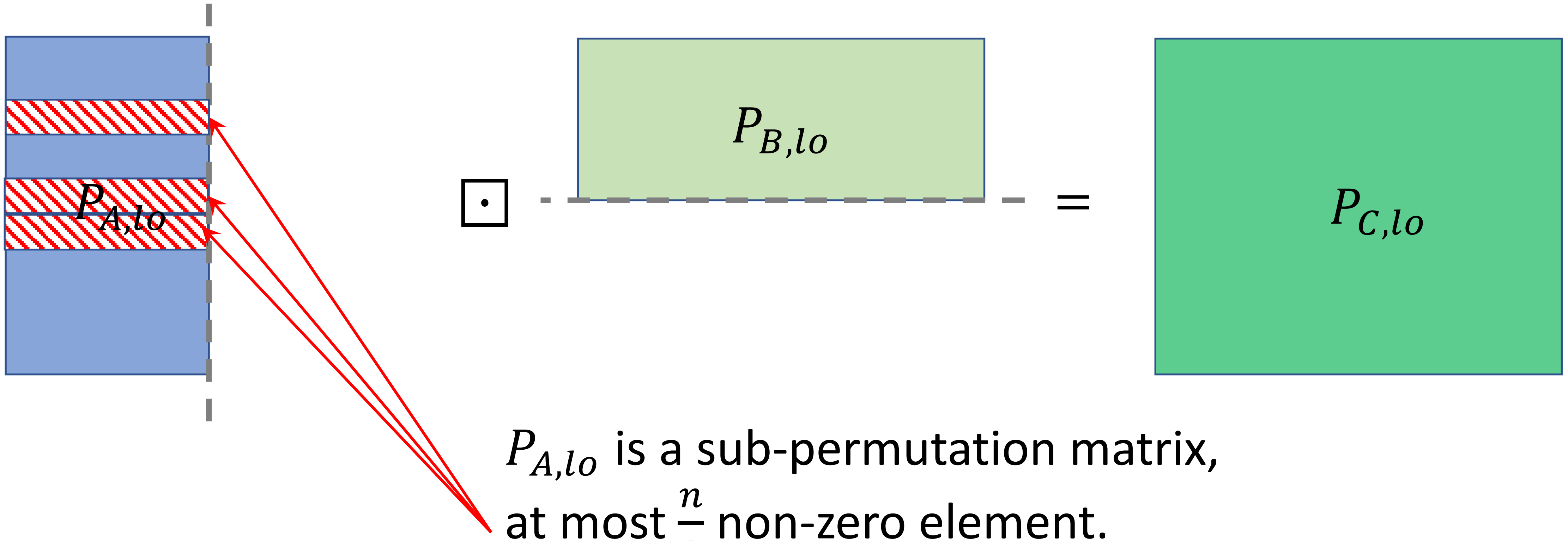
There is a parallel algorithm that computes an LIS in $O(nlg^2 n)$ work and $O(\lg^4 n)$ span.

