HPCA project

Batch Sort et Merge Path Sort

AlGORITHM OVERVIEW

POLYTECH SORBONNE

SORBONNE UNIVERSITÉ

Saulnier Solène Robin Clément MAIN5 2020/2021

PROBLEM TO SOLVE

How to sort efficiently an array of elements **using GPU** ?

variable names :

- A,B and M are array of int
- A,B are sorted
- We save our result in M
- M is not sorted on the initialisation

TABLES OF CONTENTS

01

MERGE ON ONE BLOCK

Of two sorted array,
A and B

02

MERGE ON SEVERAL BLOCKS

Of two sorted array,
A and B

03

“MERGE SORT” ON GPU

tree approach

04

SORT WITH BATCHES

tree approach, batches
and streams

01

MERGE ON ONE BLOCK

ALGO mergeSmall_k

A

1	2	5	6	6	9	11	15	16

B

4	7	8	10	12	13	14

ALGO mergeSmall_k

A \ B	4	7	8	10	12	13	14
1							
2							
5							
6							
6							
9							
11							
15							
16							

ALGO mergeSmall_k

A \ B	4	7	8	10	12	13	14
1	1	1	1	1	1	1	1
2	1	1	1	1	1	1	1
5	0	1	1	1	1	1	1
6	0	1	1	1	1	1	1
6	0	1	1	1	1	1	1
9	0	0	0	1	1	1	1
11	0	0	0	0	1	1	1
15	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0

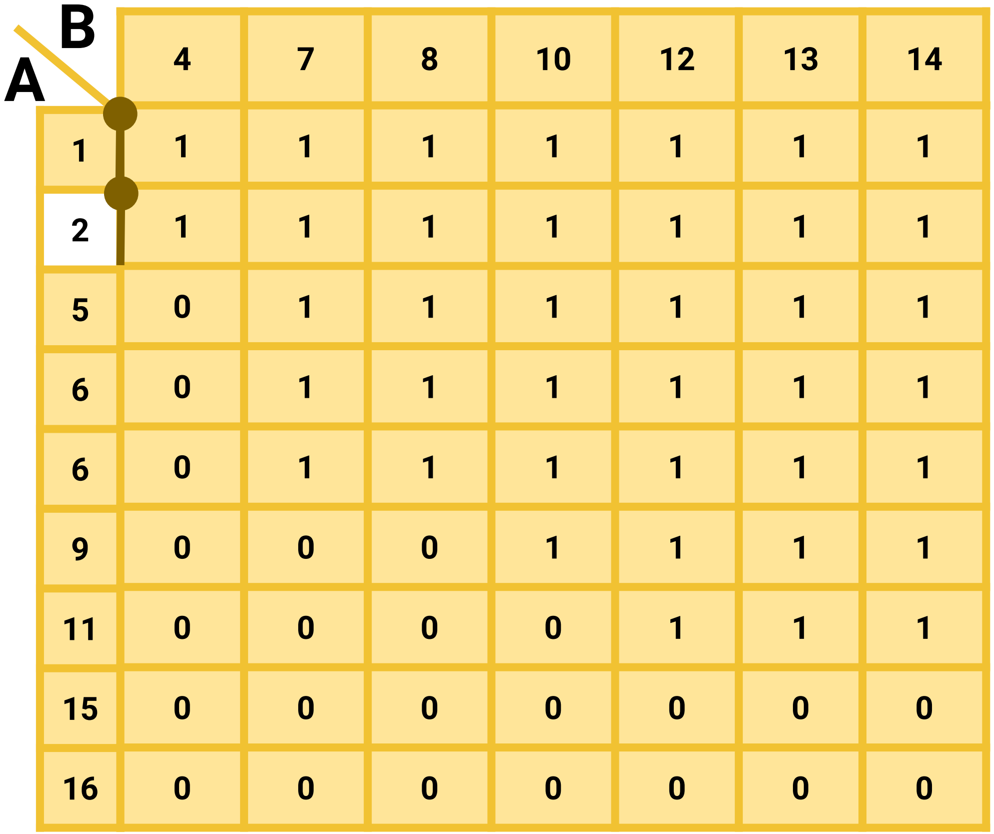
ALGO mergeSmall_k

B A	4	7	8	10	12	13	14
1	1	1	1	1	1	1	1
2	1	1	1	1	1	1	1
5	0	1	1	1	1	1	1
6	0	1	1	1	1	1	1
6	0	1	1	1	1	1	1
9	0	0	0	1	1	1	1
11	0	0	0	0	1	1	1
15	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0

M

1

ALGO mergeSmall_k

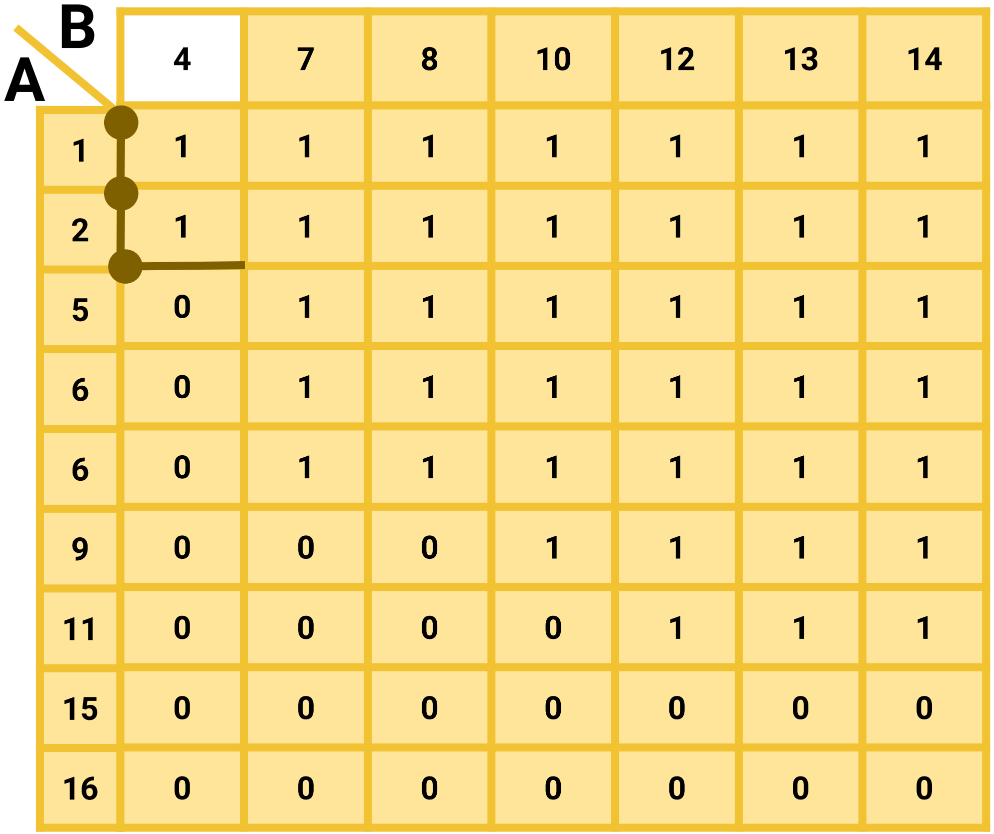


A \ B	4	7	8	10	12	13	14
1	1	1	1	1	1	1	1
2	1	1	1	1	1	1	1
5	0	1	1	1	1	1	1
6	0	1	1	1	1	1	1
6	0	1	1	1	1	1	1
9	0	0	0	1	1	1	1
11	0	0	0	0	1	1	1
15	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0

M

1	2

ALGO mergeSmall_k

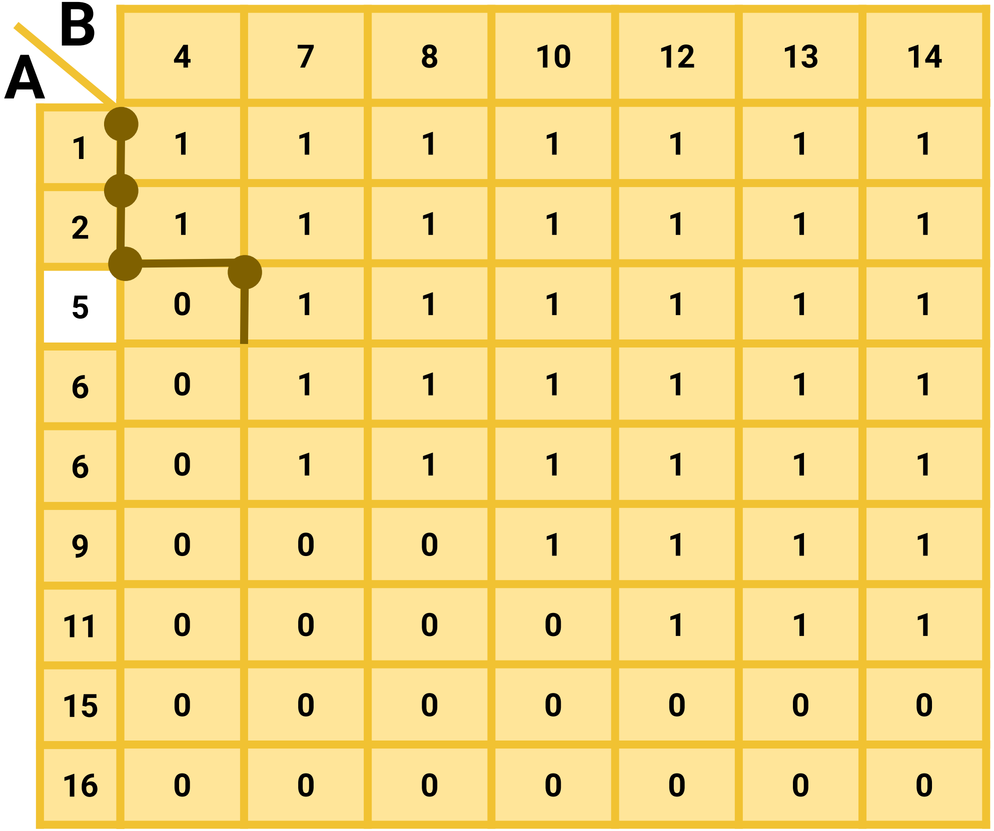


A \ B	4	7	8	10	12	13	14
1	1	1	1	1	1	1	1
2	1	1	1	1	1	1	1
5	0	1	1	1	1	1	1
6	0	1	1	1	1	1	1
6	0	1	1	1	1	1	1
9	0	0	0	1	1	1	1
11	0	0	0	0	1	1	1
15	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0

M

1	2	4

ALGO mergeSmall_k

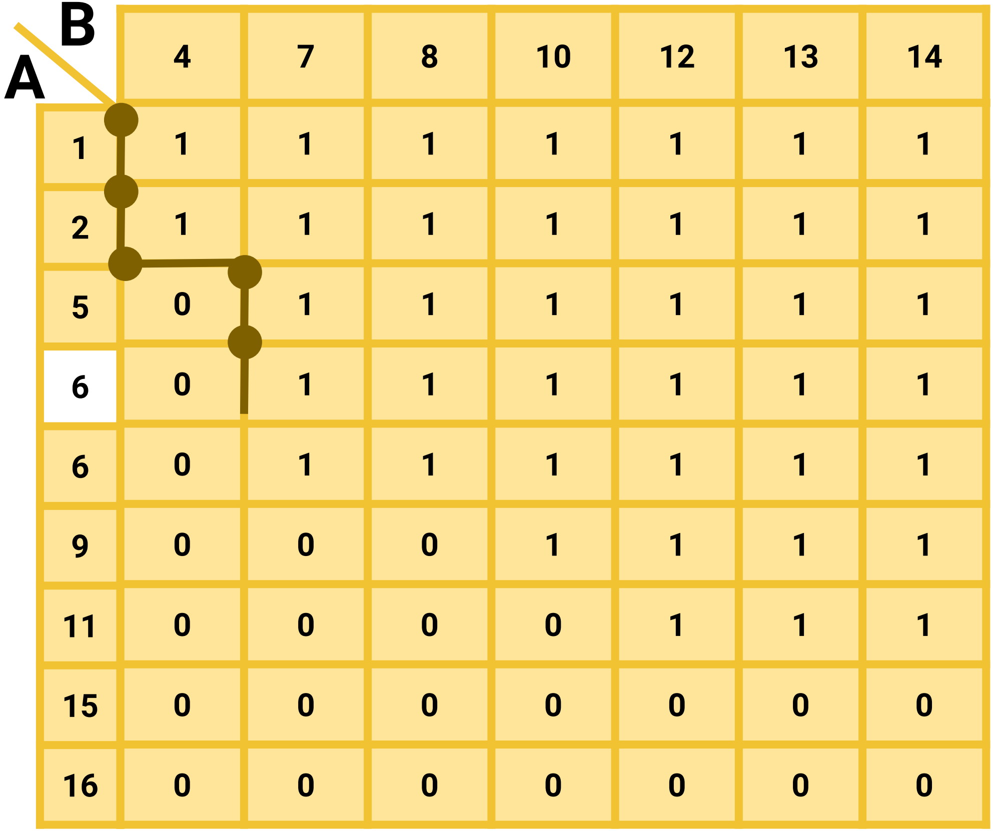


A \ B	4	7	8	10	12	13	14
1	1	1	1	1	1	1	1
2	1	1	1	1	1	1	1
5	0	1	1	1	1	1	1
6	0	1	1	1	1	1	1
6	0	1	1	1	1	1	1
9	0	0	0	1	1	1	1
11	0	0	0	0	1	1	1
15	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0

M

1	2	4	5

ALGO mergeSmall_k

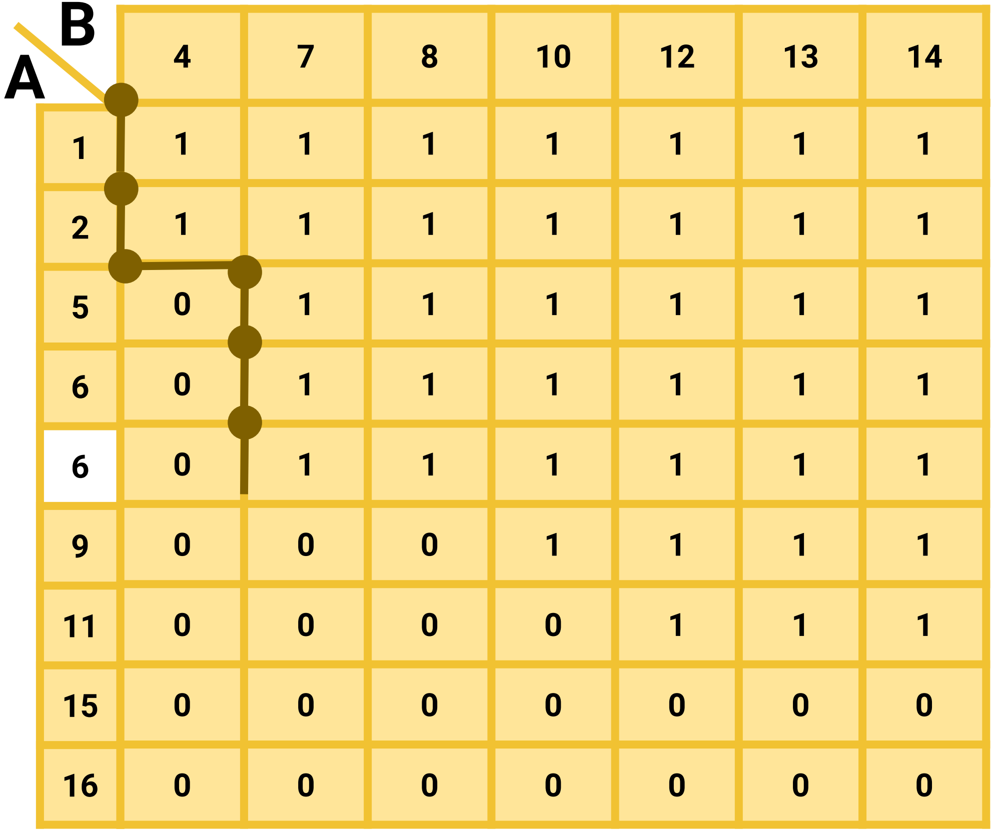


A \ B	4	7	8	10	12	13	14
1	1	1	1	1	1	1	1
2	1	1	1	1	1	1	1
5	0	1	1	1	1	1	1
6	0	1	1	1	1	1	1
6	0	1	1	1	1	1	1
9	0	0	0	1	1	1	1
11	0	0	0	0	1	1	1
15	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0

M

1	2	4	5	6

ALGO mergeSmall_k



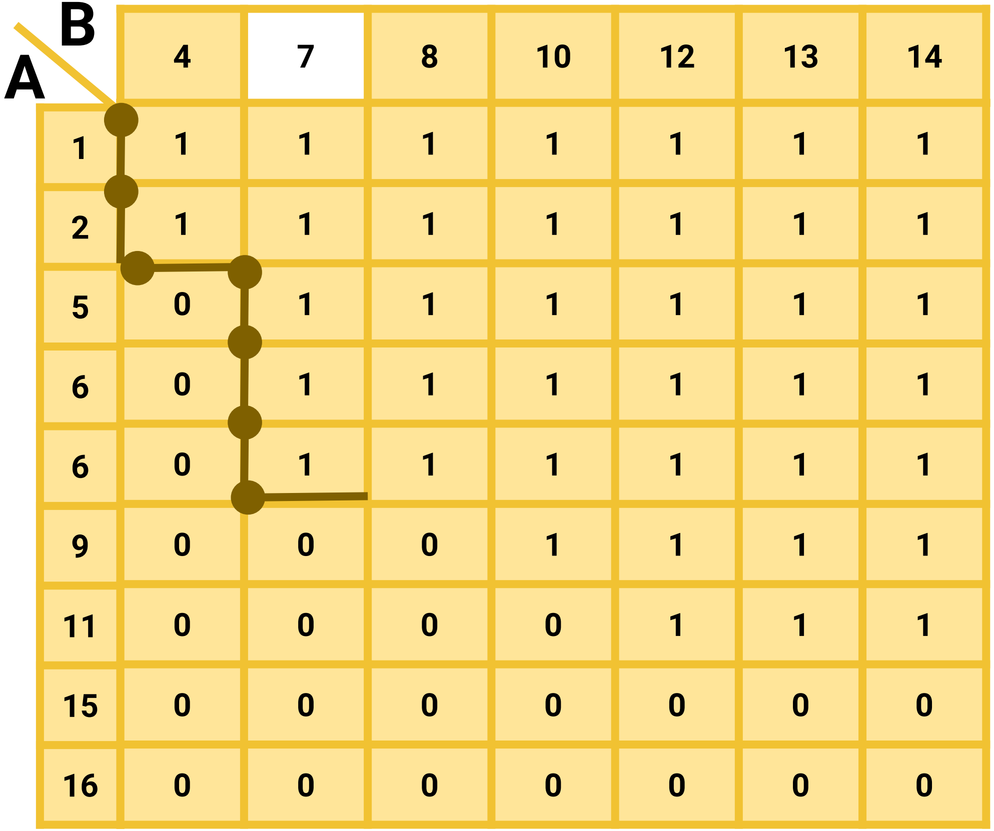
A \ B	4	7	8	10	12	13	14
1	1	1	1	1	1	1	1
2	1	1	1	1	1	1	1
5	0	1	1	1	1	1	1
6	0	1	1	1	1	1	1
6	0	1	1	1	1	1	1
9	0	0	0	1	1	1	1
11	0	0	0	0	1	1	1
15	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0

M

1	2	4	5	6

… 6

ALGO mergeSmall_k



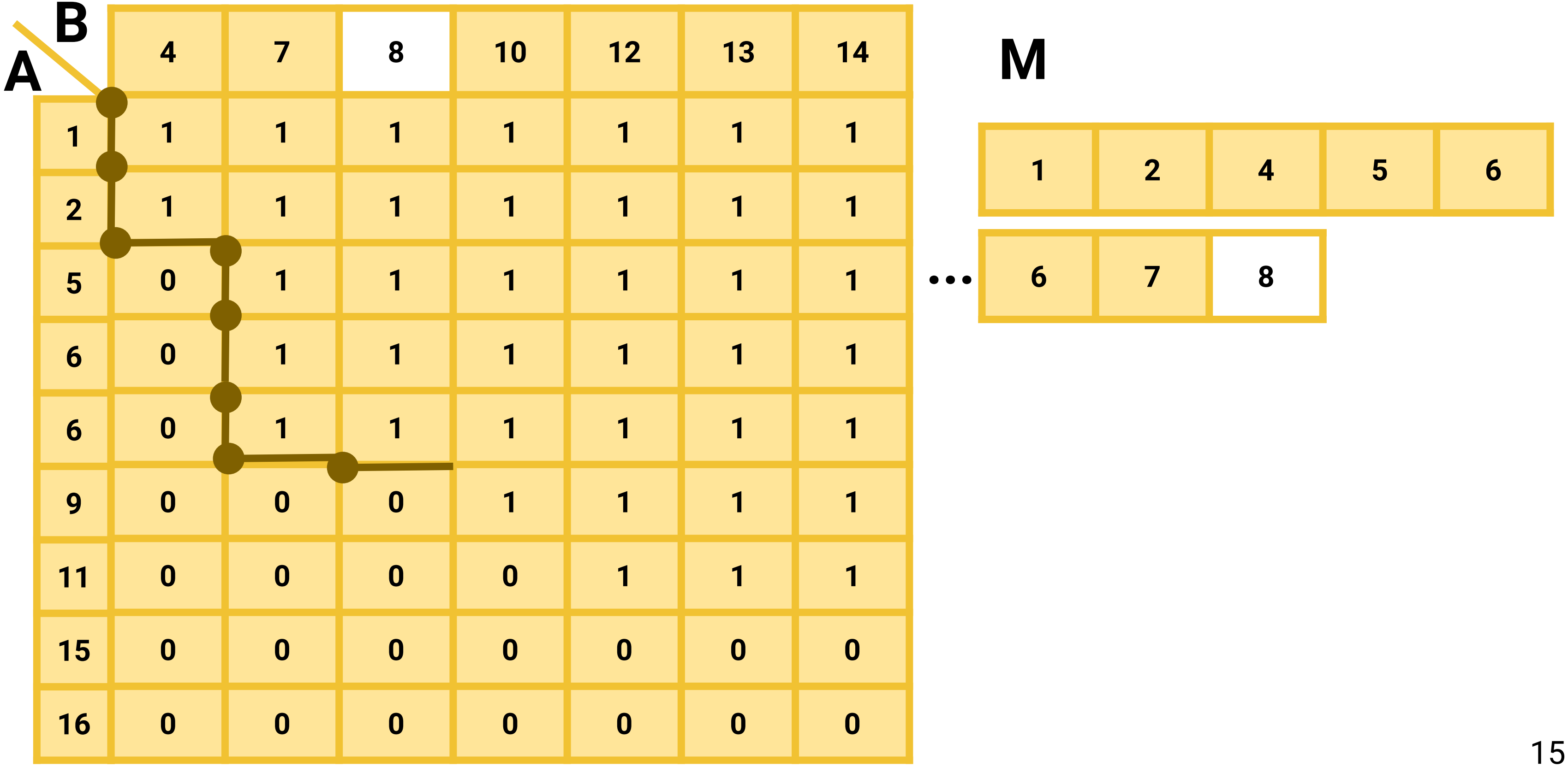
A \ B	4	7	8	10	12	13	14
1	1	1	1	1	1	1	1
2	1	1	1	1	1	1	1
5	0	1	1	1	1	1	1
6	0	1	1	1	1	1	1
6	0	1	1	1	1	1	1
9	0	0	0	1	1	1	1
11	0	0	0	0	1	1	1
15	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0

M

1	2	4	5	6

...	6	7

ALGO mergeSmall_k



A \ B	4	7	8	10	12	13	14
1	1	1	1	1	1	1	1
2	1	1	1	1	1	1	1
5	0	1	1	1	1	1	1
6	0	1	1	1	1	1	1
6	0	1	1	1	1	1	1
9	0	0	0	1	1	1	1
11	0	0	0	0	1	1	1
15	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0

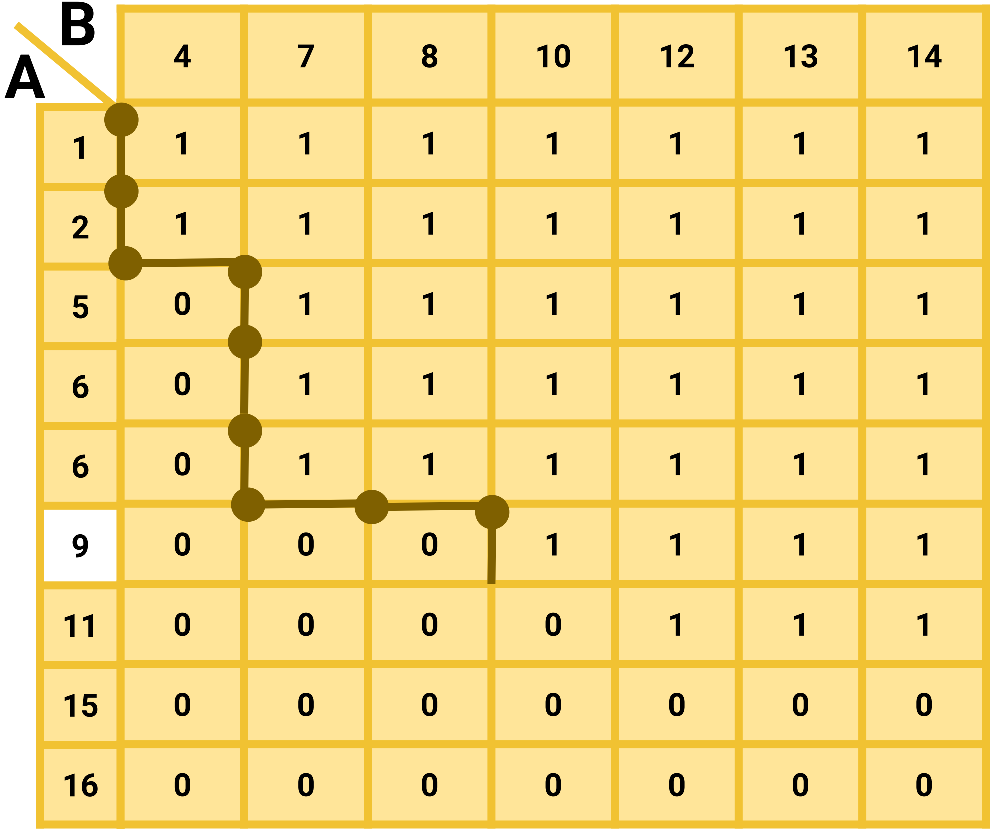
M

1	2	4	5	6

…

6	7	8

ALGO mergeSmall_k



A \ B	4	7	8	10	12	13	14
1	1	1	1	1	1	1	1
2	1	1	1	1	1	1	1
5	0	1	1	1	1	1	1
6	0	1	1	1	1	1	1
6	0	1	1	1	1	1	1
9	0	0	0	1	1	1	1
11	0	0	0	0	1	1	1
15	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0

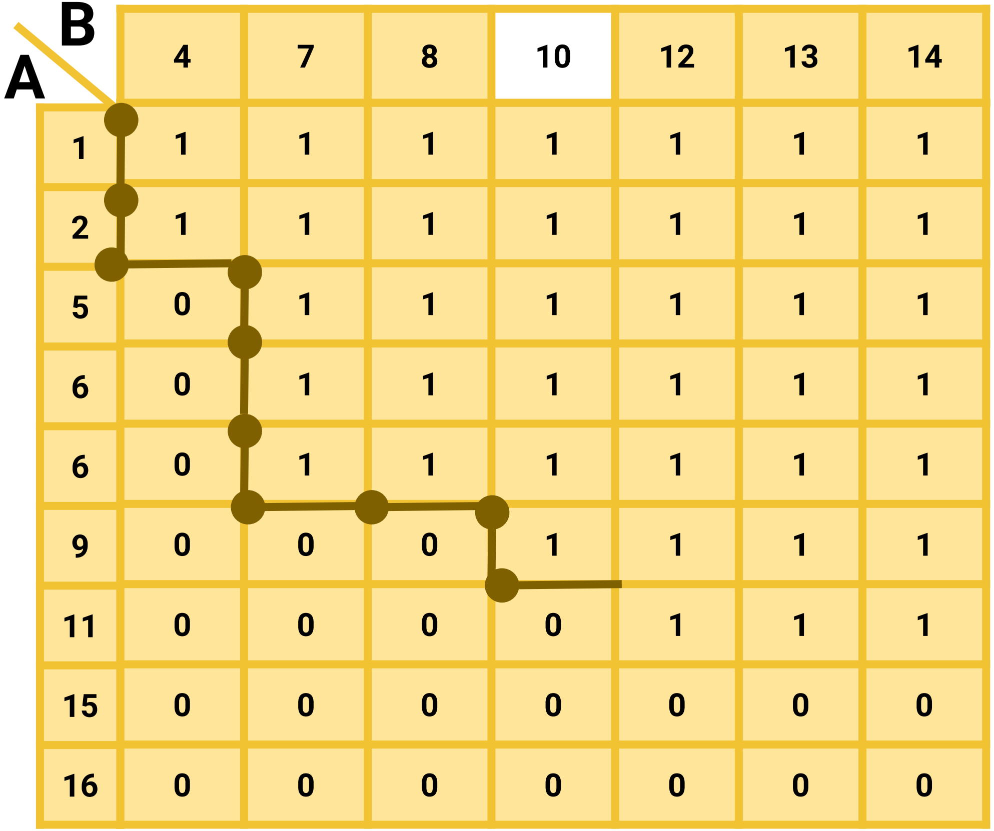
M

1	2	4	5	6

…

6	7	8	9

ALGO mergeSmall_k



A \ B	4	7	8	10	12	13	14
1	1	1	1	1	1	1	1
2	1	1	1	1	1	1	1
5	0	1	1	1	1	1	1
6	0	1	1	1	1	1	1
6	0	1	1	1	1	1	1
9	0	0	0	1	1	1	1
11	0	0	0	0	1	1	1
15	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0

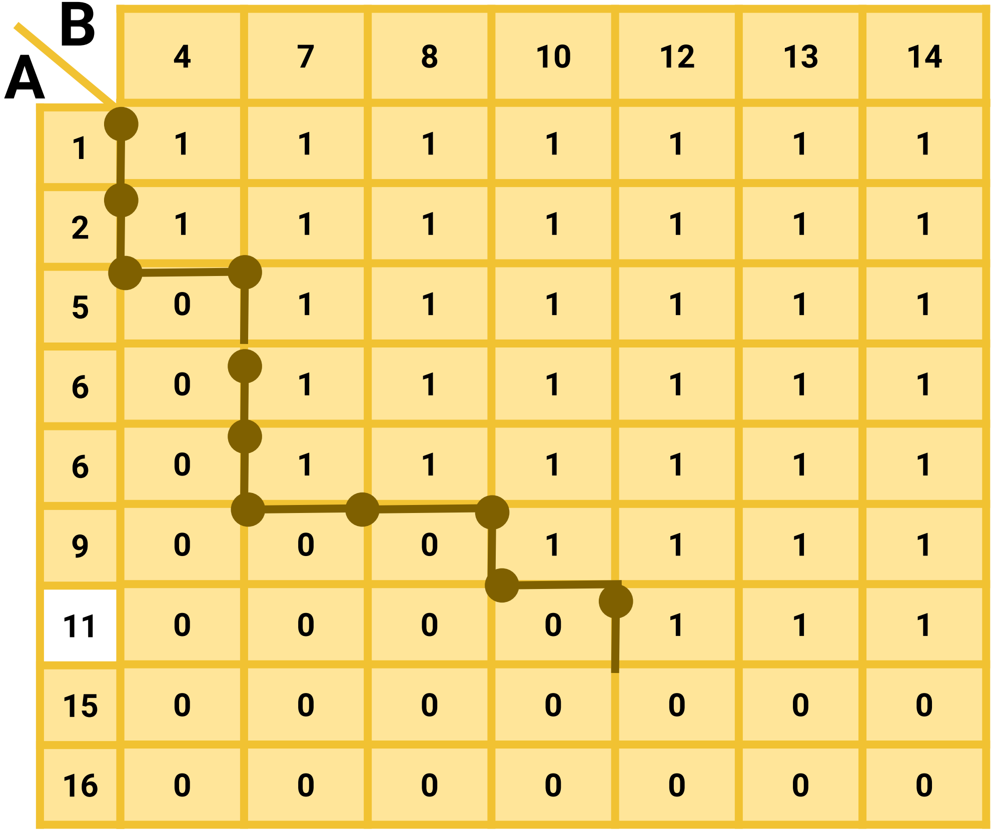
M

1	2	4	5	6

…

6	7	8	9	10

ALGO mergeSmall_k



A \ B	4	7	8	10	12	13	14
1	1	1	1	1	1	1	1
2	1	1	1	1	1	1	1
5	0	1	1	1	1	1	1
6	0	1	1	1	1	1	1
6	0	1	1	1	1	1	1
9	0	0	0	1	1	1	1
11	0	0	0	0	1	1	1
15	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0

M

1	2	4	5	6

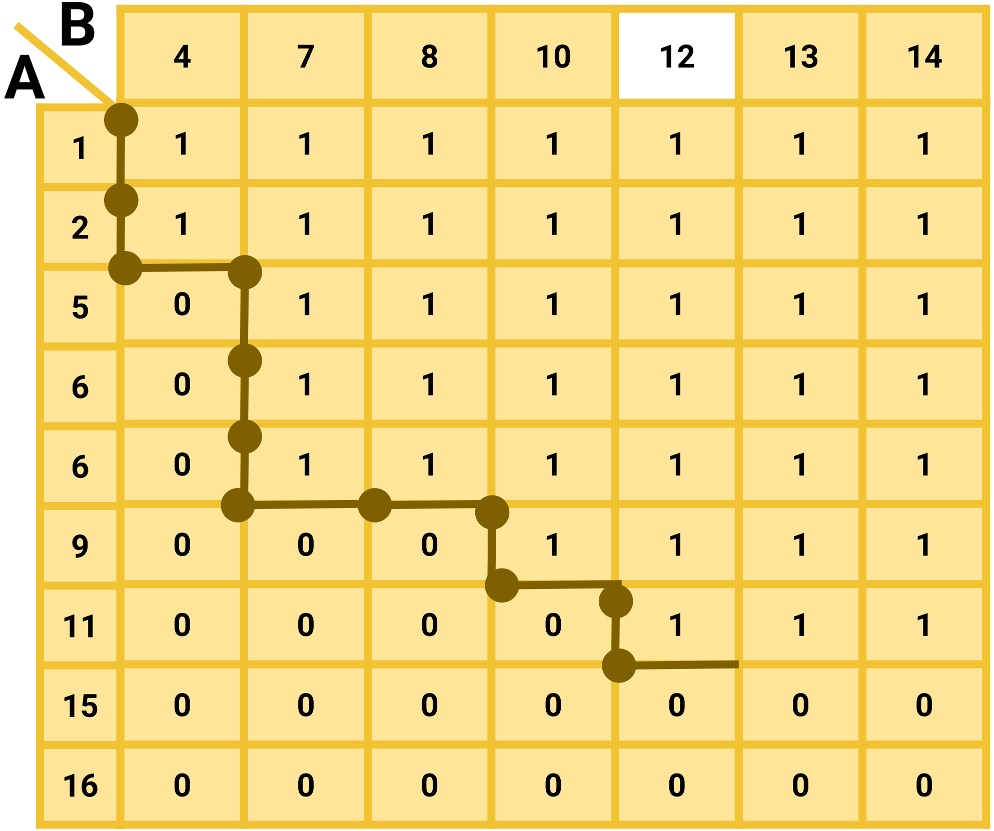
...

6	7	8	9	10

...