ISMMM: Framework of Krusche and Tiskin's Algorithm 2010



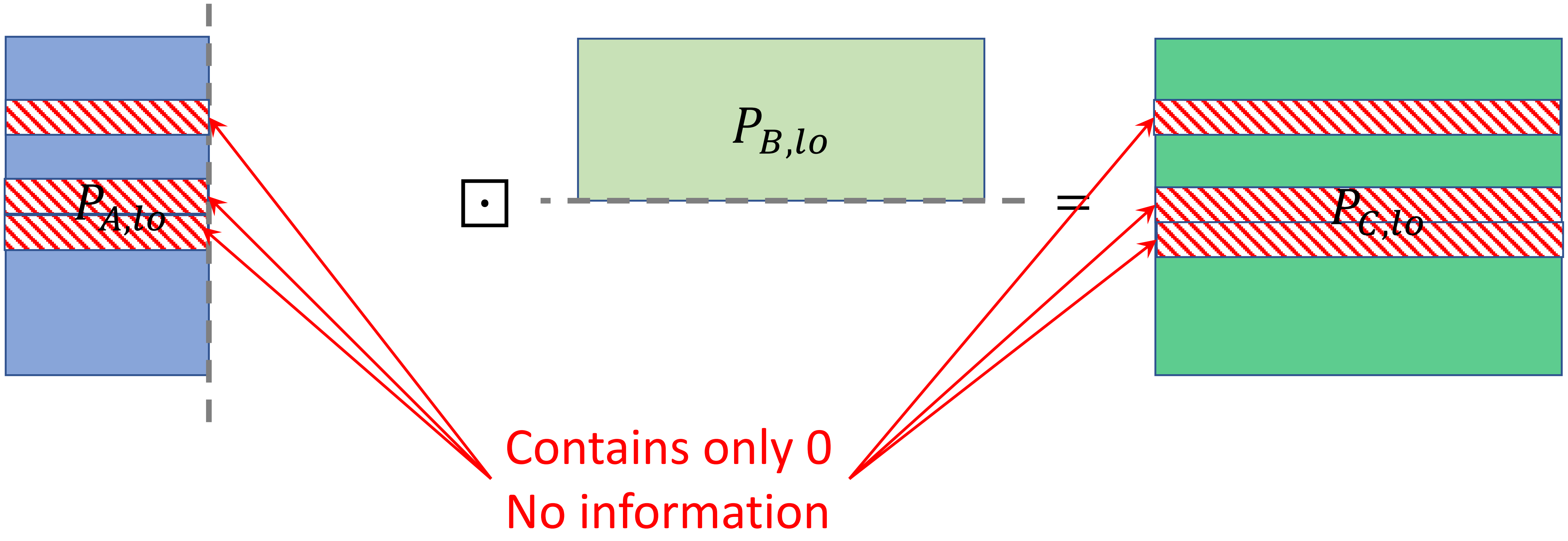
Divide-and-conquer method

Implicit subunit-Monge matrix multiplication: Divide

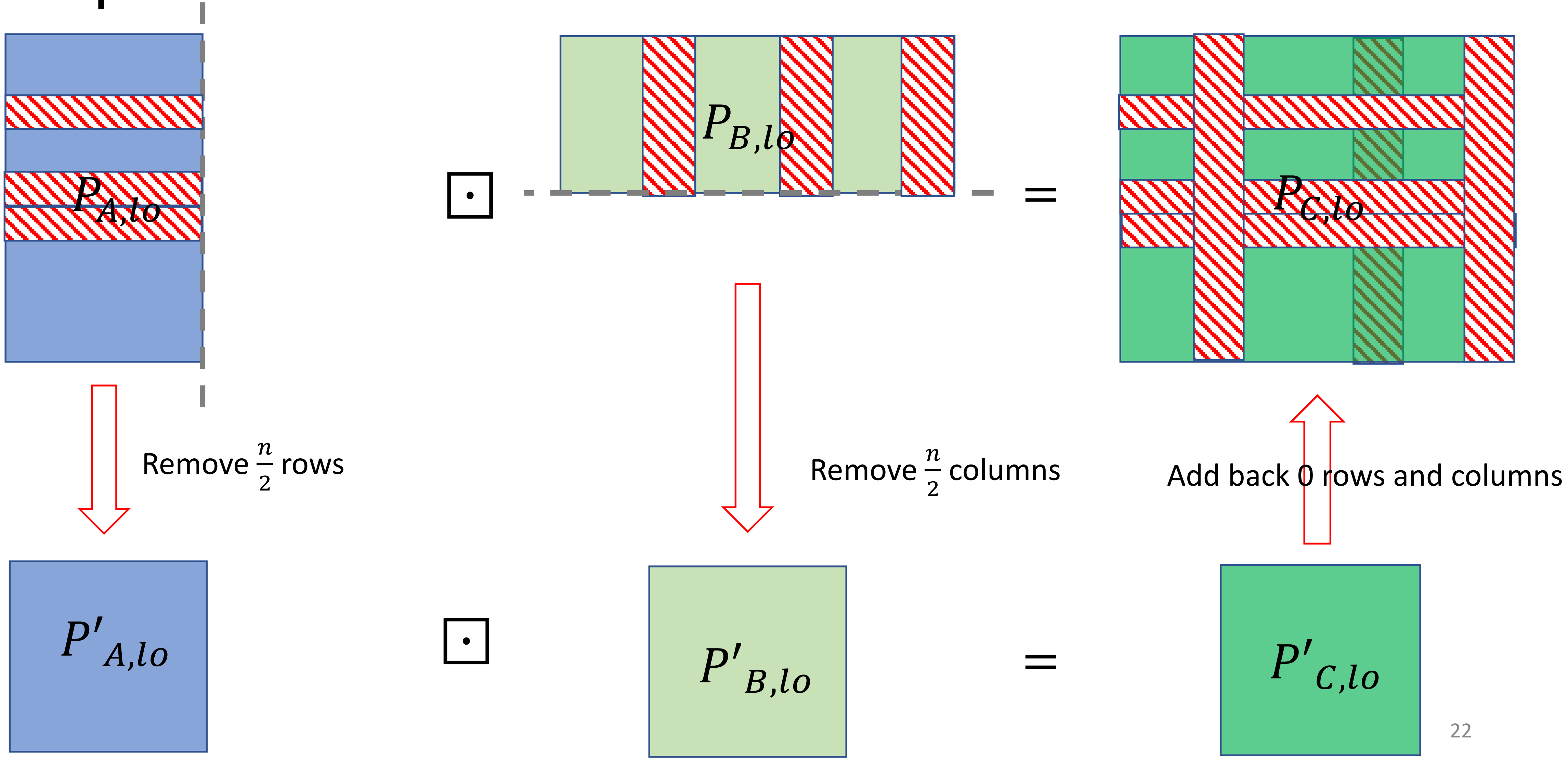


$P_{A,lo}$ is a sub-permutation matrix,
at most $\frac{n}{2}$ non-zero element.
At least $\frac{n}{2}$ rows contain only 0

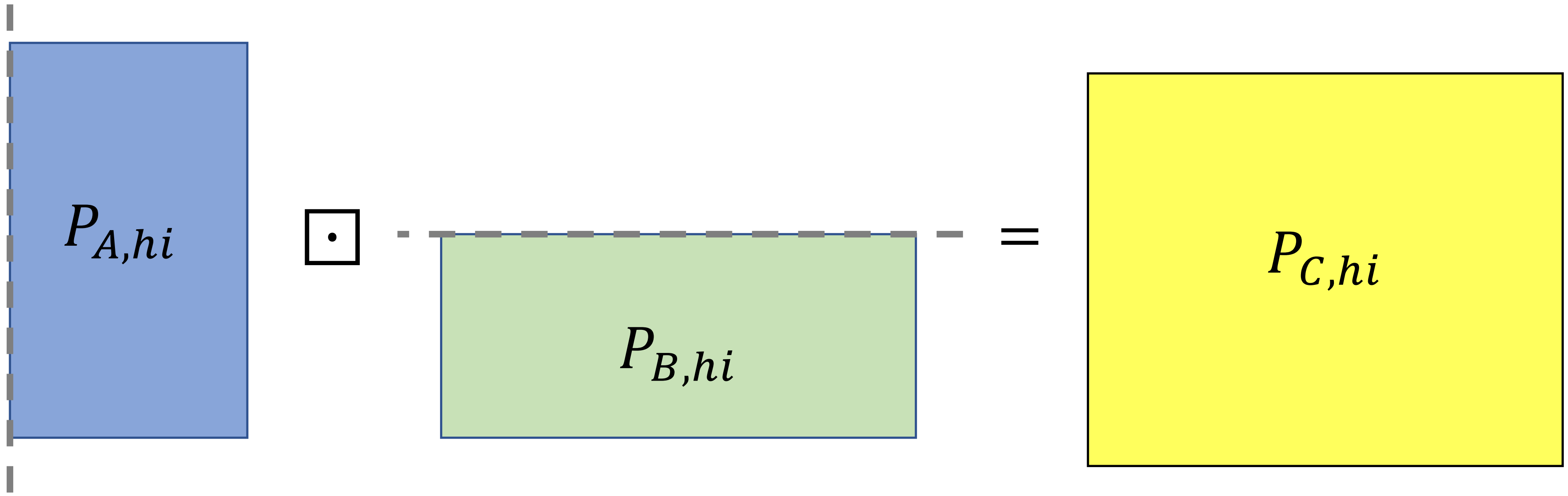
Implicit subunit-Monge matrix multiplication: Divide



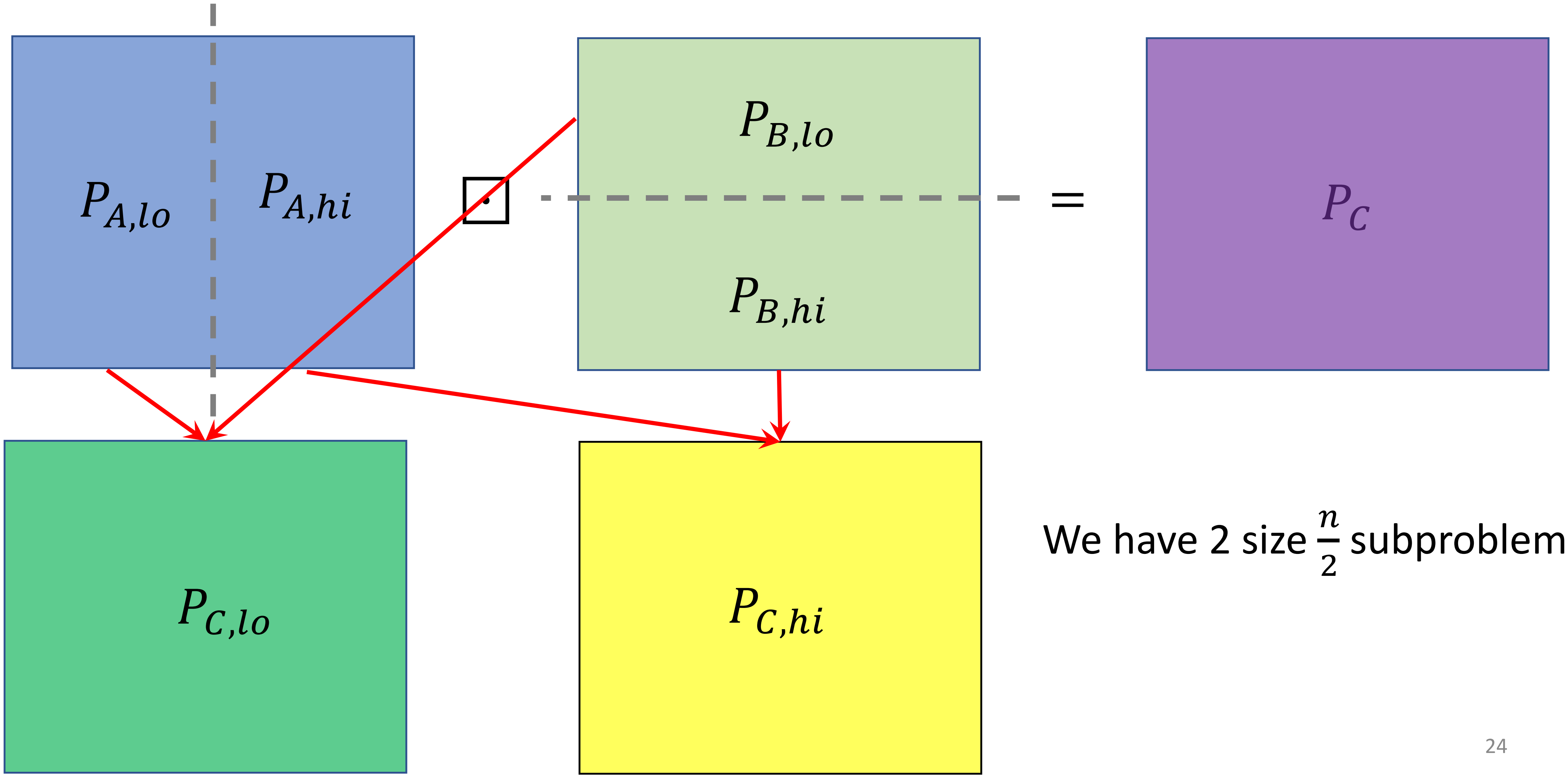
Implicit subunit-Monge matrix multiplication: Reduce subproblem size