11

ALGO mergeSmall_k

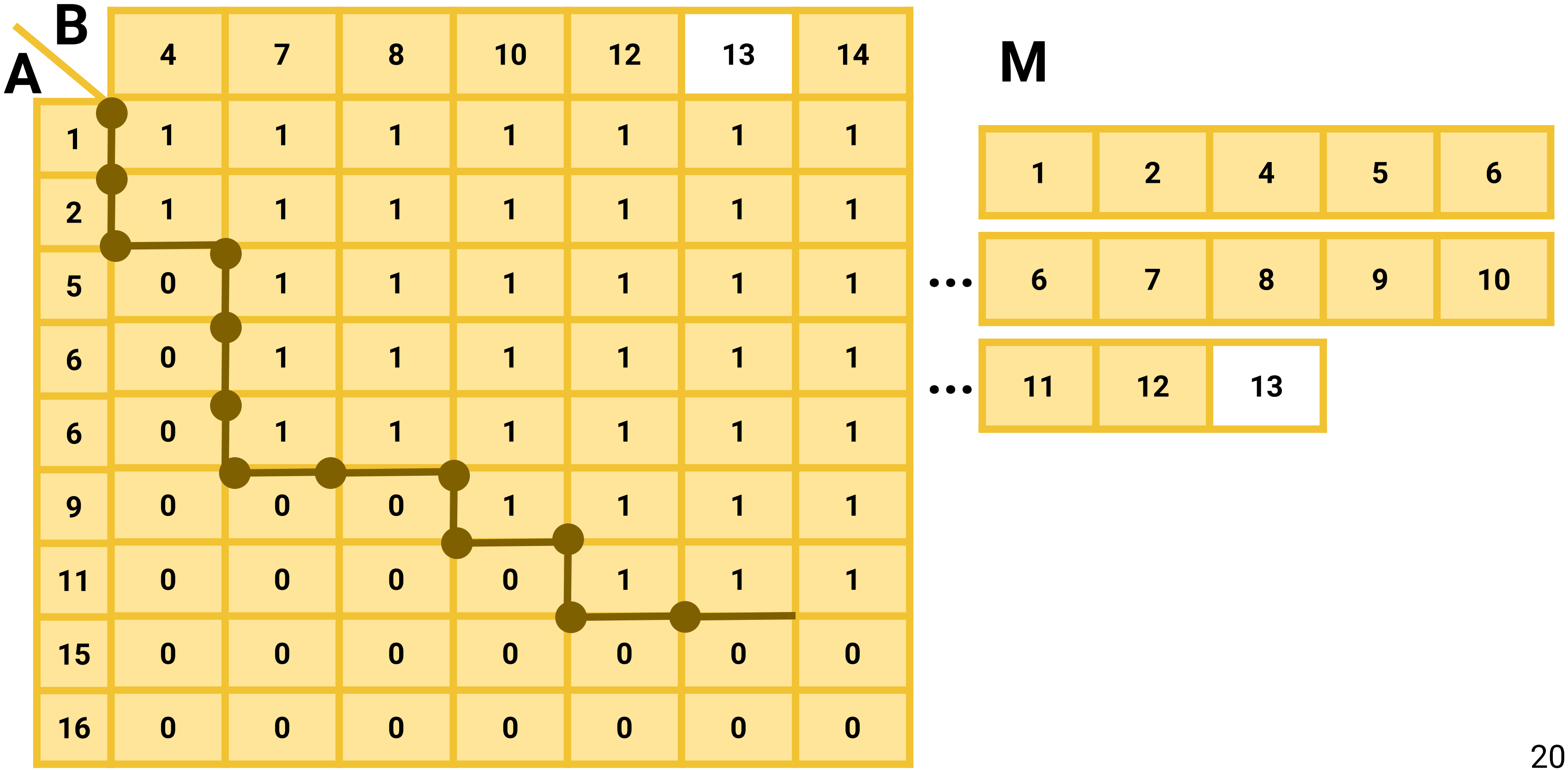


B A	4	7	8	10	12	13	14
1	1	1	1	1	1	1	1
2	1	1	1	1	1	1	1
5	0	1	1	1	1	1	1
6	0	1	1	1	1	1	1
6	0	1	1	1	1	1	1
9	0	0	0	1	1	1	1
11	0	0	0	0	1	1	1
15	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0

M

	1	2	4	5	6
…	6	7	8	9	10
…	11	12			

ALGO mergeSmall_k

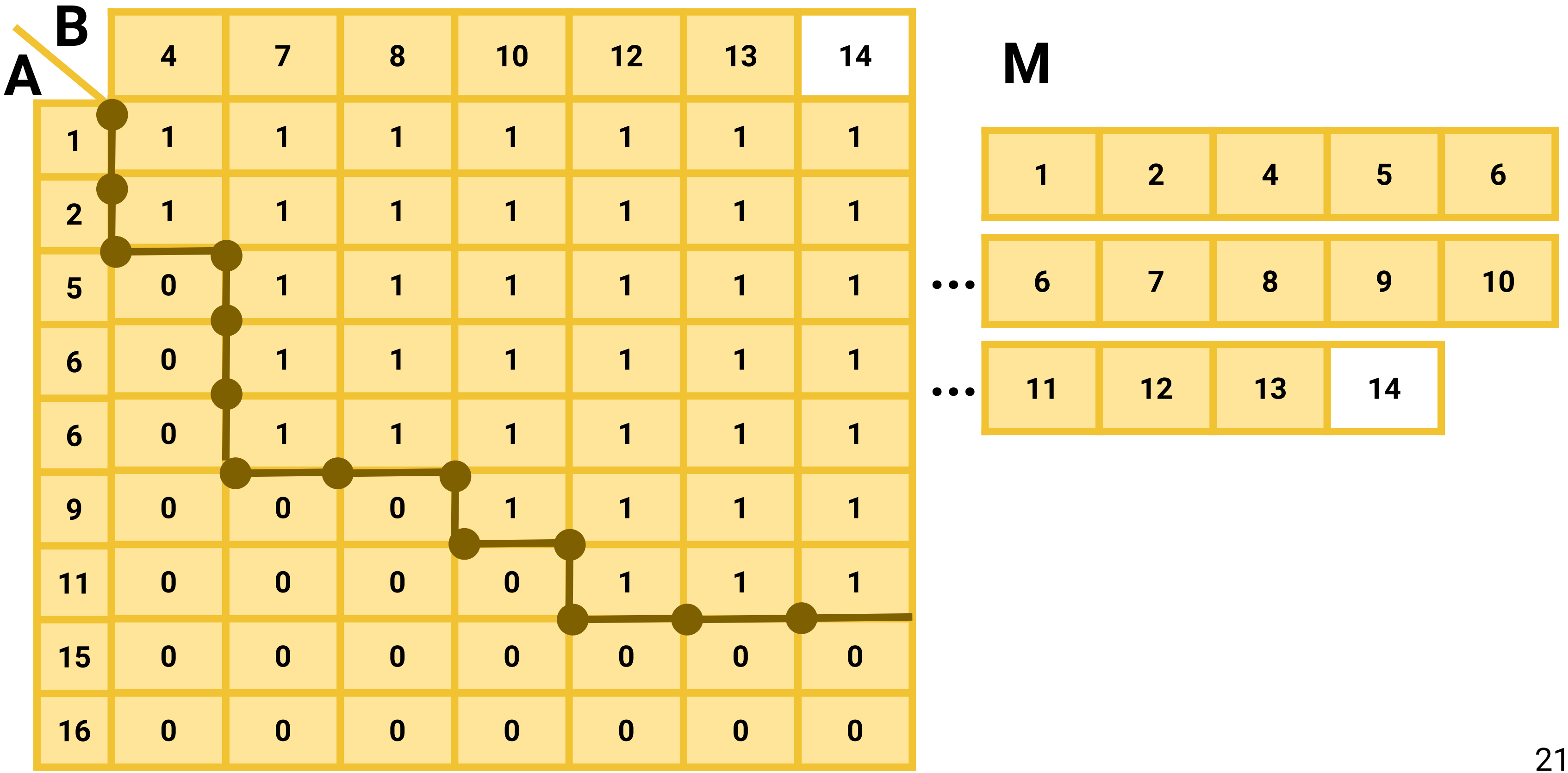


A \ B	4	7	8	10	12	13	14
1	1	1	1	1	1	1	1
2	1	1	1	1	1	1	1
5	0	1	1	1	1	1	1
6	0	1	1	1	1	1	1
6	0	1	1	1	1	1	1
9	0	0	0	1	1	1	1
11	0	0	0	0	1	1	1
15	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0

M

	1	2	4	5	6
•••	6	7	8	9	10
•••	11	12	13		

ALGO mergeSmall_k

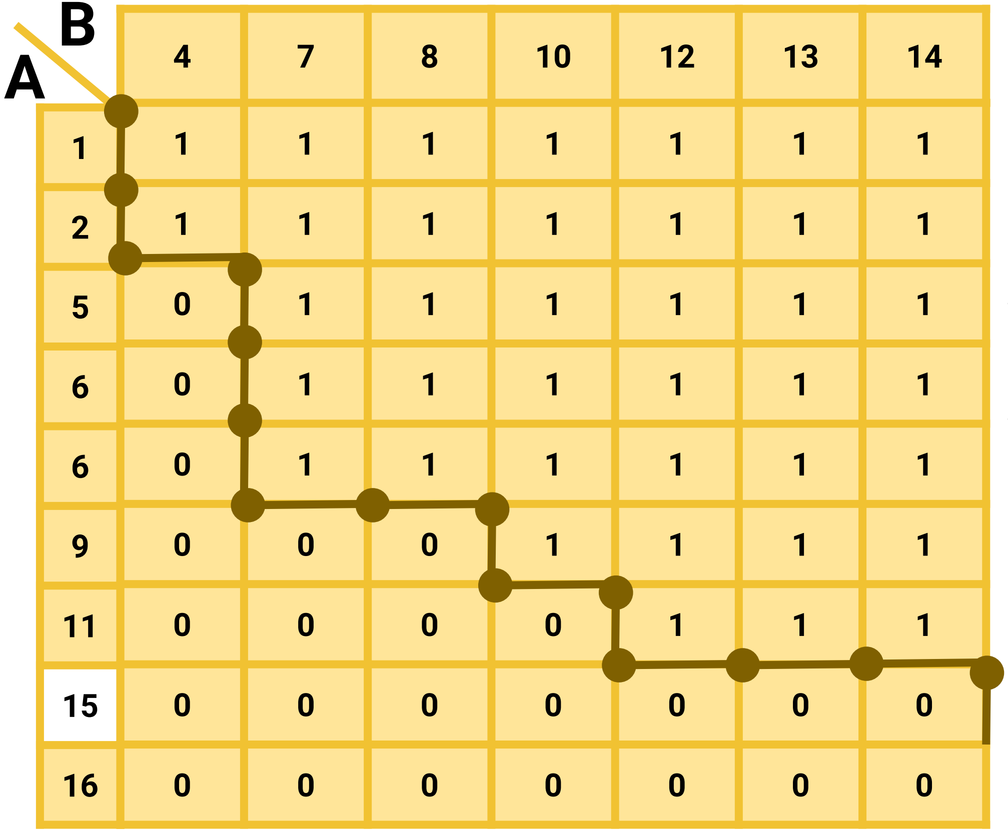


A \ B	4	7	8	10	12	13	14
1	1	1	1	1	1	1	1
2	1	1	1	1	1	1	1
5	0	1	1	1	1	1	1
6	0	1	1	1	1	1	1
6	0	1	1	1	1	1	1
9	0	0	0	1	1	1	1
11	0	0	0	0	1	1	1
15	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0

M

	1	2	4	5	6
•••	6	7	8	9	10
•••	11	12	13	14	

ALGO mergeSmall_k

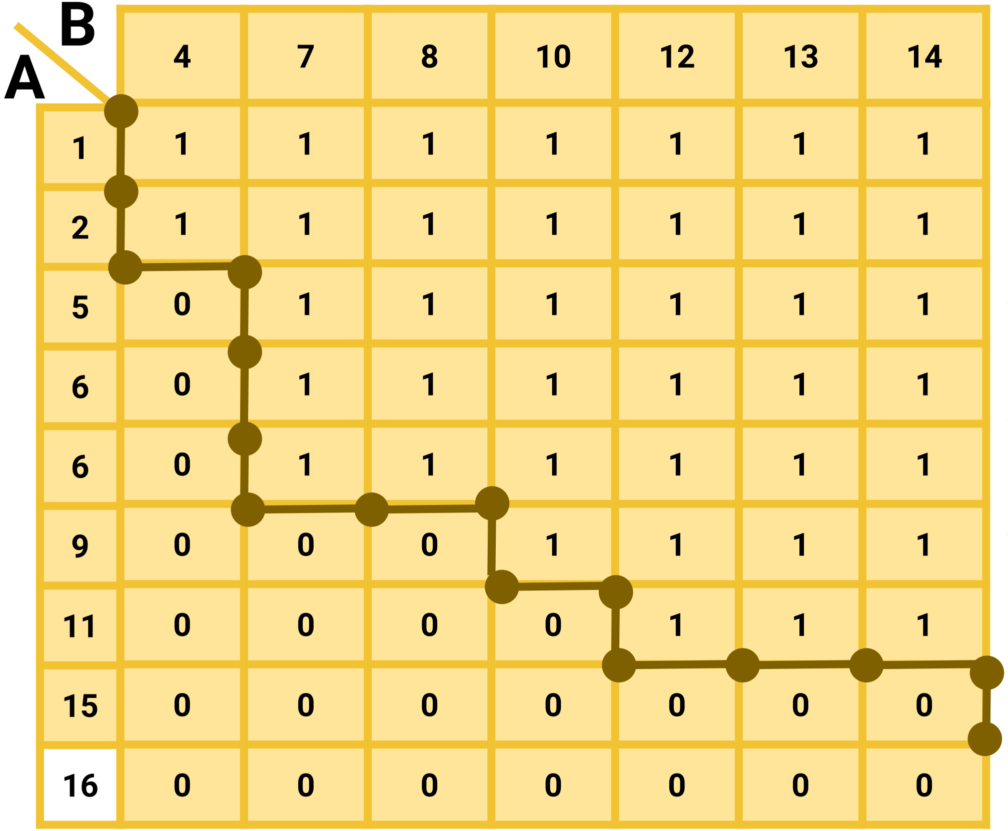


A \ B	4	7	8	10	12	13	14
1	1	1	1	1	1	1	1
2	1	1	1	1	1	1	1
5	0	1	1	1	1	1	1
6	0	1	1	1	1	1	1
6	0	1	1	1	1	1	1
9	0	0	0	1	1	1	1
11	0	0	0	0	1	1	1
15	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0

M

	1	2	4	5	6
...	6	7	8	9	10
...	11	12	13	14	15

ALGO mergeSmall_k

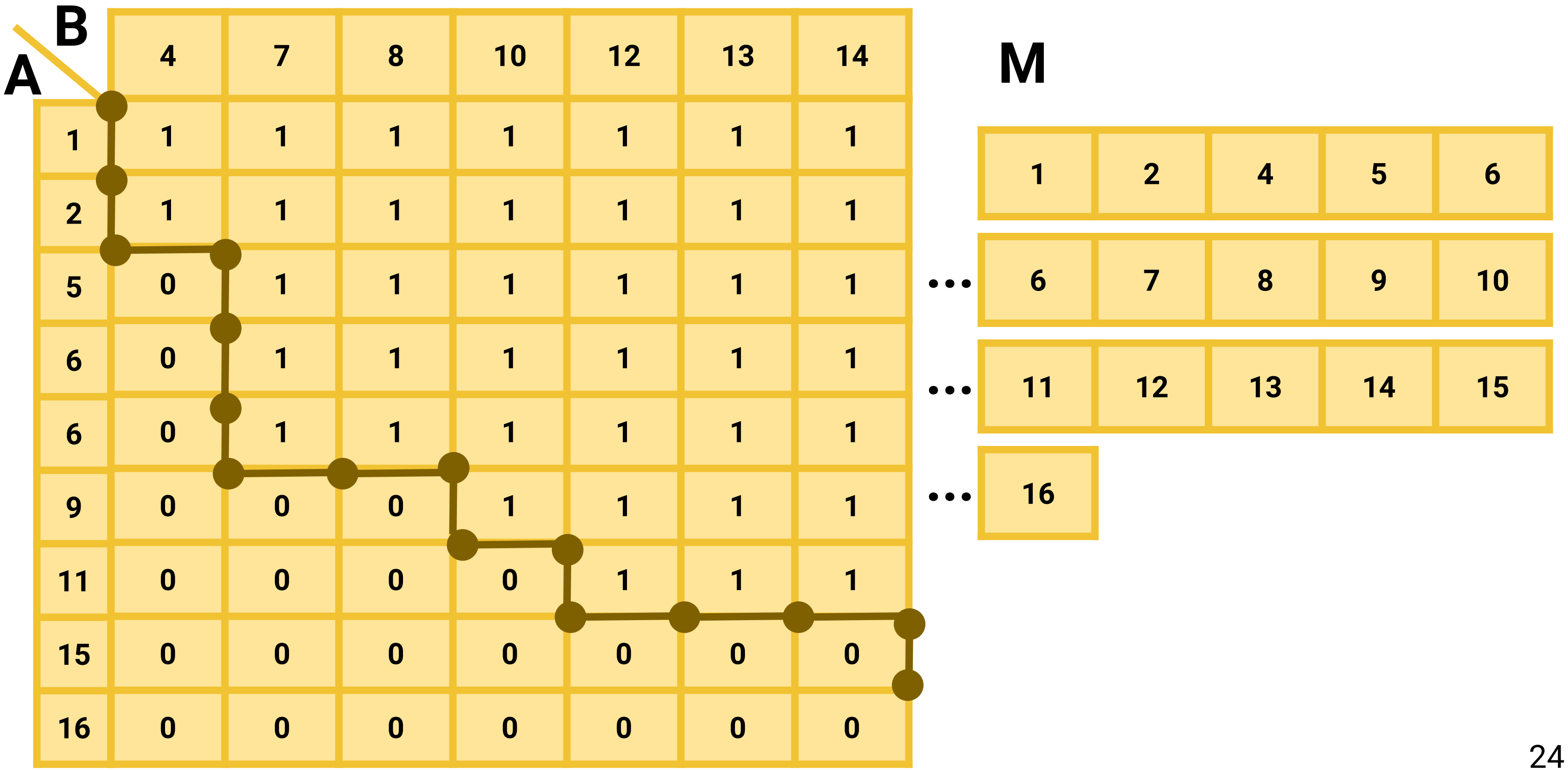


A \ B	4	7	8	10	12	13	14
1	1	1	1	1	1	1	1
2	1	1	1	1	1	1	1
5	0	1	1	1	1	1	1
6	0	1	1	1	1	1	1
6	0	1	1	1	1	1	1
9	0	0	0	1	1	1	1
11	0	0	0	0	1	1	1
15	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0