Implicit subunit-Monge matrix multiplication

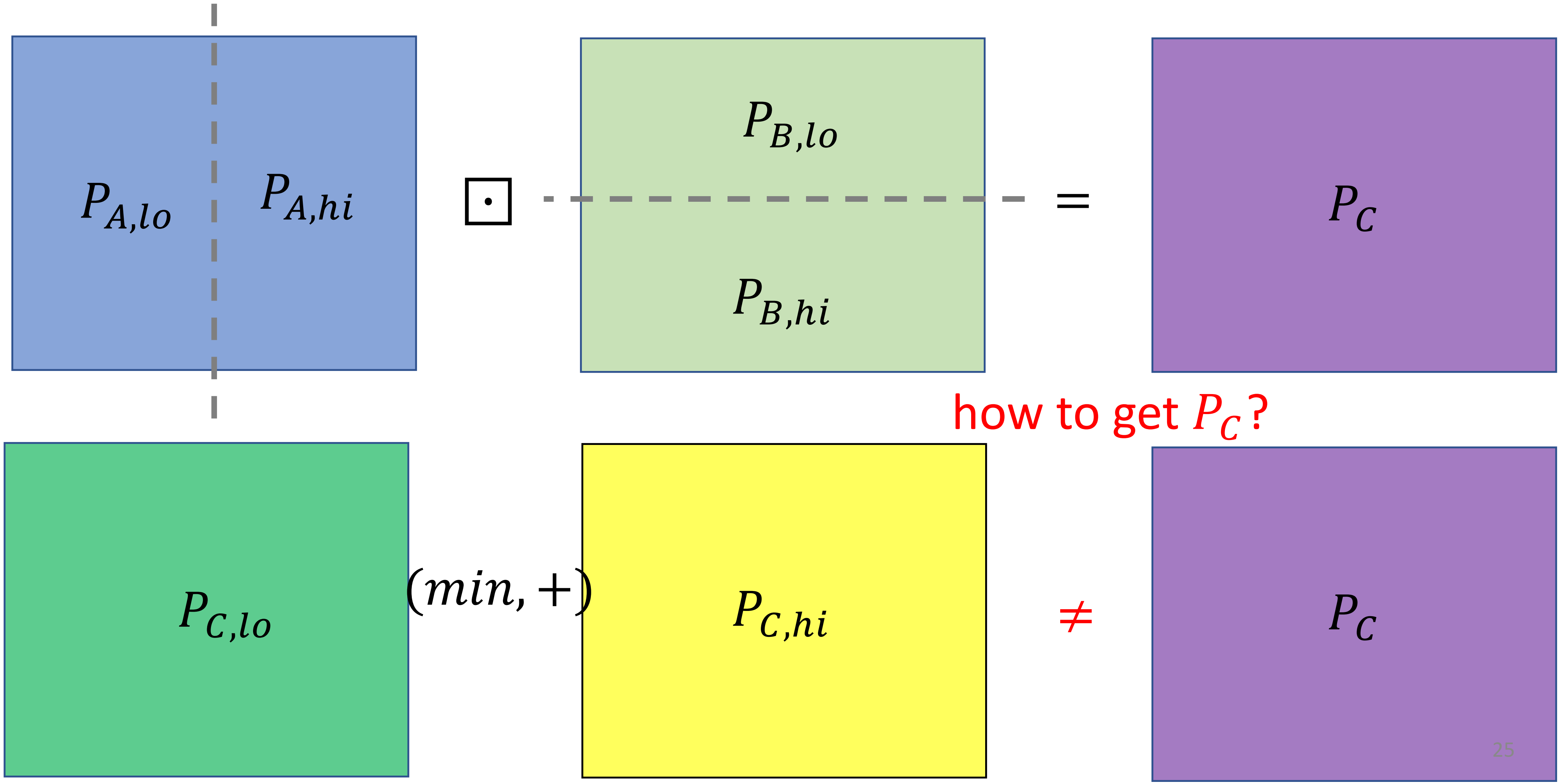


Implicit subunit-Monge matrix multiplication: Divide

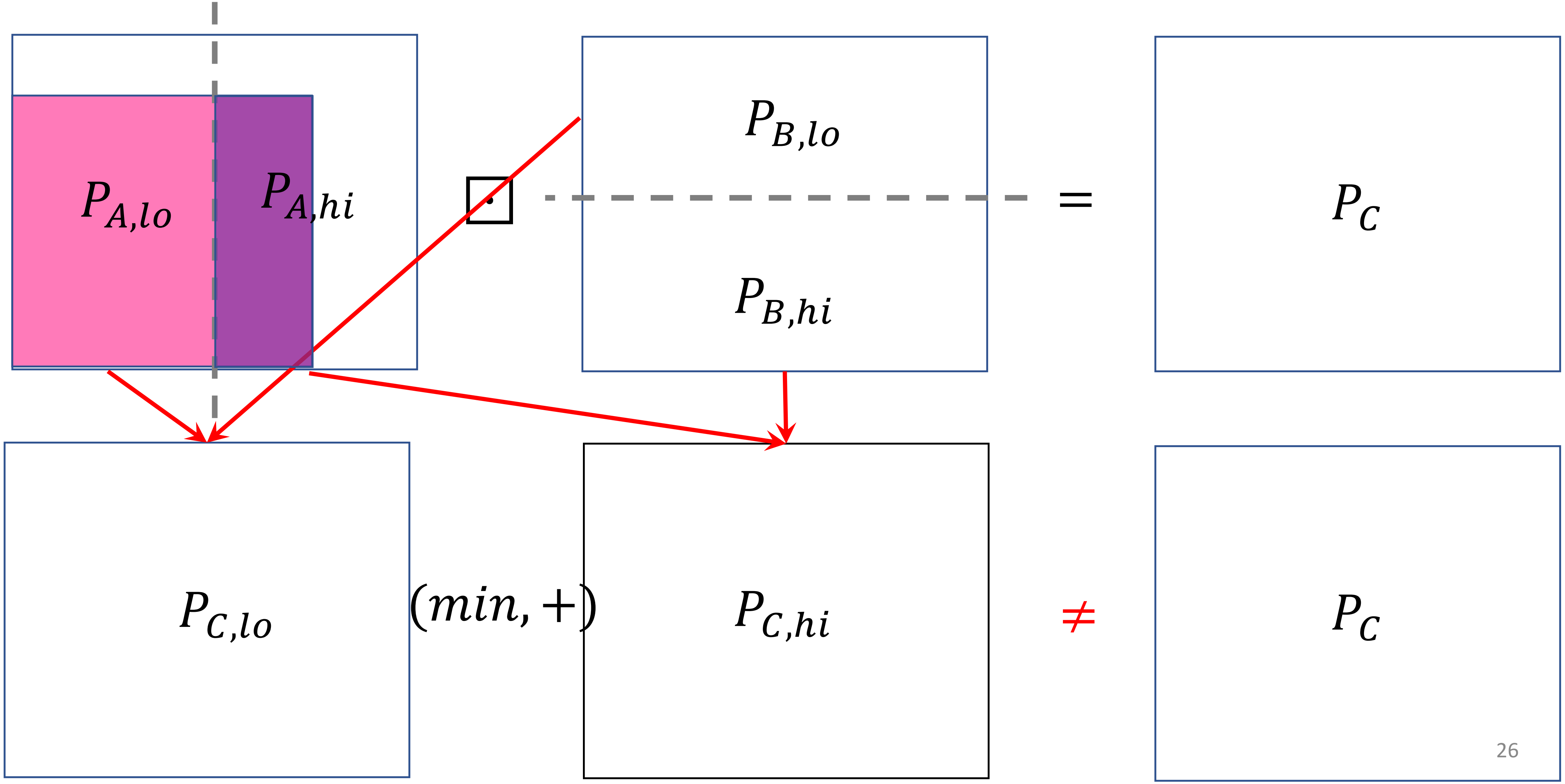


We have 2 size $\frac{n}{2}$ subproblems

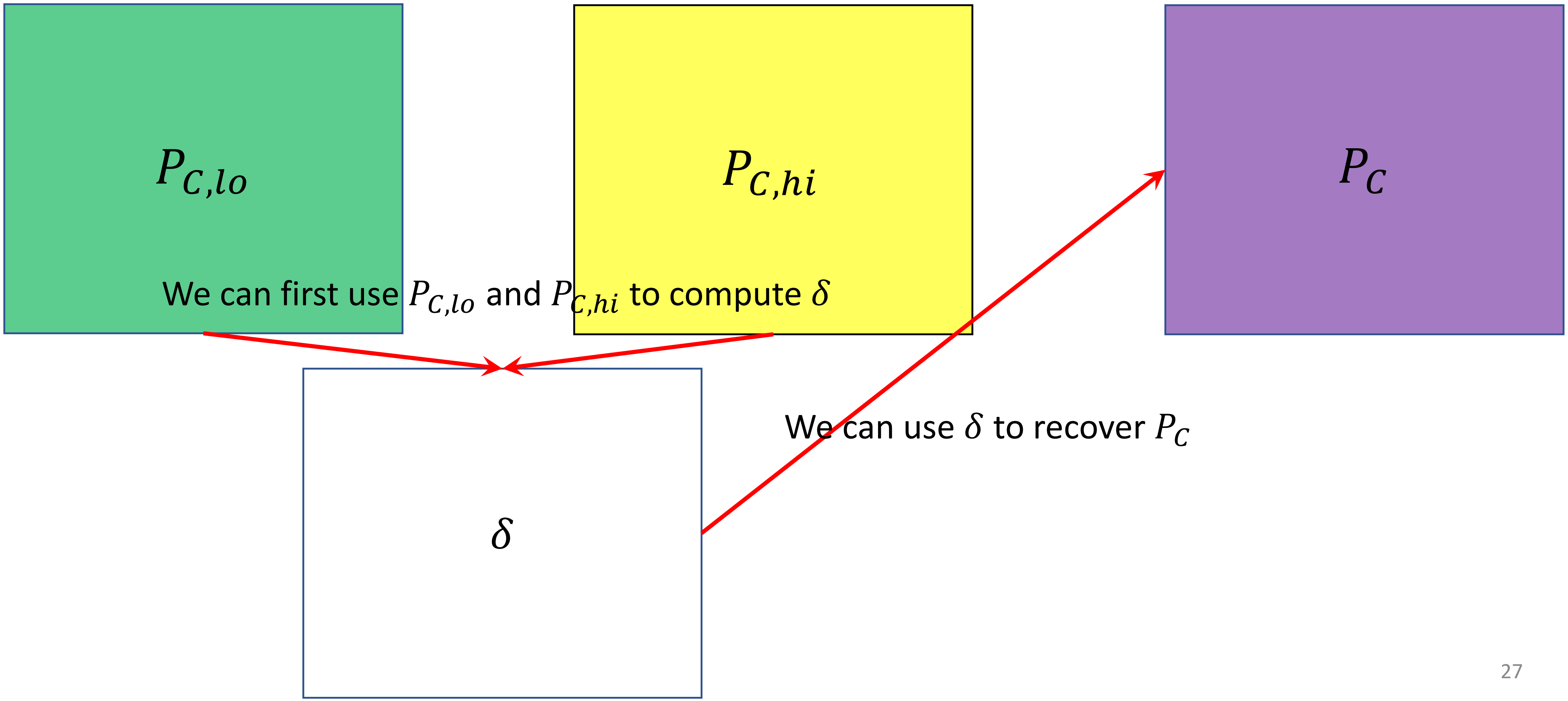
Implicit subunit-Monge matrix multiplication: combine



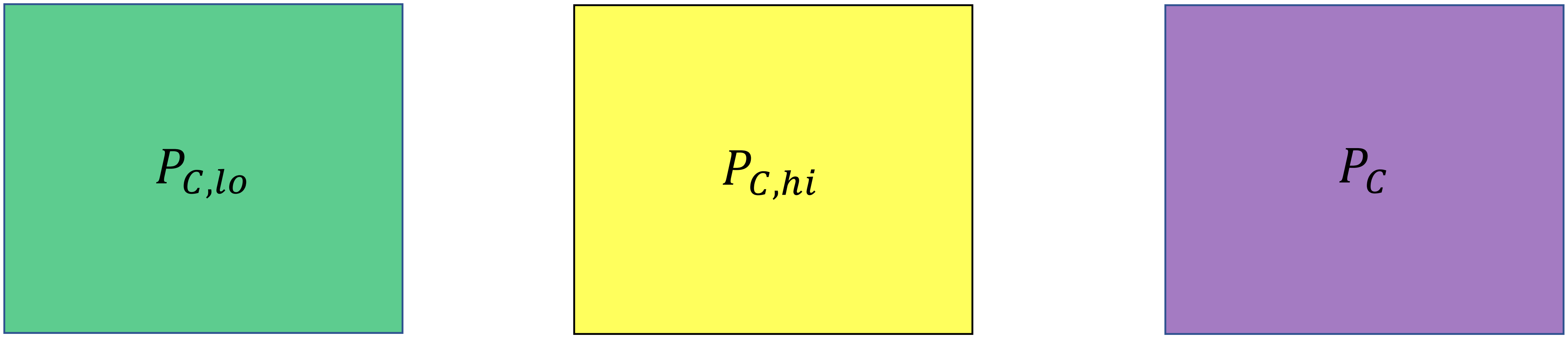
Implicit subunit-Monge matrix multiplication: difficulty of combine

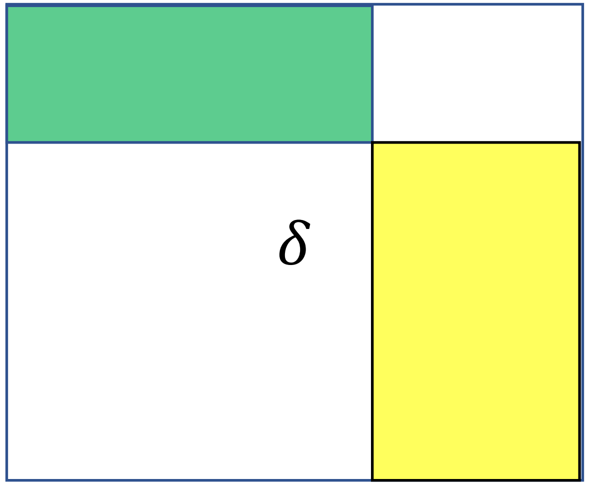


Implicit subunit-Monge matrix multiplication: δ matrix



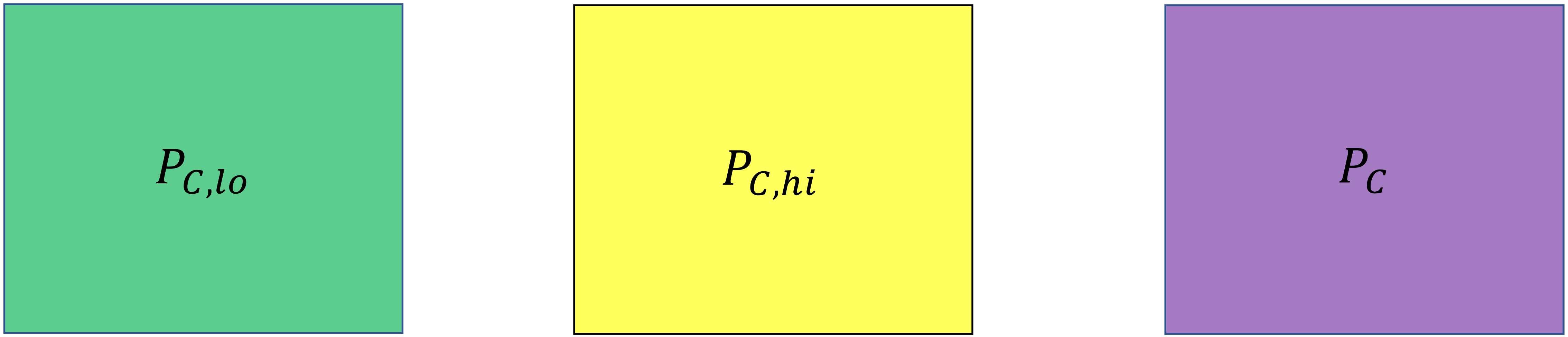
Implicit subunit-Monge matrix multiplication: δ matrix

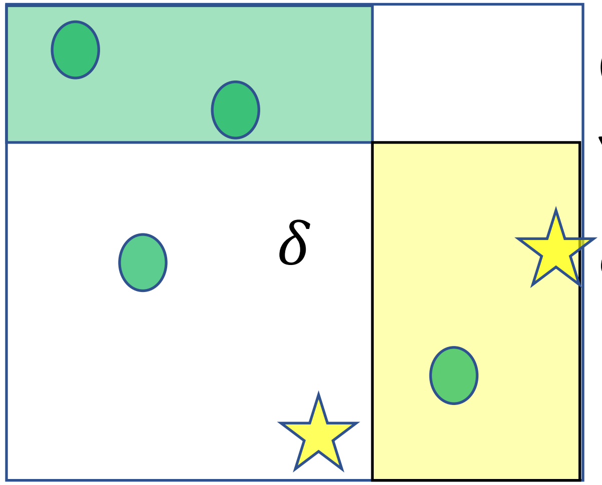




$\delta(i,j)$ = the left upper area sum of $P_{c,lo}$ − the right bottom area sum of $P_{c,hi}$