M

	1	2	4	5	6
•••	6	7	8	9	10
•••	11	12	13	14	15
•••	16				

ALGO mergeSmall_k

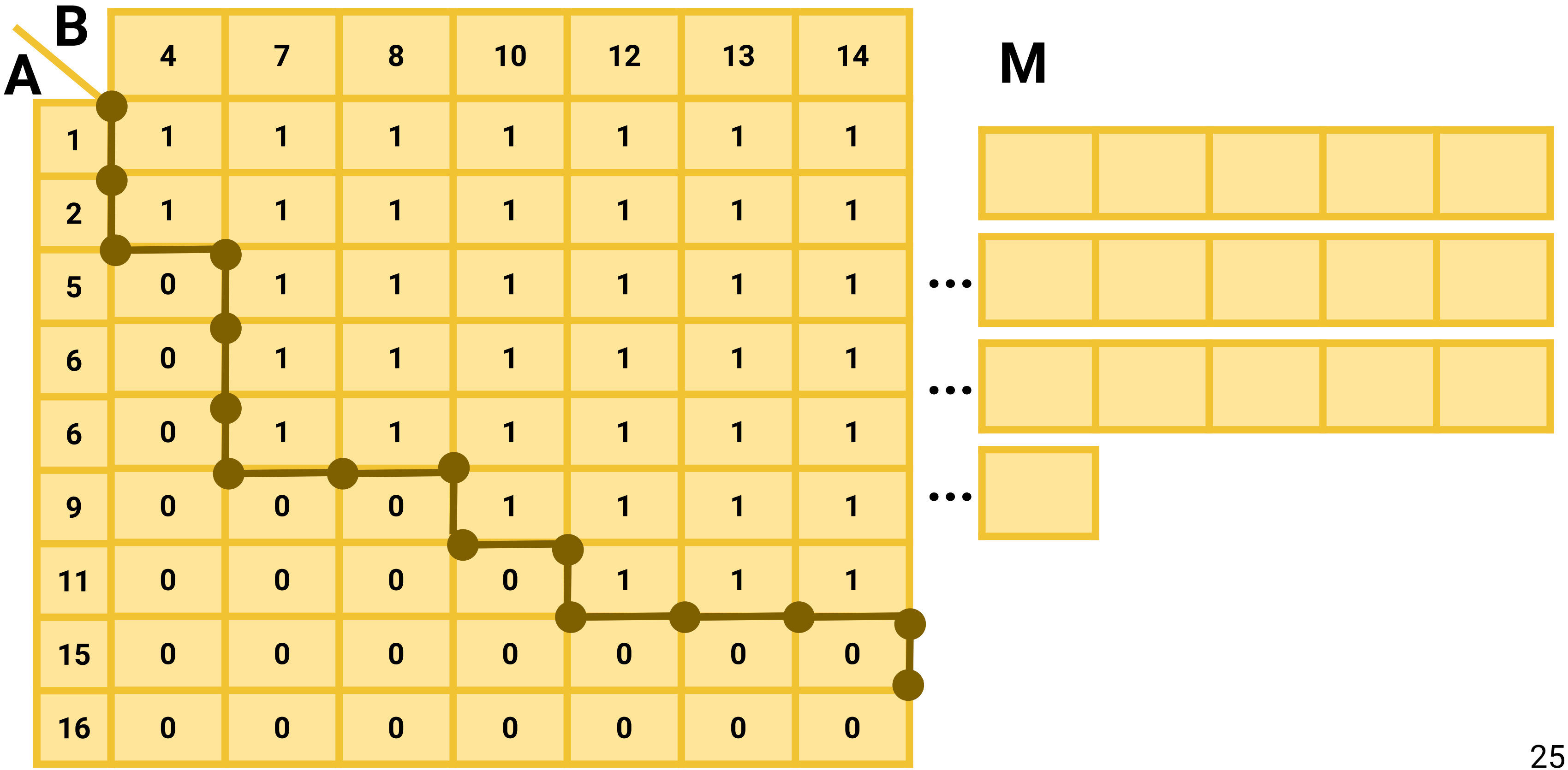


A \ B	4	7	8	10	12	13	14
1	1	1	1	1	1	1	1
2	1	1	1	1	1	1	1
5	0	1	1	1	1	1	1
6	0	1	1	1	1	1	1
6	0	1	1	1	1	1	1
9	0	0	0	1	1	1	1
11	0	0	0	0	1	1	1
15	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0

M

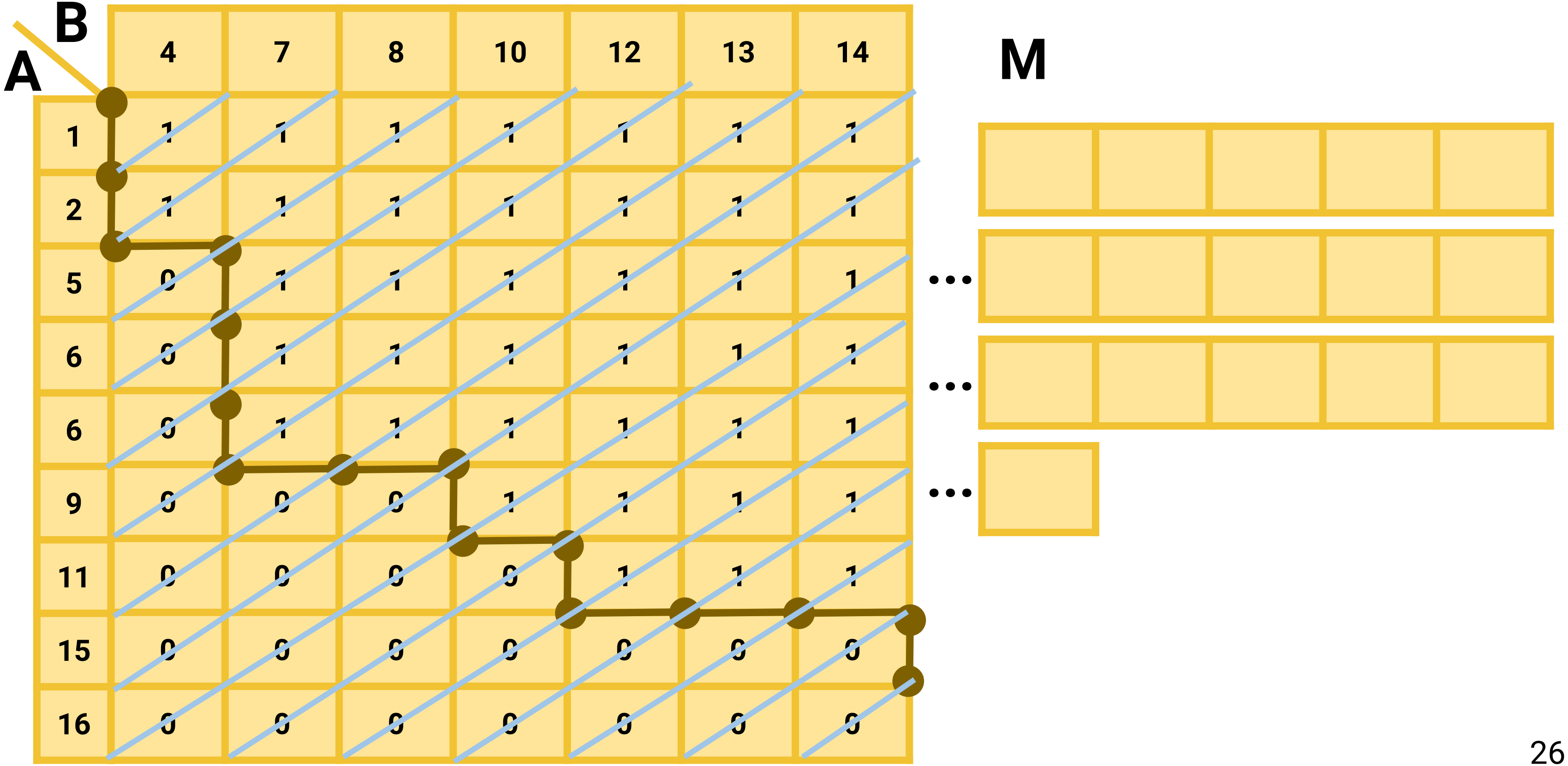
	1	2	4	5	6
...	6	7	8	9	10
...	11	12	13	14	15
...	16				

ALGO mergeSmall_k



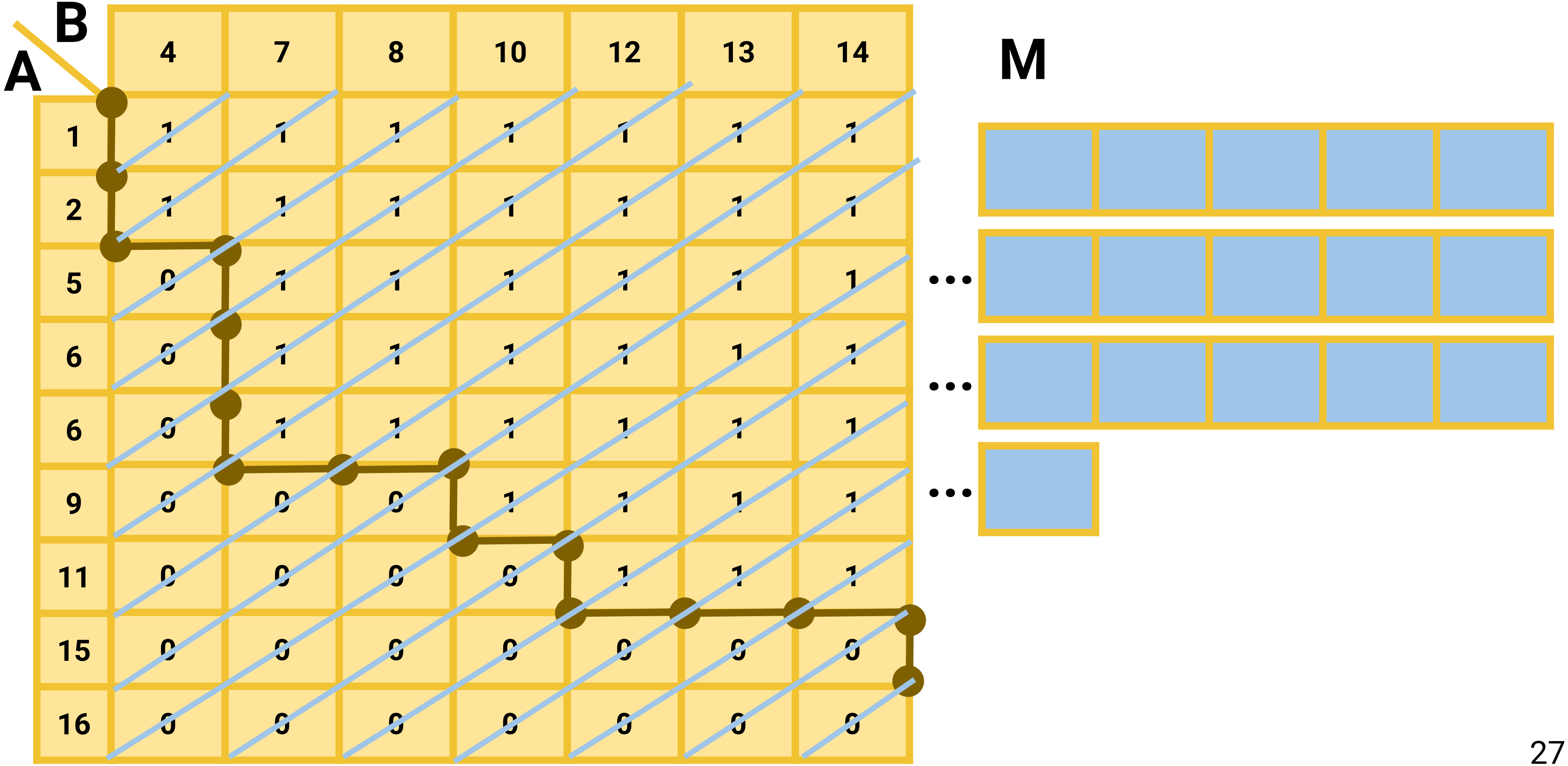
B A	4	7	8	10	12	13	14
1	1	1	1	1	1	1	1
2	1	1	1	1	1	1	1
5	0	1	1	1	1	1	1
6	0	1	1	1	1	1	1
6	0	1	1	1	1	1	1
9	0	0	0	1	1	1	1
11	0	0	0	0	1	1	1
15	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0

ALGO mergeSmall_k



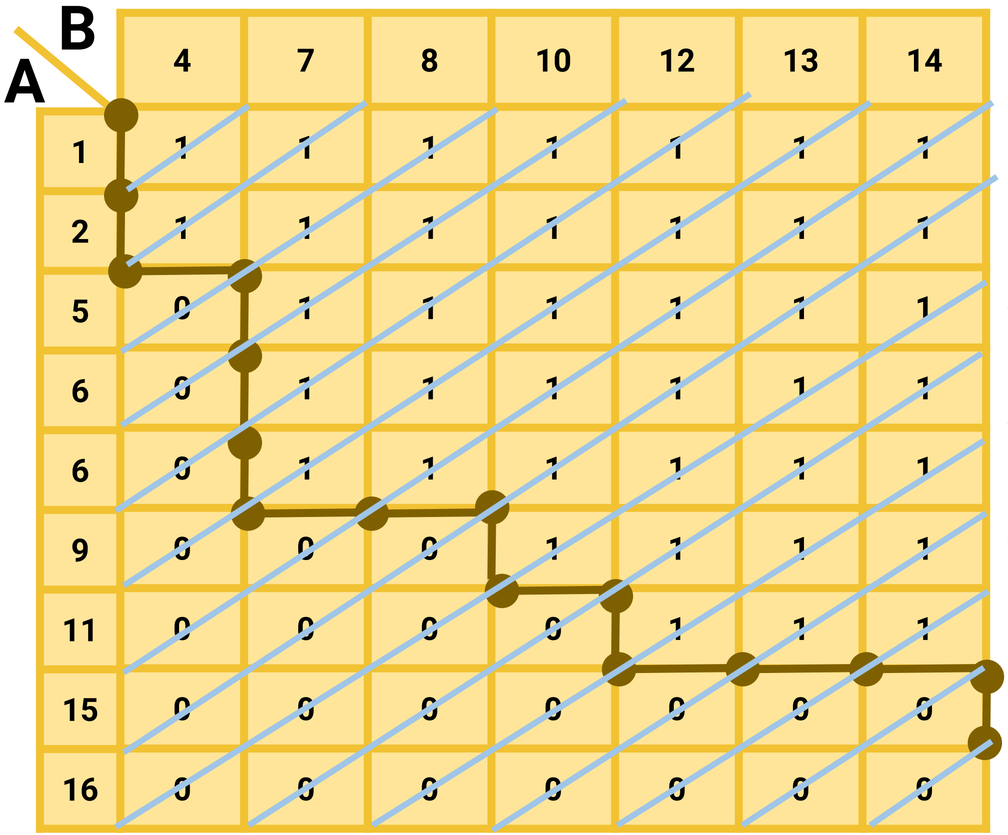
A \ B	4	7	8	10	12	13	14
1	1	1	1	1	1	1	1
2	1	1	1	1	1	1	1
5	0	1	1	1	1	1	1
6	0	1	1	1	1	1	1
6	0	1	1	1	1	1	1
9	0	0	0	1	1	1	1
11	0	0	0	0	1	1	1
15	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0

ALGO mergeSmall_k



A \ B	4	7	8	10	12	13	14
1	1	1	1	1	1	1	1
2	1	1	1	1	1	1	1
5	0	1	1	1	1	1	1
6	0	1	1	1	1	1	1
6	0	1	1	1	1	1	1
9	0	0	0	1	1	1	1
11	0	0	0	0	1	1	1
15	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0

ALGO mergeSmall_k

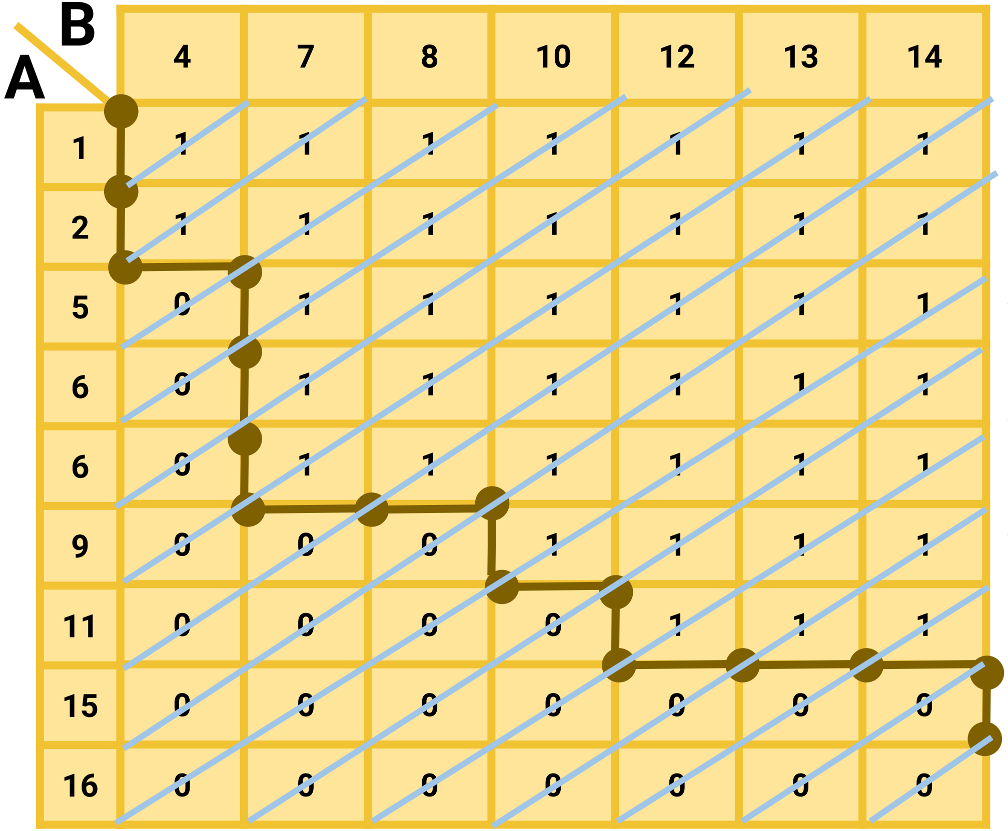


A \ B	4	7	8	10	12	13	14
1	1	1	1	1	1	1	1
2	1	1	1	1	1	1	1
5	0	1	1	1	1	1	1
6	0	1	1	1	1	1	1
6	0	1	1	1	1	1	1
9	0	0	0	1	1	1	1
11	0	0	0	0	1	1	1
15	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0