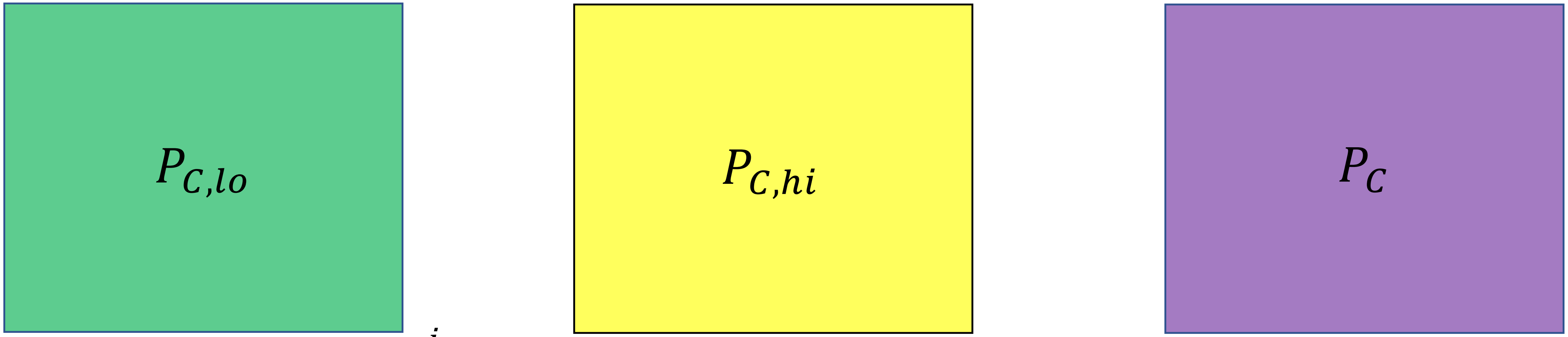
Implicit subunit-Monge matrix multiplication: δ matrix

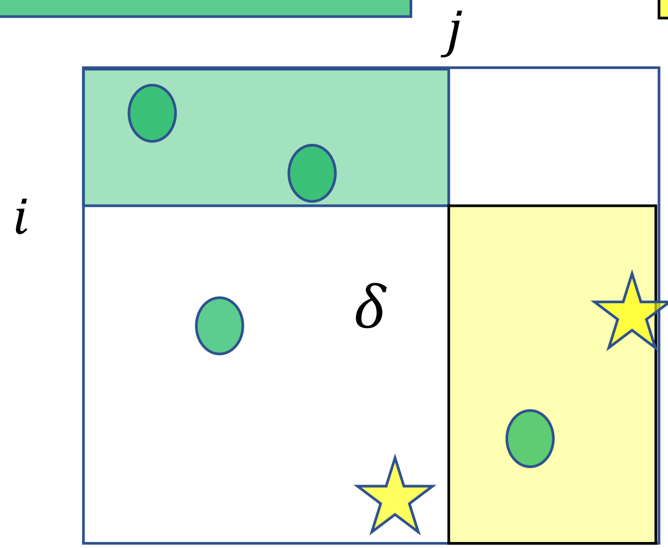




Green circle represents non-zero term of $P_{c,lo}$,
Yellow star represents non-zero term of $P_{c,hi}$
$\delta(i,j)$ = 2 – 1 = 1

Implicit subunit-Monge matrix multiplication: δ matrix





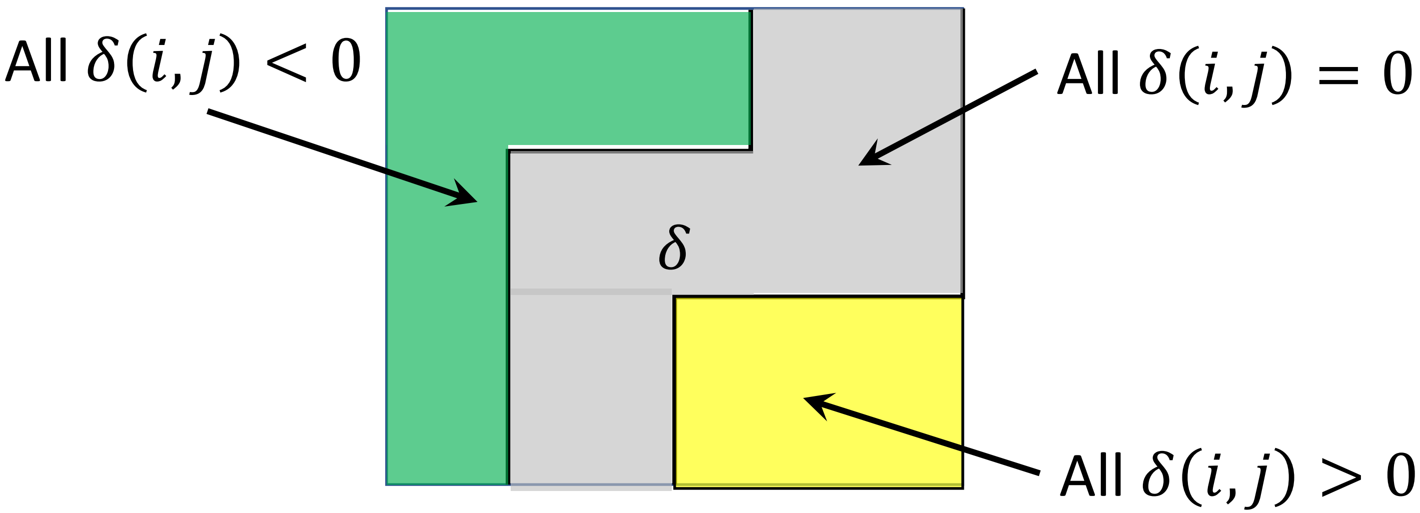
Green circle represents non-zero term of $P_{c,lo}$,

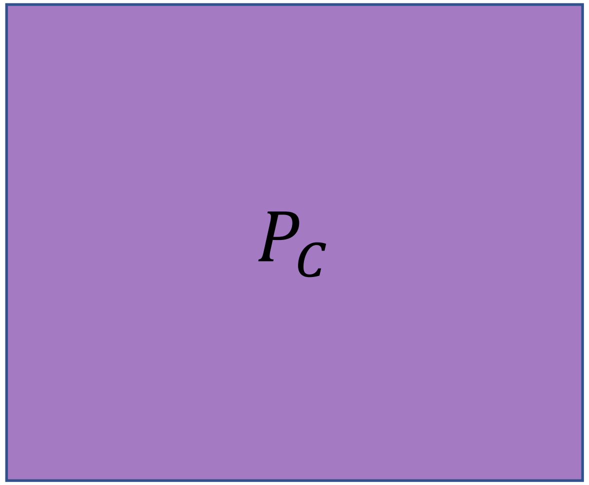
Yellow star represents non-zero term of $P_{c,hi}$

$\delta(i,j)$ = 2 – 1 = 1

If we increase i or j, $\delta(i,j)$ never decreases.

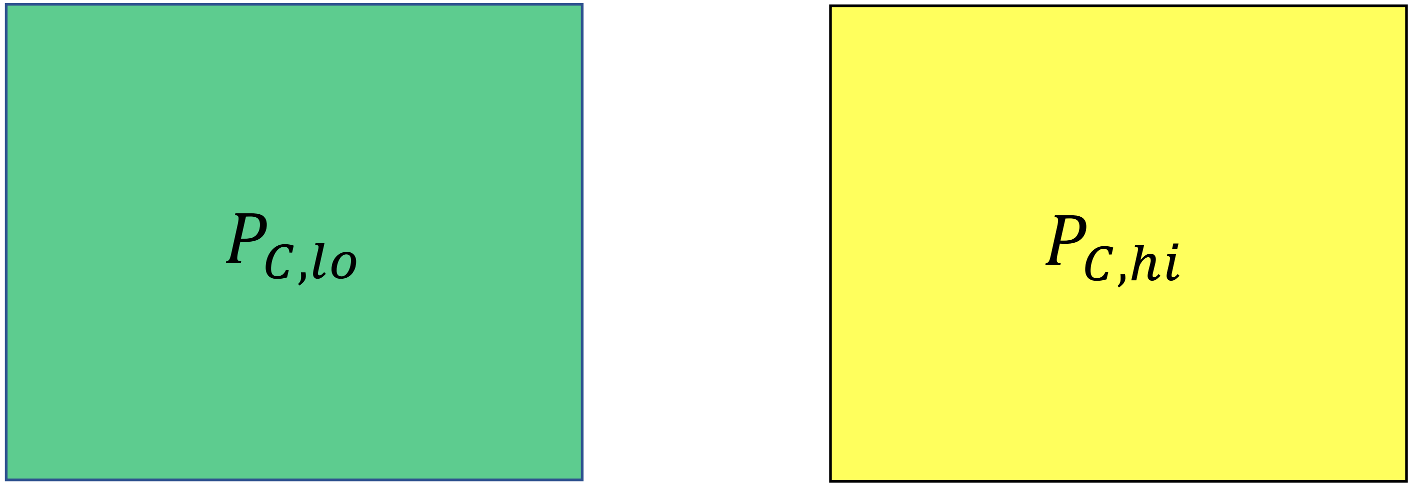
Implicit subunit-Monge matrix multiplication: splitting line

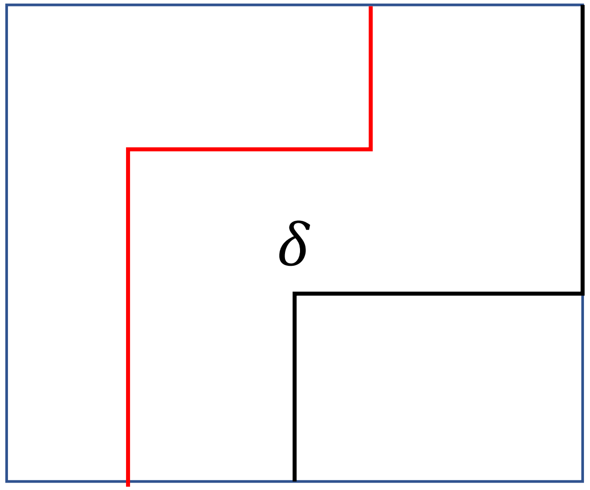




Lemma: Given $P_{c,lo}$ and $P_{c,hi}$, if one can decide three area of δ in $O\big(W(n)\big)$ work and $O\big(S(n)\big)$ span, one can compute P_c in $O\big(W(n)\big)$ work and $O\big(S(n)\big)$ span.

Implicit subunit-Monge matrix multiplication

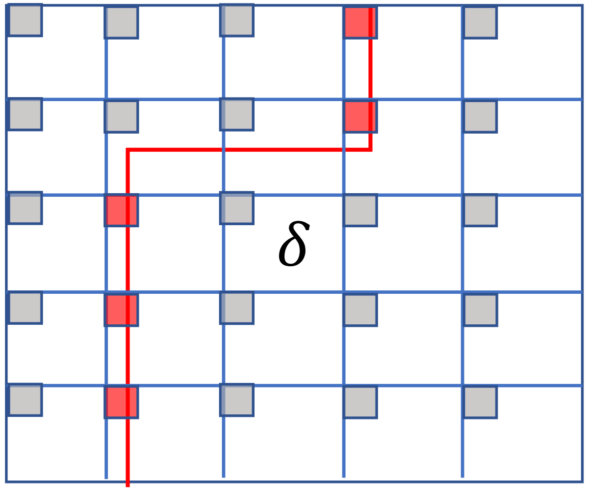




Given $P_{c,lo}$ and $P_{c,hi}$, how can we compute the line splitting the negative and non-negative area of δ in the parallel model?

There is an algorithm computing the splitting line in $O(n)$ work and $O(\lg^2 n)$ span.

Implicit subunit-Monge matrix multiplication



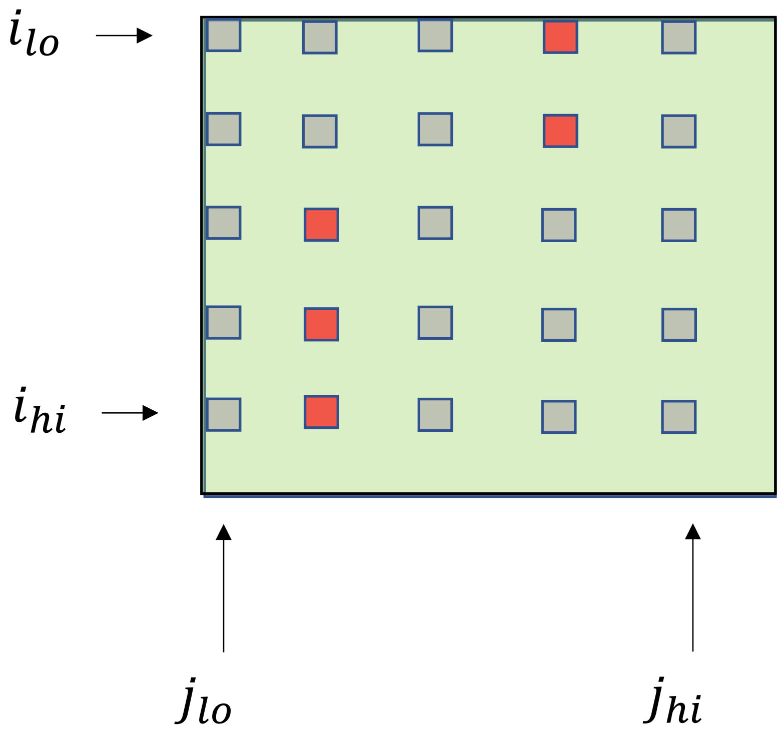
We partition δ to $\frac{n}{L} \times \frac{n}{L}$ grid, each grid contains $L \times L$ points, we only consider the top-left $\delta(i, j)$ point. $L = O(\lg^2 n)$,

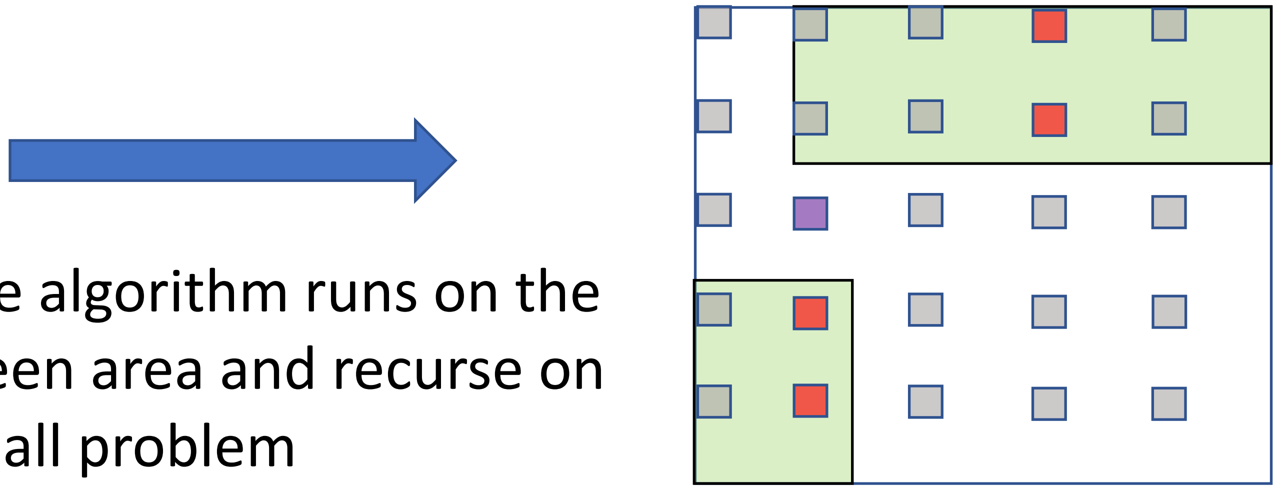
Instead of computing the splitting line, we compute the point that is closest to the splitting line, in each row grid .