M

	1	2	4	5	6
…	6	7	8	9	10
…	11	12	13	14	15
…	16				

ALGO mergeSmall_k

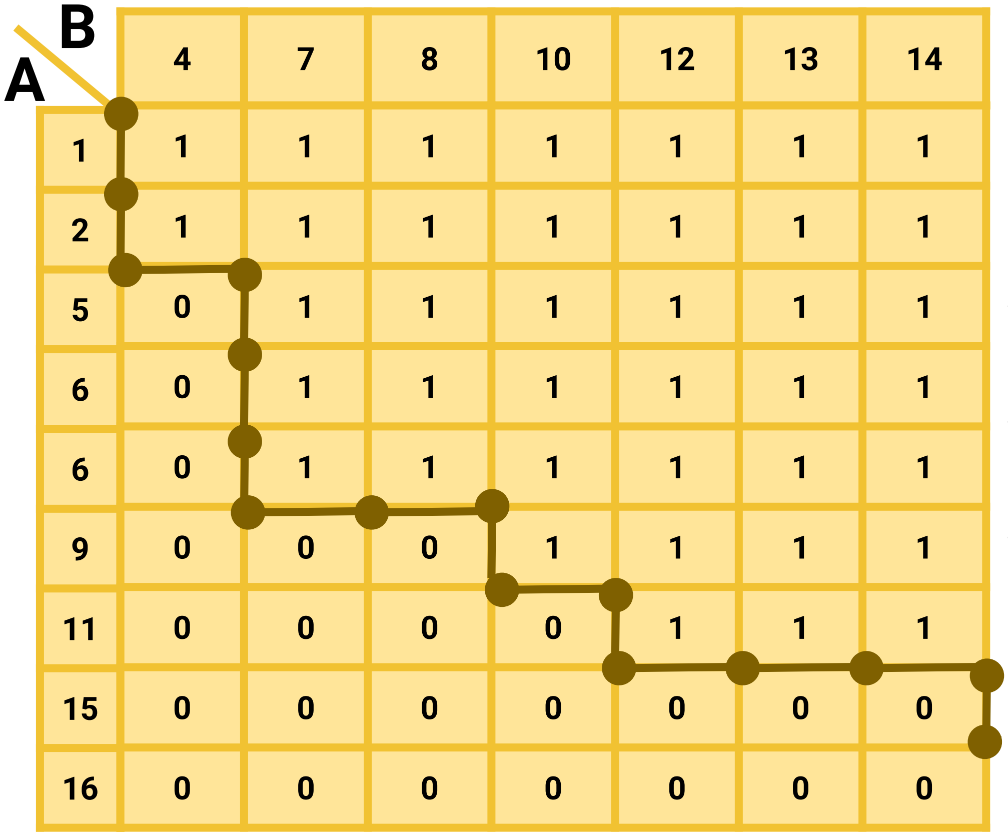


A \ B	4	7	8	10	12	13	14
1	1	1	1	1	1	1	1
2	1	1	1	1	1	1	1
5	0	1	1	1	1	1	1
6	0	1	1	1	1	1	1
6	0	1	1	1	1	1	1
9	0	0	0	1	1	1	1
11	0	0	0	0	1	1	1
15	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0

M

	1	2	4	5	6
…	6	7	8	9	10
…	11	12	13	14	15
…	16				

ALGO mergeSmall_k



A \ B	4	7	8	10	12	13	14
1	1	1	1	1	1	1	1
2	1	1	1	1	1	1	1
5	0	1	1	1	1	1	1
6	0	1	1	1	1	1	1
6	0	1	1	1	1	1	1
9	0	0	0	1	1	1	1
11	0	0	0	0	1	1	1
15	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0

M

	1	2	4	5	6
•••	6	7	8	9	10
•••	11	12	13	14	15
•••	16				

02

MERGE ON SEVERAL BLOCKS

ALGO pathBig_k

B A	4	7	8	10	12	13	14
1	1	1	1	1	1	1	1
2	1	1	1	1	1	1	1
5	0	1	1	1	1	1	1
6	0	1	1	1	1	1	1
6	0	1	1	1	1	1	1
9	0	0	0	1	1	1	1
11	0	0	0	0	1	1	1
15	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0

Path

1

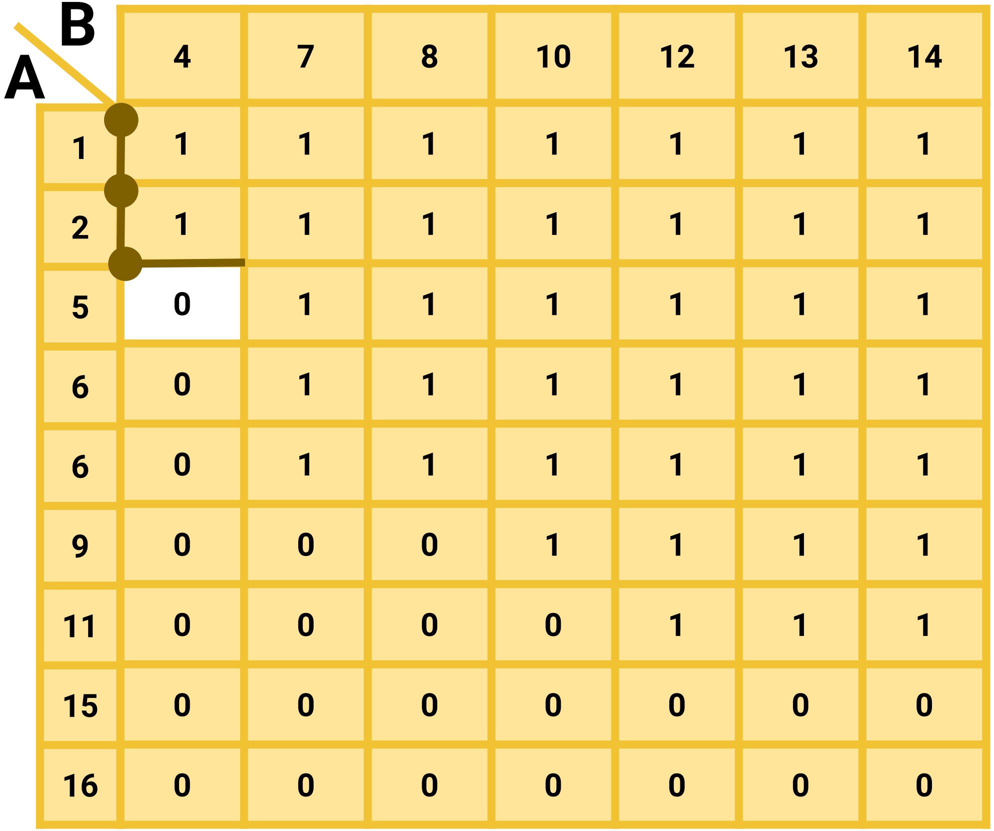
ALGO pathBig_k

B A	4	7	8	10	12	13	14
1	1	1	1	1	1	1	1
2	1	1	1	1	1	1	1
5	0	1	1	1	1	1	1
6	0	1	1	1	1	1	1
6	0	1	1	1	1	1	1
9	0	0	0	1	1	1	1
11	0	0	0	0	1	1	1
15	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0

Path

1	1

ALGO pathBig_k

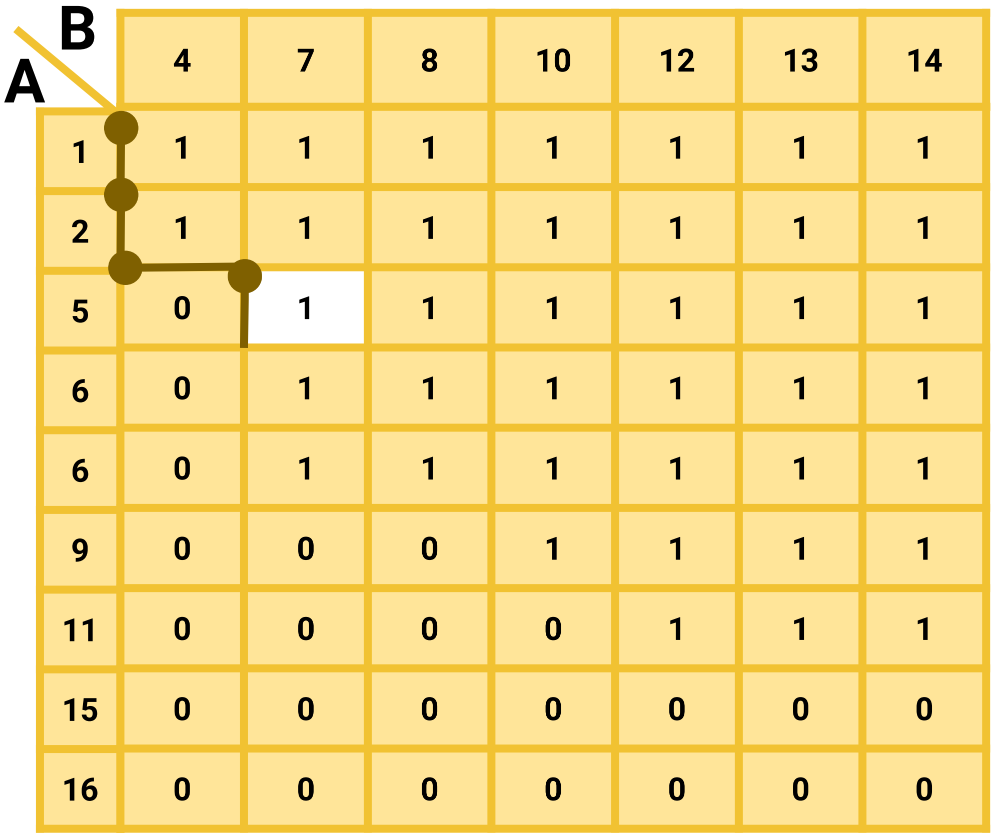


A \ B	4	7	8	10	12	13	14
1	1	1	1	1	1	1	1
2	1	1	1	1	1	1	1
5	0	1	1	1	1	1	1
6	0	1	1	1	1	1	1
6	0	1	1	1	1	1	1
9	0	0	0	1	1	1	1
11	0	0	0	0	1	1	1
15	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0

Path

1	1	0

ALGO pathBig_k

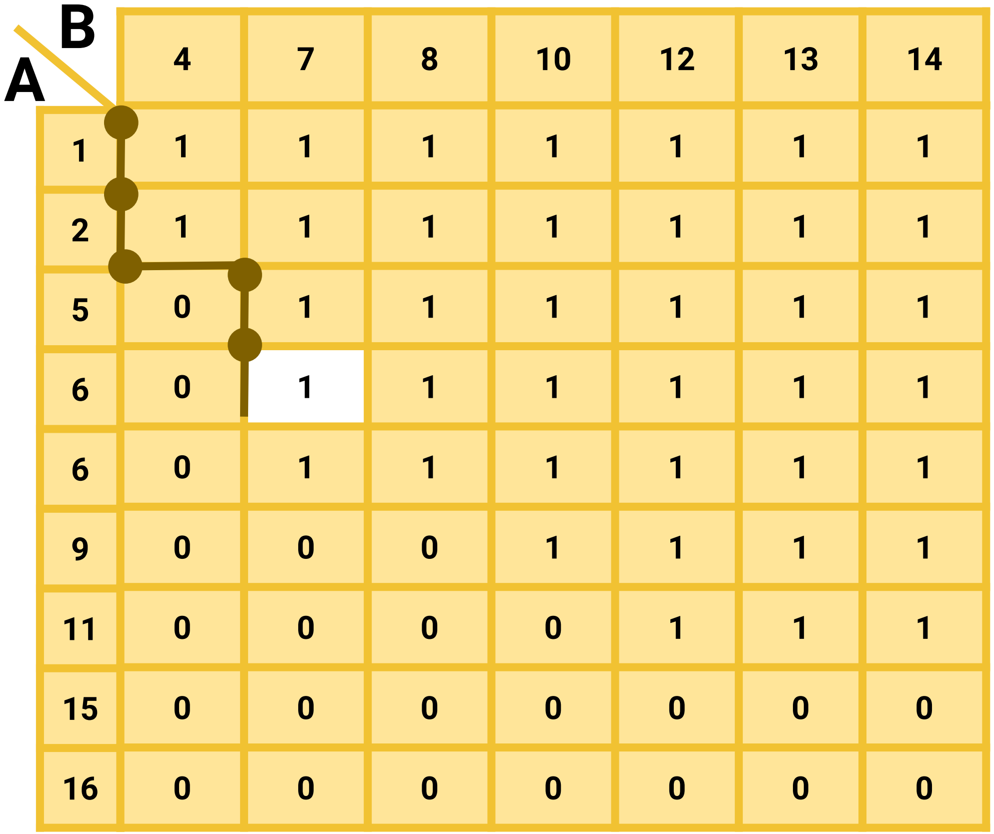


A \ B	4	7	8	10	12	13	14
1	1	1	1	1	1	1	1
2	1	1	1	1	1	1	1
5	0	1	1	1	1	1	1
6	0	1	1	1	1	1	1
6	0	1	1	1	1	1	1
9	0	0	0	1	1	1	1
11	0	0	0	0	1	1	1
15	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0

Path

1	1	0	1

ALGO pathBig_k

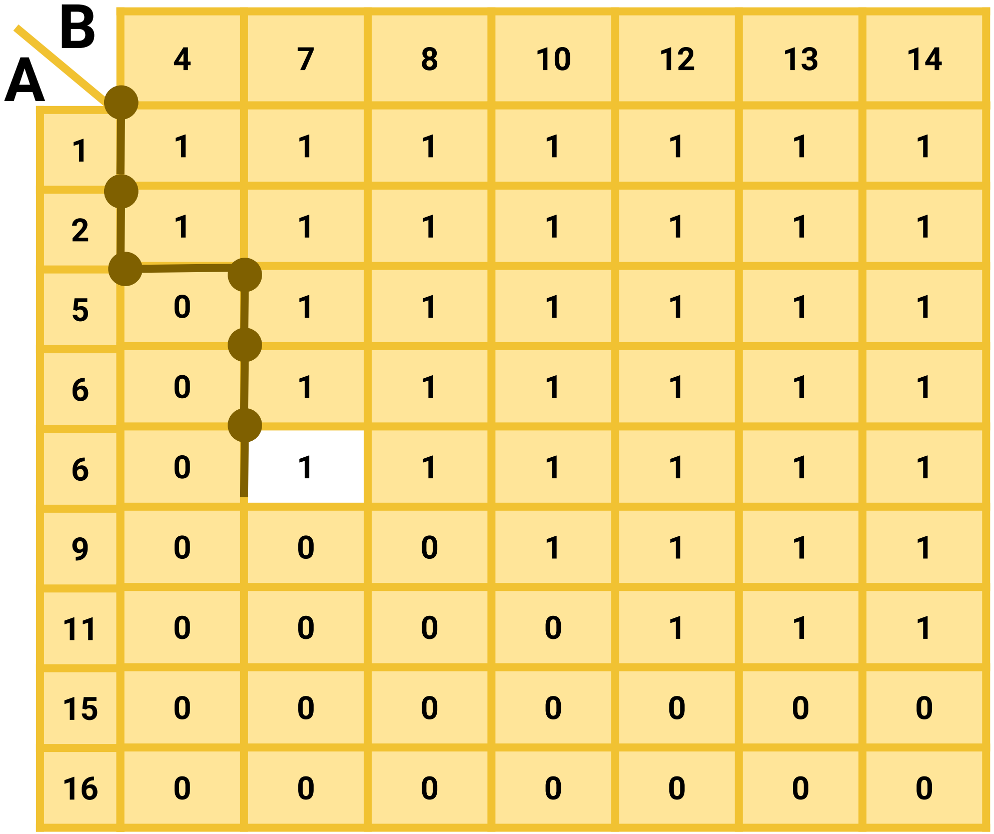


A \ B	4	7	8	10	12	13	14
1	1	1	1	1	1	1	1
2	1	1	1	1	1	1	1
5	0	1	1	1	1	1	1
6	0	1	1	1	1	1	1
6	0	1	1	1	1	1	1
9	0	0	0	1	1	1	1
11	0	0	0	0	1	1	1
15	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0