We reduce the size of the problem to $O(\frac{n}{L})$

Implicit subunit-Monge matrix multiplication

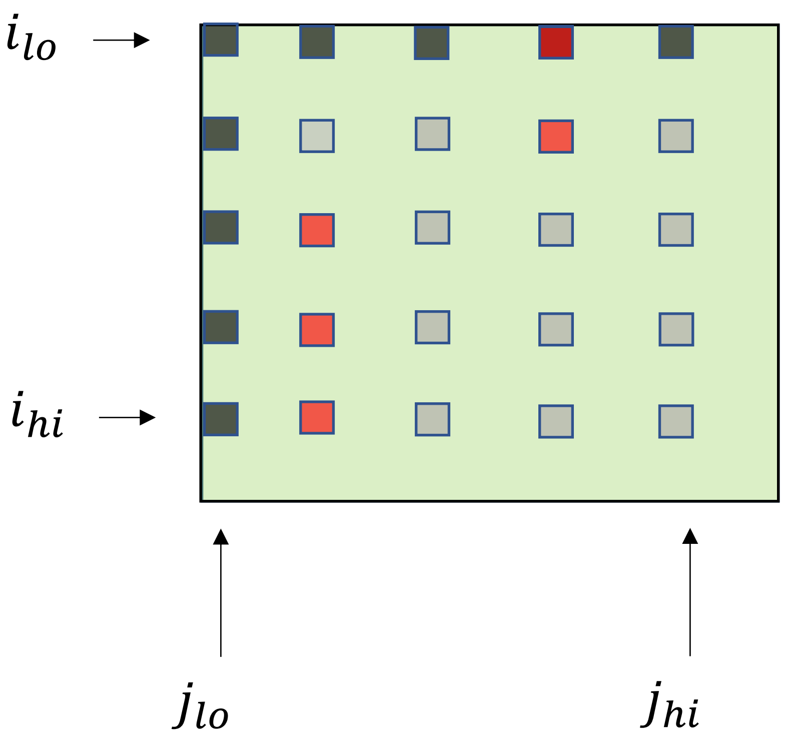
Divide-and-conquer algorithm: Input is an index rectangle $[i_{lo}, i_{hi}] \times [j_{lo}, j_{hi}]$ that splitting line crosses.





The algorithm runs on the green area and recurse on small problem

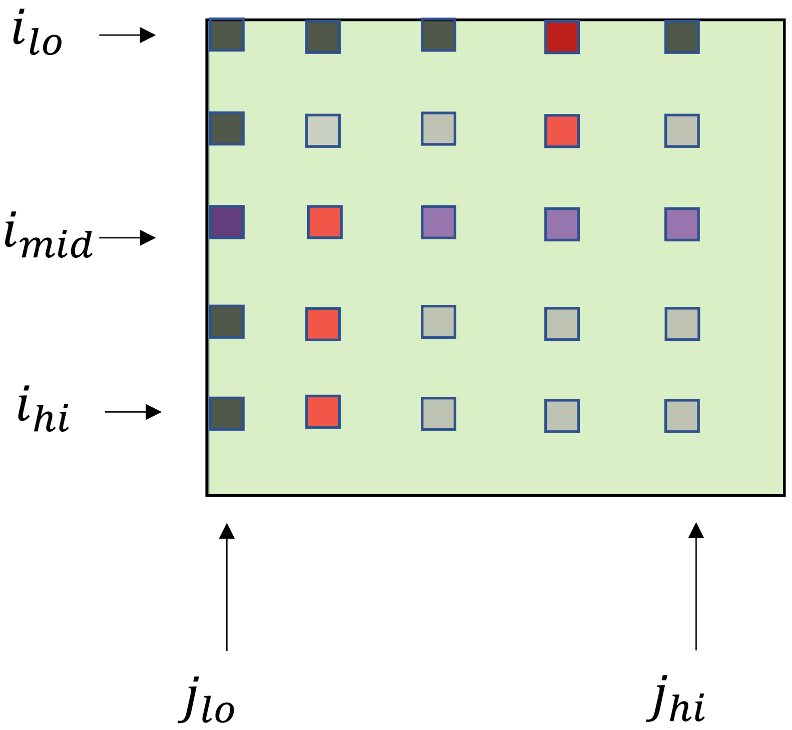
Implicit subunit-Monge matrix multiplication



Divide-and-conquer algorithm: Input is a rectangle $[i_{lo}, i_{hi}] \times [j_{lo}, j_{hi}]$ that splitting line crosses.

To recurse on the small problem, we require the top border and left border $\delta(i, j)$ is computed. Those points are used to compute red points.

Implicit subunit-Monge matrix multiplication

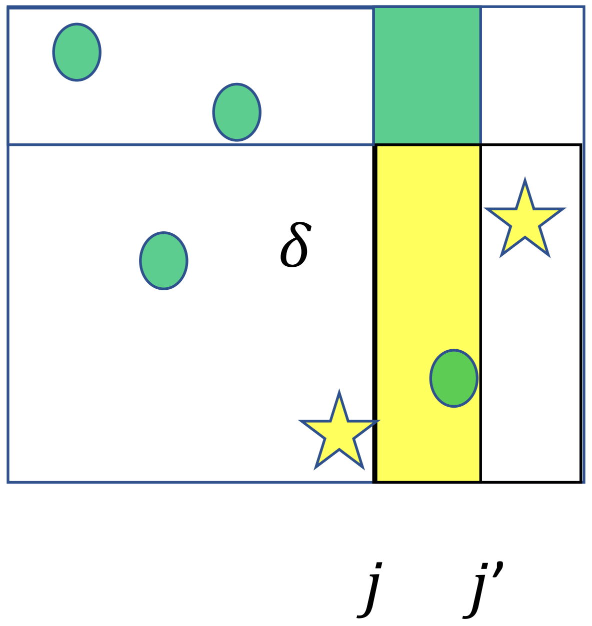


Divide-and-conquer algorithm: Input is a rectangle $[i_{lo}, i_{hi}] \times [j_{lo}, j_{hi}]$ that splitting line crosses.

We require the top border and left border $\delta(i, j)$ is computed.

The we compute the middle line $\delta(i_{mid}, j)$ value, for $j \in [j_{lo}, j_{hi}]$
we can use the mid $\delta(i_{mid}, j_{lo})$ value.

Implicit subunit-Monge matrix multiplication



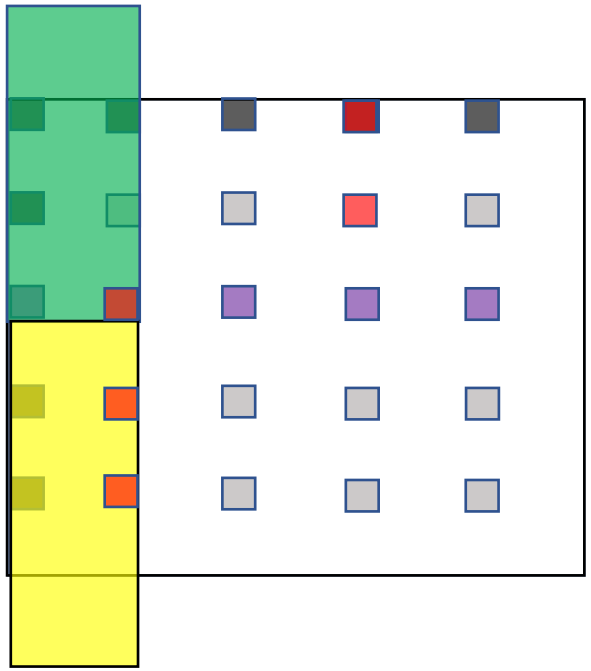
Green circle represents non-zero term of $P_{c,lo}$,

Yellow pentagram represents non-zero term of $P_{c,hi}$

$\delta(i,j') - \delta(i,j) =$ The green area sum of $P_{c,lo}$

+ the yellow area sum of $P_{c,hi}$

Implicit subunit-Monge matrix multiplication

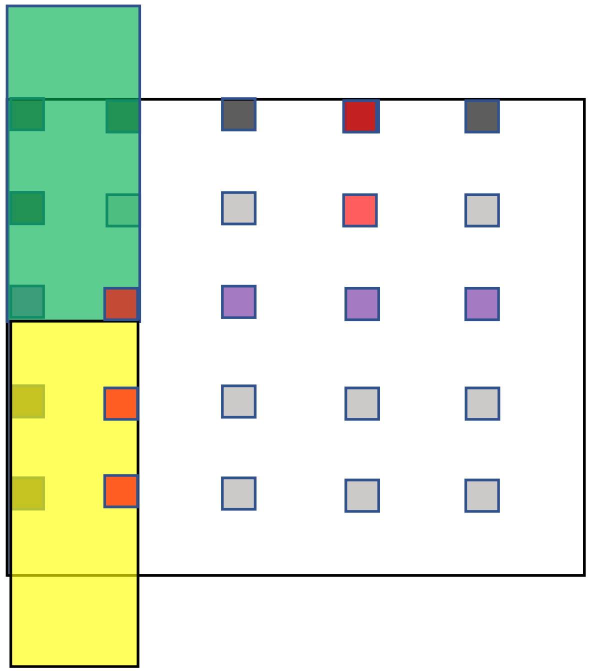


The we compute the mid line $\delta(i_{mid}, j)$ value, for $j \in [j_{lo}, j_{hi}]$ we can use the mid $\delta(i_{mid}, j_{lo})$ value.

We need a data structure to answer The green area sum of $P_{c,lo}$ + the yellow area sum of $P_{c,hi}$.

Key observation: there are only $O(L)$ non-zero elements in this area. We can sort those elements, it takes $O(\mathrm{L} \lg L)$ to construct the data structure and $O(\lg L)$ to make a query.

Implicit subunit-Monge matrix multiplication



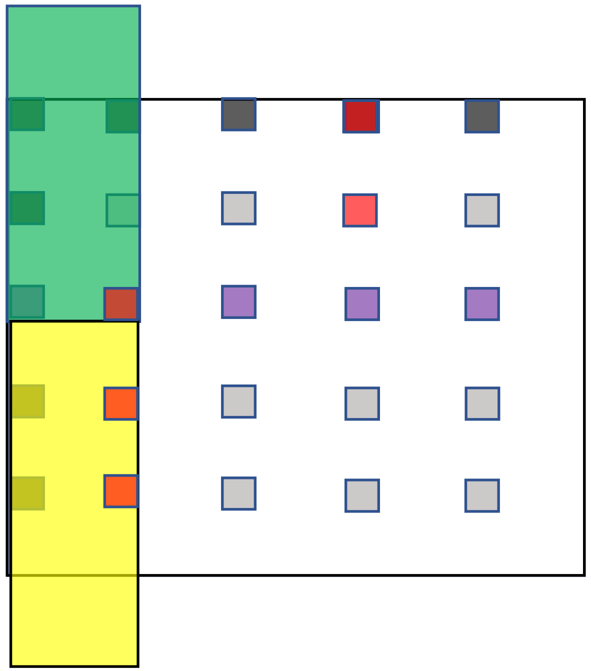
The we compute the mid line $\delta(i_{mid}, j)$ value, for $j \in [j_{lo}, j_{hi}]$ we can use the mid $\delta(i_{mid}, j_{lo})$ value.

We need a data structure to answer The green area sum of $P_{c,lo}$ + the yellow area sum of $P_{c,hi}$.

Key observation: there are only $O(L)$ non-zero elements in this area. We can sort those elements, it takes $O(\mathrm{L}\lg L)$ to construct the data structure and $O(\lg L)$ to make a query.

We have $O(\frac{n}{L})$ such data structure, so it takes $O(\mathrm{n}\lg L)$ times to construct all data structure.

Implicit subunit-Monge matrix multiplication



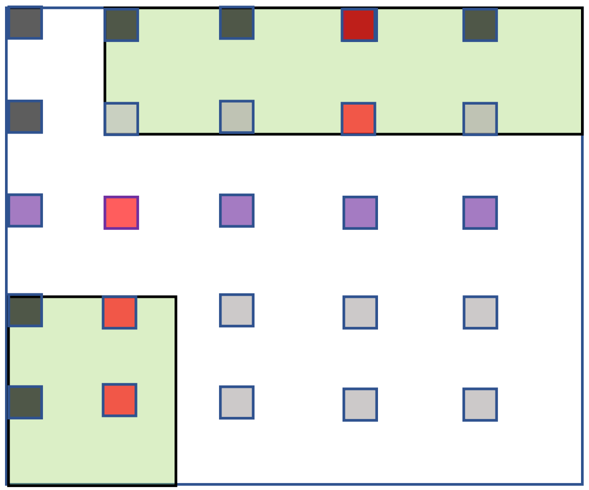
The we compute the mid line $\delta(i_{mid}, j)$ value, for $j \in [j_{lo}, j_{hi}]$ we can use the mid $\delta(i_{mid}, j_{lo})$ value.

We need a data structure to answer The green area sum of $P_{c,lo}$ + the yellow area sum of $P_{c,hi}$.

Key observation: there is only $O(L)$ non-zero elements in this area. We can sort those elements, it takes $O(\mathrm{L} \lg L)$ to construct the data structure and $O(\lg L)$ to make a query.

Last observation: sort $L = O(\lg^2 n)$ elements can be done in $O(L)$ work if we use integer sort. The data structure can be Constructed in $O(n)$ work.

Implicit subunit-Monge matrix multiplication



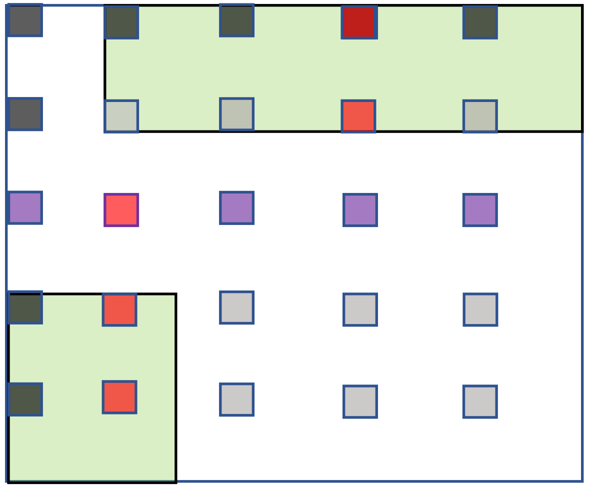
Divide-and-conquer algorithm: Input is a rectangle $[i_{lo}, i_{hi}] \times [j_{lo}, j_{hi}]$ that splitting line crosses.

Then we recurse on the two subproblems, each recursion decreases the rectangle area by 2. We have at most $O(\lg n)$ level of recursion. We only have $O(\frac{n}{L})$ elements.

The total work is $O(\frac{n}{L} \lg n \ \lg L)$ and span is $O(\lg^2 n)$.

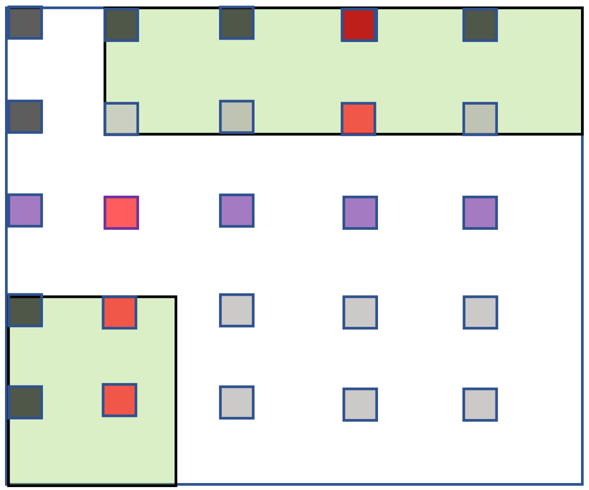
Computing data structure takes $O(n)$ work.

Implicit subunit-Monge matrix multiplication



Given $P_{c,lo}$ and $P_{c,hi}$, one can compute P_c in $O(n)$ work and $O(\lg^2 n)$ span.

Implicit subunit-Monge matrix multiplication



Given $P_{c,lo}$ and $P_{c,hi}$, one can compute P_c in $O(n)$ work and $O(\lg^2 n)$ span.

↓

There is a parallel algorithm solving the ISMMM problem in $O(nlgn)$ work and $O(\lg^3 n)$ span.

↓

There is a parallel algorithm that computes an LIS in $O(nlg^2n)$ work and $O(\lg^4 n)$ span.

Future work

- “Rank” of all elements?
  - i-th “rank” is the LIS ending at i-th element
  - Fast parallel dynamic LIS. (sequential: [Kociumaka & Seddighin 2021])
- Weighted LIS?
- Sublinear time parallel approximated LIS [Andoni, Nosatzki, Sinha, & Stein 2022]
- Work-Optimal with polylog(n) span LIS?

Q & A