Path

1	1	0	1	1

ALGO pathBig_k



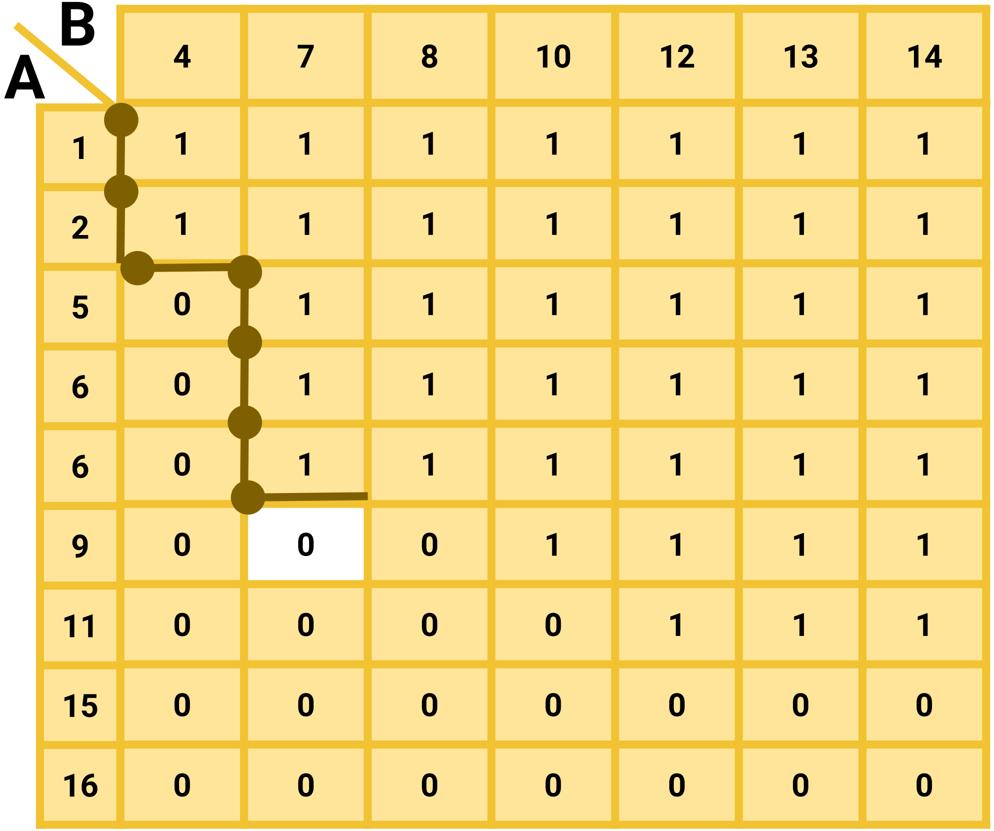
A \ B	4	7	8	10	12	13	14
1	1	1	1	1	1	1	1
2	1	1	1	1	1	1	1
5	0	1	1	1	1	1	1
6	0	1	1	1	1	1	1
6	0	1	1	1	1	1	1
9	0	0	0	1	1	1	1
11	0	0	0	0	1	1	1
15	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0

Path

1	1	0	1	1

… 1

ALGO pathBig_k



A \ B	4	7	8	10	12	13	14
1	1	1	1	1	1	1	1
2	1	1	1	1	1	1	1
5	0	1	1	1	1	1	1
6	0	1	1	1	1	1	1
6	0	1	1	1	1	1	1
9	0	0	0	1	1	1	1
11	0	0	0	0	1	1	1
15	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0

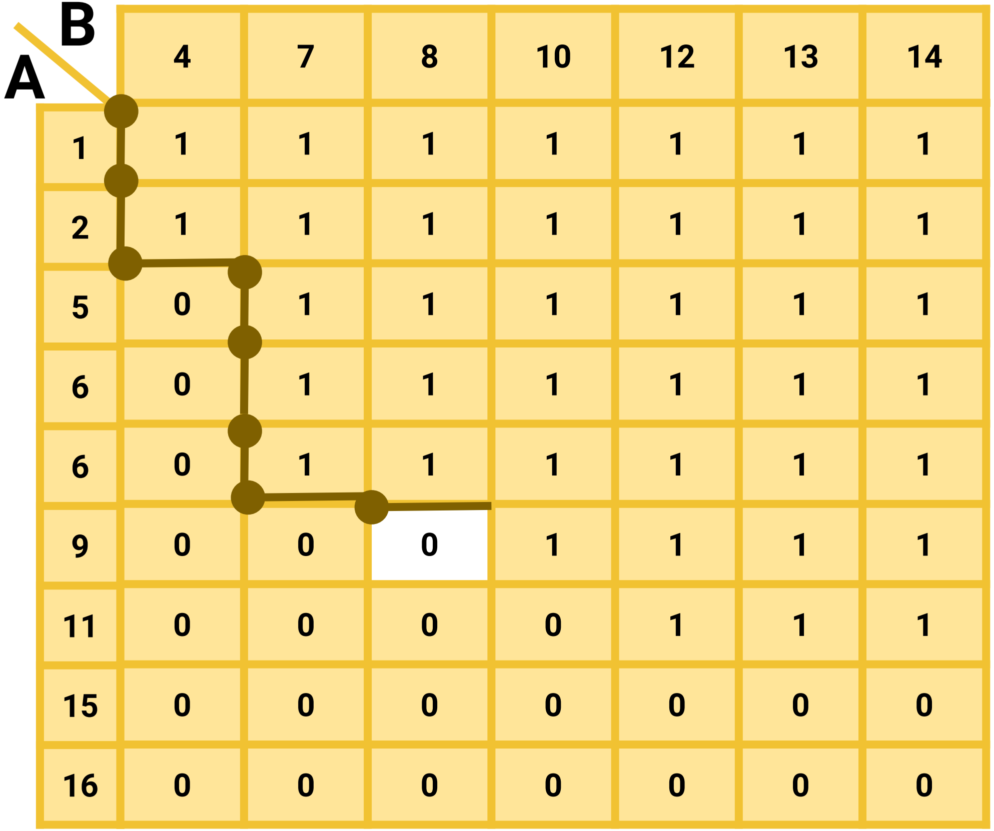
Path

1	1	0	1	1

…

1	0

ALGO pathBig_k



A B	4	7	8	10	12	13	14
1	1	1	1	1	1	1	1
2	1	1	1	1	1	1	1
5	0	1	1	1	1	1	1
6	0	1	1	1	1	1	1
6	0	1	1	1	1	1	1
9	0	0	0	1	1	1	1
11	0	0	0	0	1	1	1
15	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0

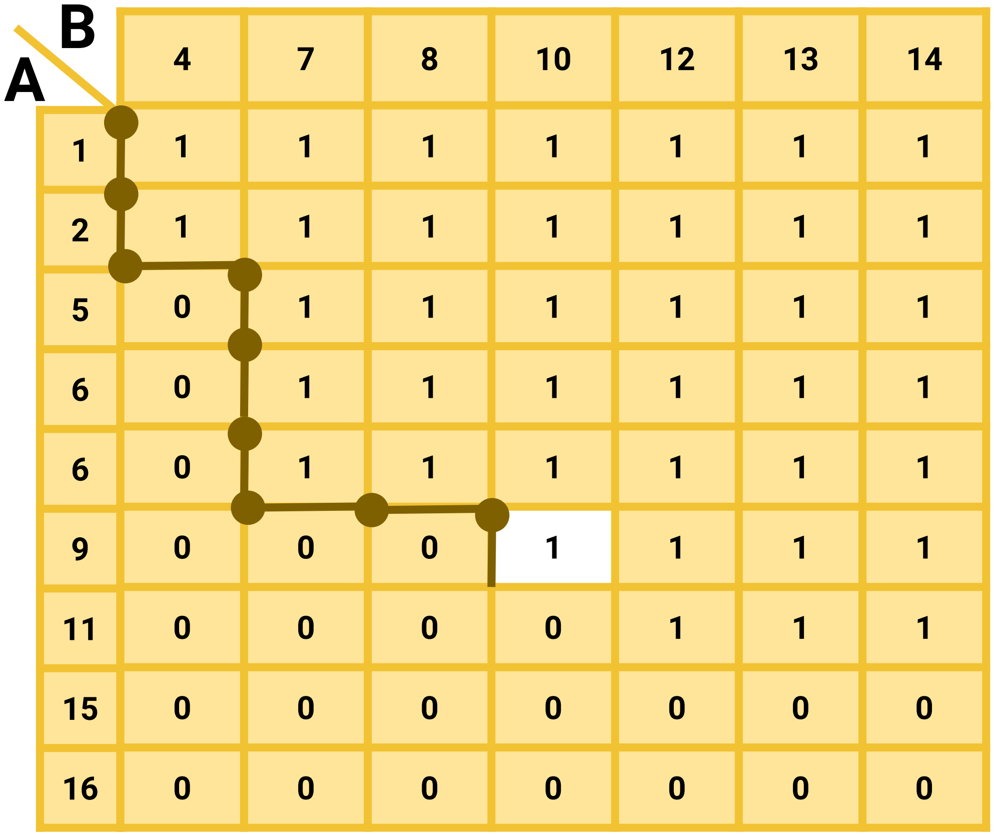
Path

1	1	0	1	1

…

1	0	0

ALGO pathBig_k



A \ B	4	7	8	10	12	13	14
1	1	1	1	1	1	1	1
2	1	1	1	1	1	1	1
5	0	1	1	1	1	1	1
6	0	1	1	1	1	1	1
6	0	1	1	1	1	1	1
9	0	0	0	1	1	1	1
11	0	0	0	0	1	1	1
15	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0

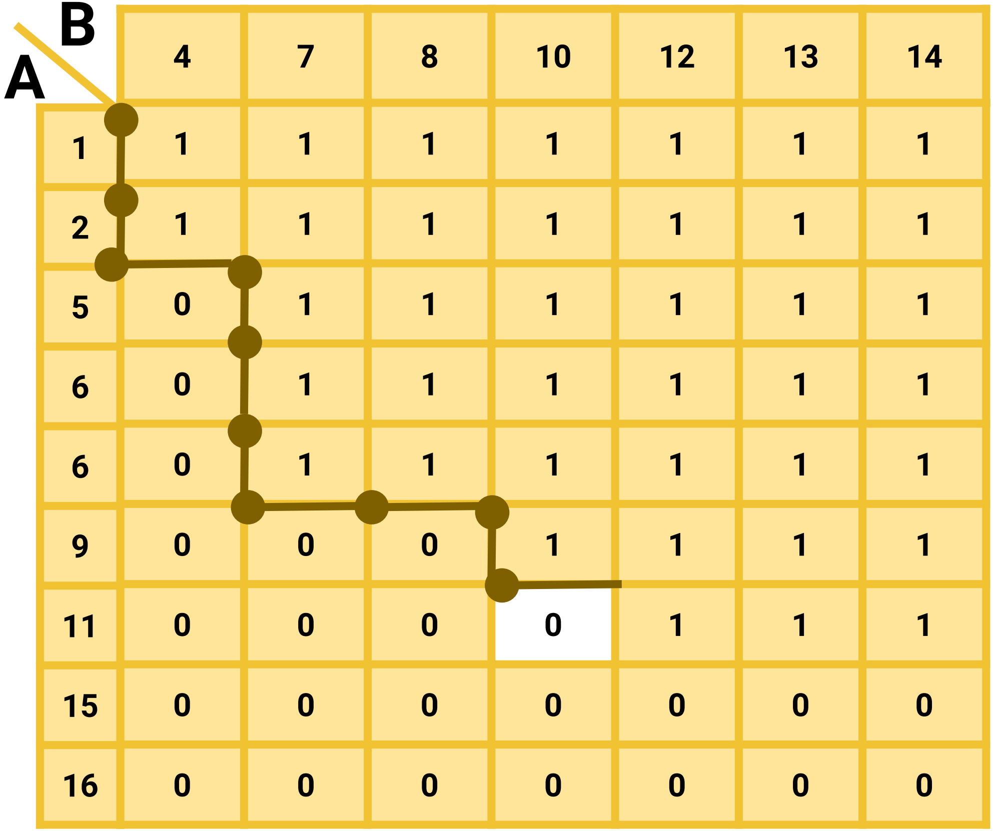
Path

1	1	0	1	1

…

1	0	0	1

ALGO pathBig_k



A \ B	4	7	8	10	12	13	14
1	1	1	1	1	1	1	1
2	1	1	1	1	1	1	1
5	0	1	1	1	1	1	1
6	0	1	1	1	1	1	1
6	0	1	1	1	1	1	1
9	0	0	0	1	1	1	1
11	0	0	0	0	1	1	1
15	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0

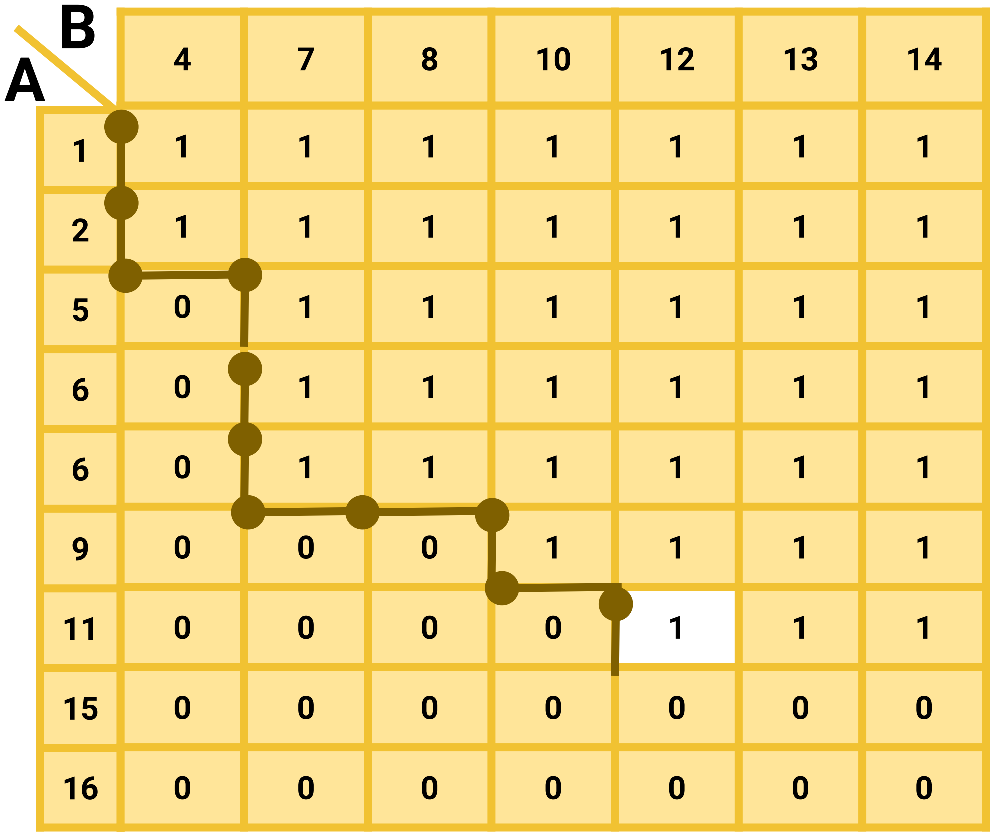
Path

1	1	0	1	1

...

1	0	0	1	0

ALGO pathBig_k



A \ B	4	7	8	10	12	13	14
1	1	1	1	1	1	1	1
2	1	1	1	1	1	1	1
5	0	1	1	1	1	1	1
6	0	1	1	1	1	1	1
6	0	1	1	1	1	1	1
9	0	0	0	1	1	1	1
11	0	0	0	0	1	1	1
15	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0

Path

1	1	0	1	1

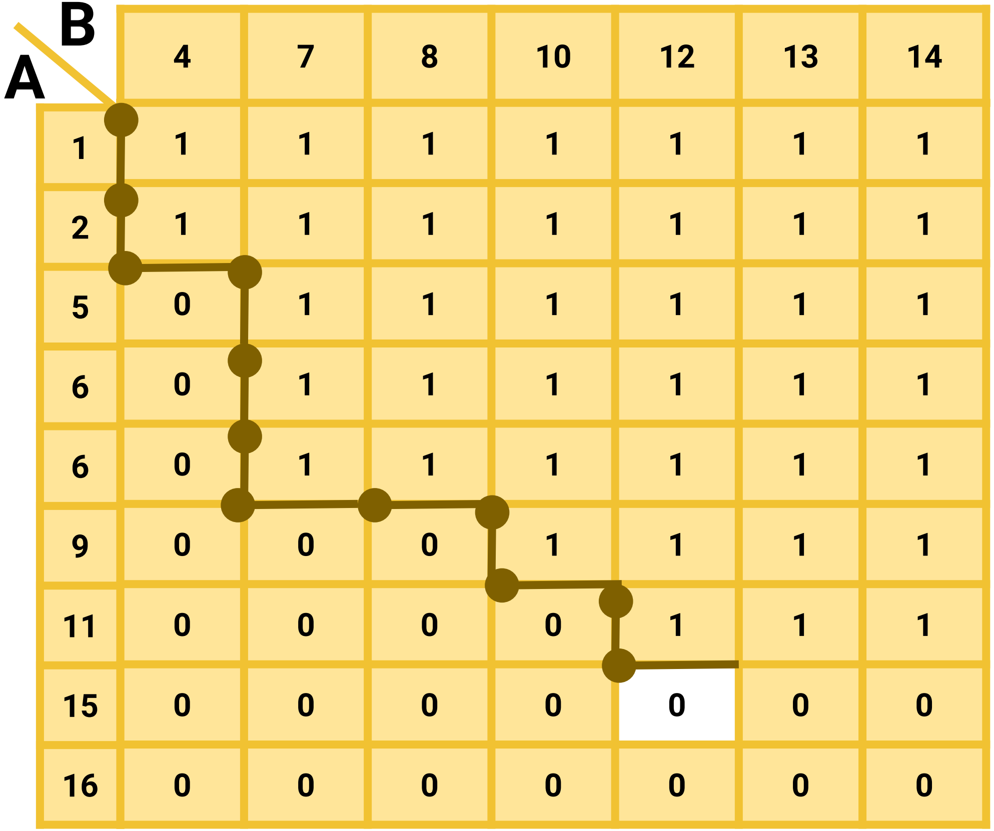
…

1	0	0	1	0

…

1

ALGO pathBig_k

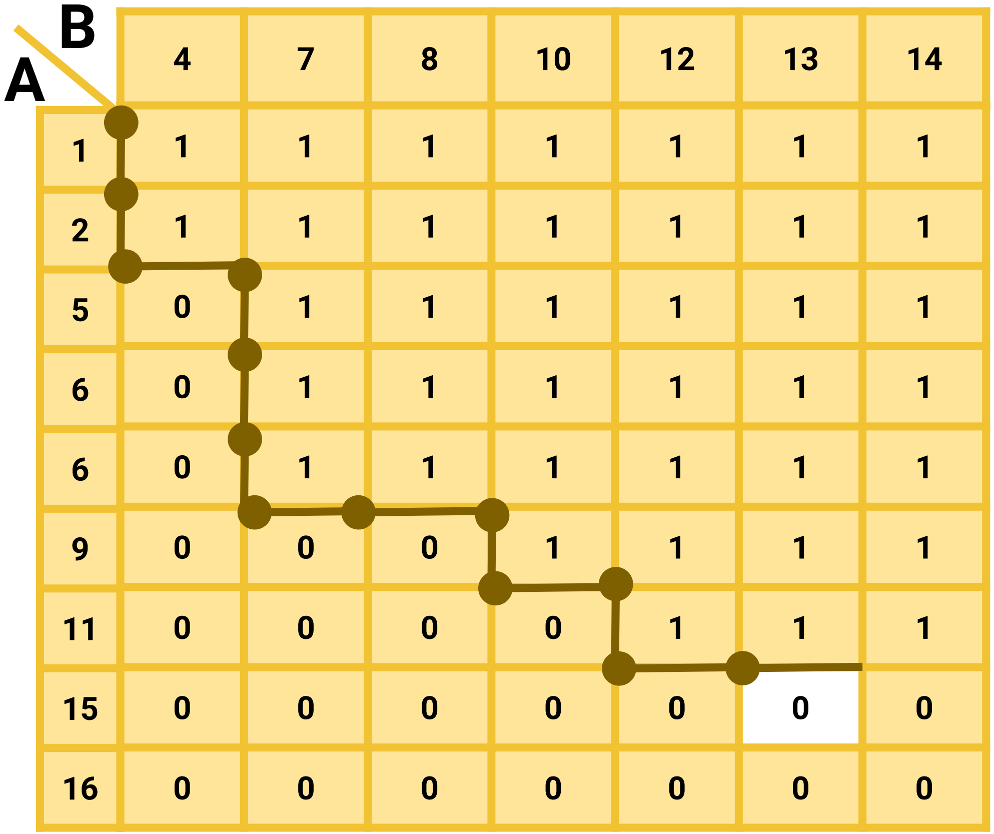


A \ B	4	7	8	10	12	13	14
1	1	1	1	1	1	1	1
2	1	1	1	1	1	1	1
5	0	1	1	1	1	1	1
6	0	1	1	1	1	1	1
6	0	1	1	1	1	1	1
9	0	0	0	1	1	1	1
11	0	0	0	0	1	1	1
15	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0

Path

	1	1	0	1	1
…	1	0	0	1	0
…	1	0			

ALGO pathBig_k



A \ B	4	7	8	10	12	13	14
1	1	1	1	1	1	1	1
2	1	1	1	1	1	1	1
5	0	1	1	1	1	1	1
6	0	1	1	1	1	1	1
6	0	1	1	1	1	1	1
9	0	0	0	1	1	1	1
11	0	0	0	0	1	1	1
15	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0

Path

1	1	0	1	1

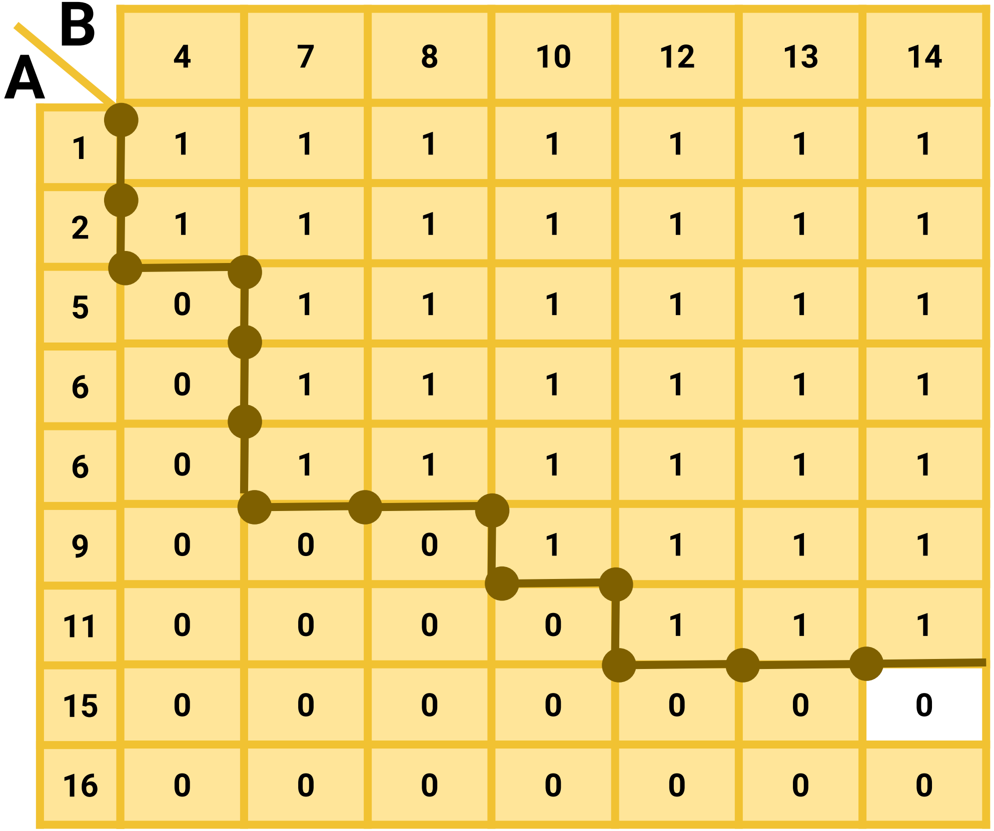
...

1	0	0	1	0

...

1	0	0

ALGO pathBig_k



A \ B	4	7	8	10	12	13	14
1	1	1	1	1	1	1	1
2	1	1	1	1	1	1	1
5	0	1	1	1	1	1	1
6	0	1	1	1	1	1	1
6	0	1	1	1	1	1	1
9	0	0	0	1	1	1	1
11	0	0	0	0	1	1	1
15	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0

Path

1	1	0	1	1

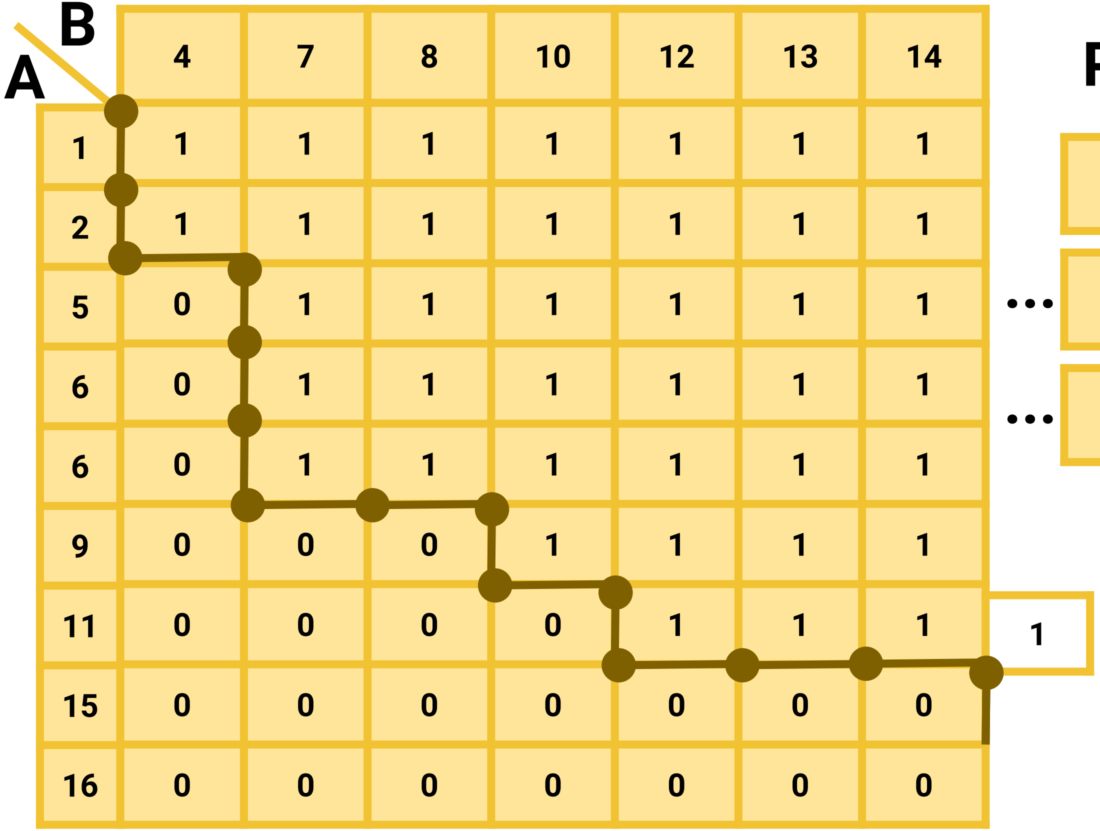
...

1	0	0	1	0

...

1	0	0	0

ALGO mergeSmall_k

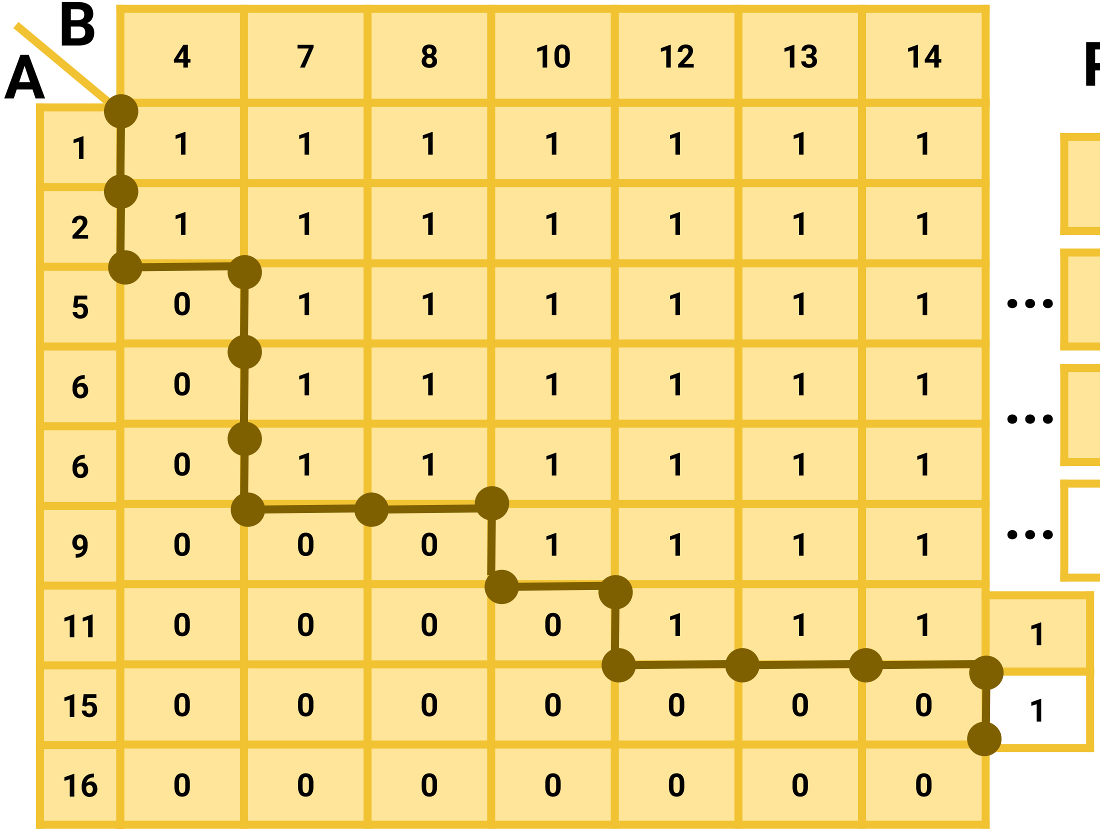


B A	4	7	8	10	12	13	14
1	1	1	1	1	1	1	1
2	1	1	1	1	1	1	1
5	0	1	1	1	1	1	1
6	0	1	1	1	1	1	1
6	0	1	1	1	1	1	1
9	0	0	0	1	1	1	1
11	0	0	0	0	1	1	1
15	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0

Path

	1	1	0	1	1
...	1	0	0	1	0
...	1	0	0	0	1

ALGO pathBig_k

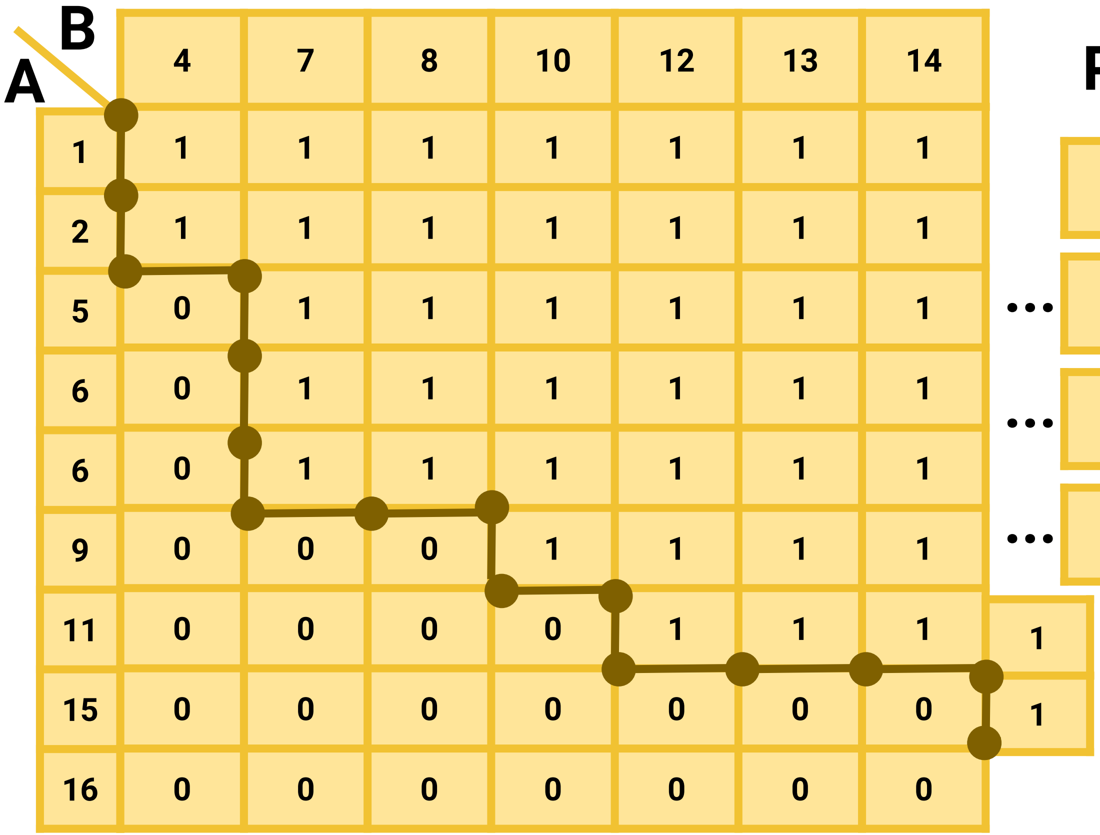


A \ B	4	7	8	10	12	13	14	
1	1	1	1	1	1	1	1	
2	1	1	1	1	1	1	1	
5	0	1	1	1	1	1	1	
6	0	1	1	1	1	1	1	
6	0	1	1	1	1	1	1	
9	0	0	0	1	1	1	1	
11	0	0	0	0	1	1	1	1
15	0	0	0	0	0	0	0	1
16	0	0	0	0	0	0	0	

Path

	1	1	0	1	1
…	1	0	0	1	0
…	1	0	0	0	1
…	1				

ALGO pathBig_k

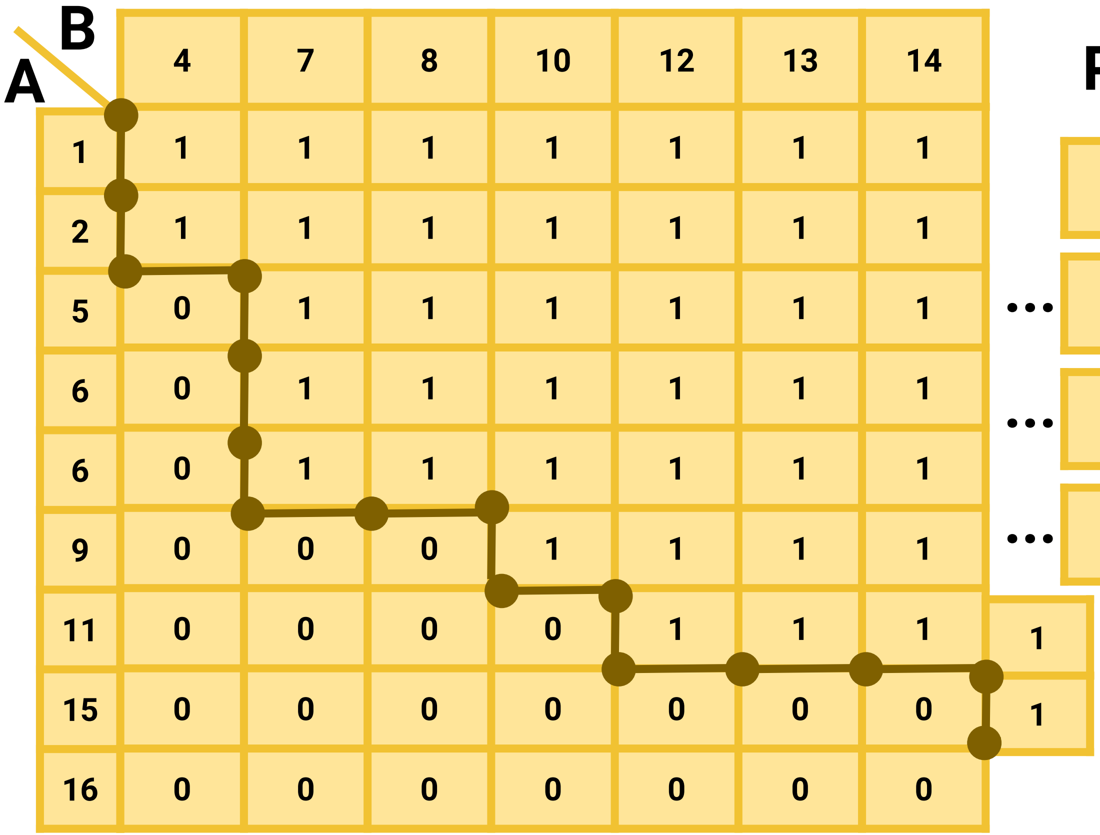


A \ B	4	7	8	10	12	13	14	
1	1	1	1	1	1	1	1	
2	1	1	1	1	1	1	1	
5	0	1	1	1	1	1	1	
6	0	1	1	1	1	1	1	
6	0	1	1	1	1	1	1	
9	0	0	0	1	1	1	1	
11	0	0	0	0	1	1	1	1
15	0	0	0	0	0	0	0	1
16	0	0	0	0	0	0	0	

Path

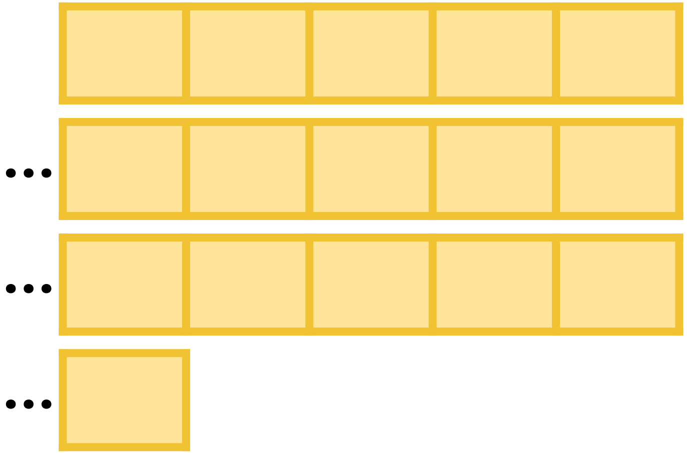
	1	1	0	1	1
…	1	0	0	1	0
…	1	0	0	0	1
…	1				

ALGO pathBig_k

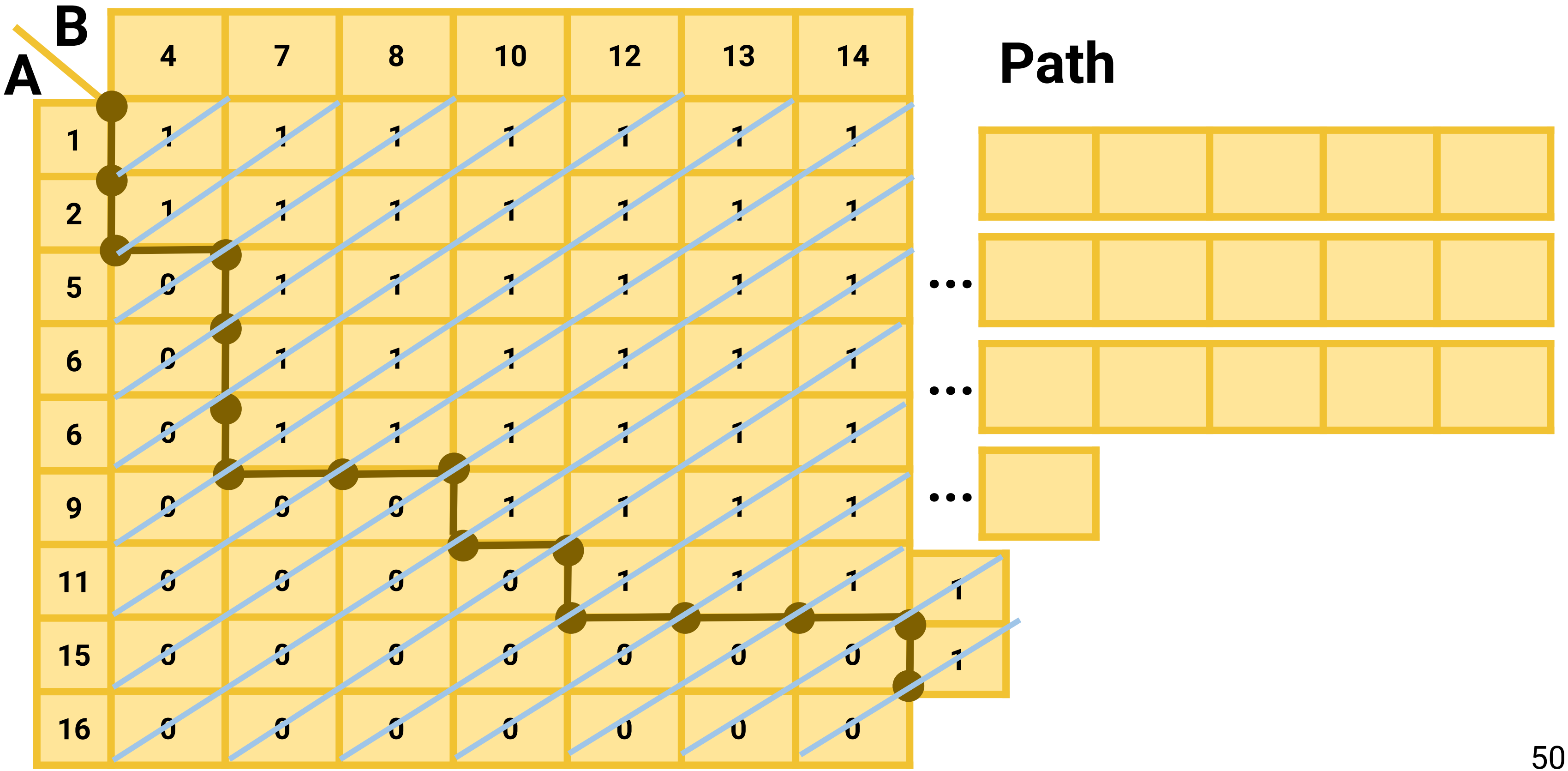


A \ B	4	7	8	10	12	13	14	
1	1	1	1	1	1	1	1	
2	1	1	1	1	1	1	1	
5	0	1	1	1	1	1	1	
6	0	1	1	1	1	1	1	
6	0	1	1	1	1	1	1	
9	0	0	0	1	1	1	1	
11	0	0	0	0	1	1	1	1
15	0	0	0	0	0	0	0	1
16	0	0	0	0	0	0	0	

Path

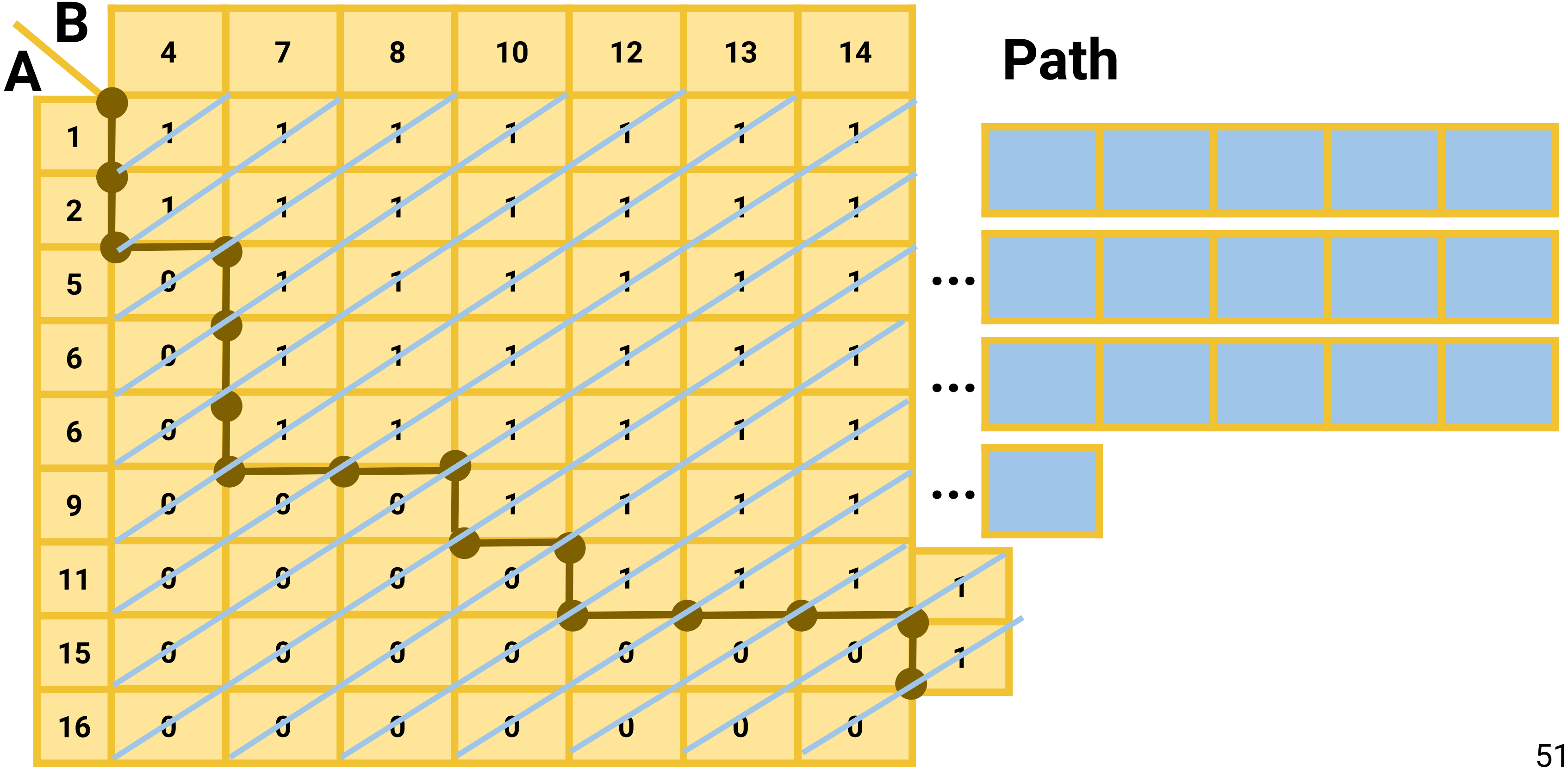


ALGO pathBig_k



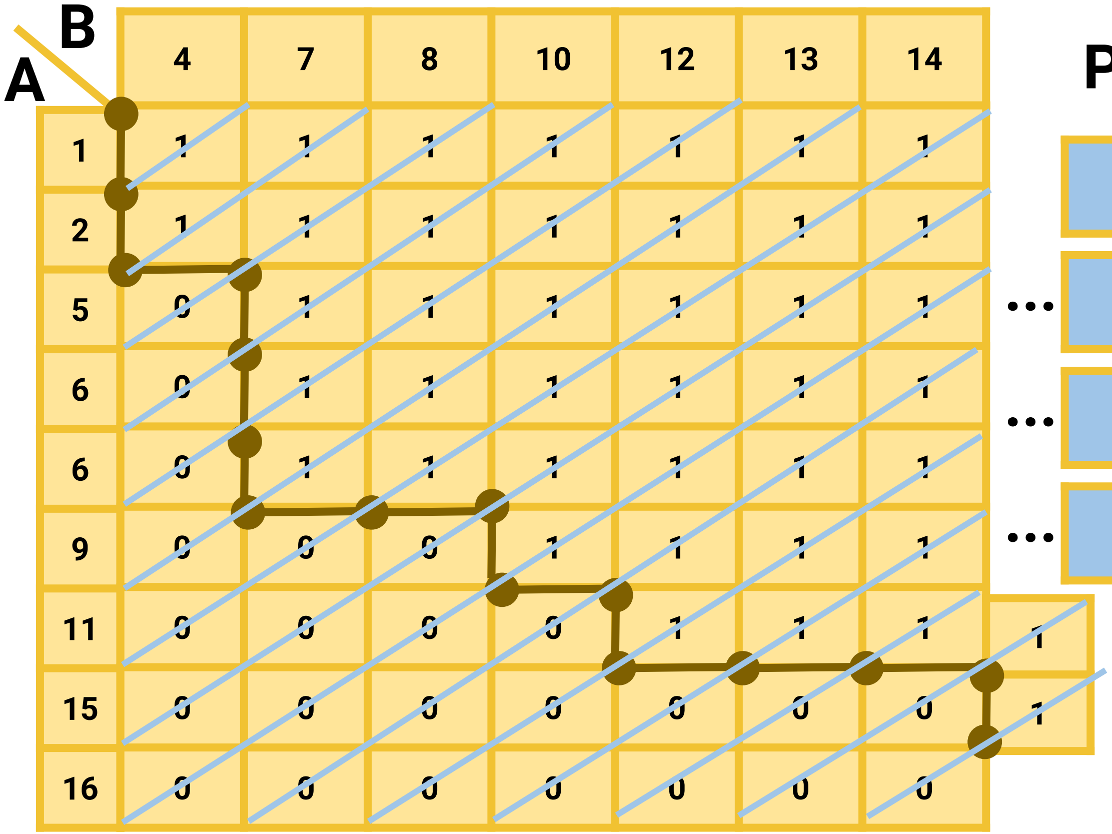
B A	4	7	8	10	12	13	14	
1	1	1	1	1	1	1	1	
2	1	1	1	1	1	1	1	
5	0	1	1	1	1	1	1	
6	0	1	1	1	1	1	1	
6	0	1	1	1	1	1	1	
9	0	0	0	1	1	1	1	
11	0	0	0	0	1	1	1	1
15	0	0	0	0	0	0	0	1
16	0	0	0	0	0	0	0	

ALGO pathBig_k



A \ B	4	7	8	10	12	13	14	
1	1	1	1	1	1	1	1	
2	1	1	1	1	1	1	1	
5	0	1	1	1	1	1	1	
6	0	1	1	1	1	1	1	
6	0	1	1	1	1	1	1	
9	0	0	0	1	1	1	1	
11	0	0	0	0	1	1	1	1
15	0	0	0	0	0	0	0	1
16	0	0	0	0	0	0	0	

ALGO pathBig_k

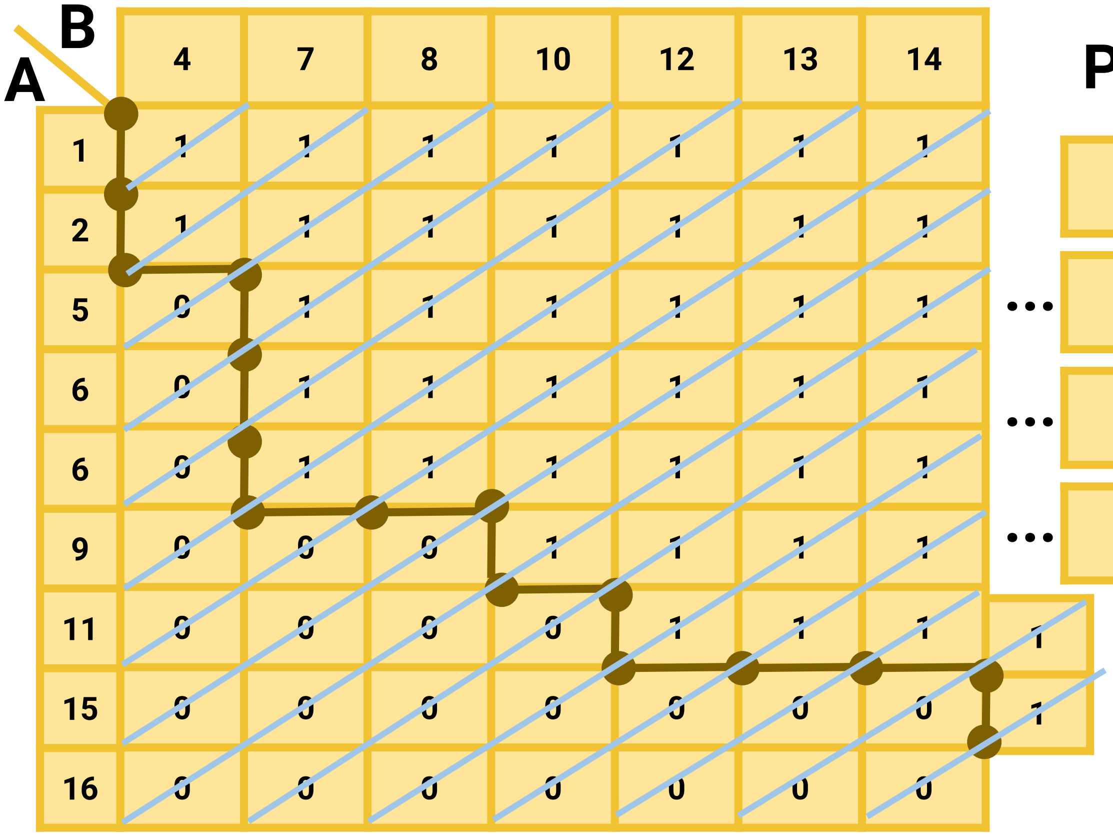


A \ B	4	7	8	10	12	13	14	
1	1	1	1	1	1	1	1	
2	1	1	1	1	1	1	1	
5	0	1	1	1	1	1	1	
6	0	1	1	1	1	1	1	
6	0	1	1	1	1	1	1	
9	0	0	0	1	1	1	1	
11	0	0	0	0	1	1	1	1
15	0	0	0	0	0	0	0	1
16	0	0	0	0	0	0	0	

Path

	1	1	0	1	1
…	1	0	0	1	0
…	1	0	0	0	1
…	1				

ALGO pathBig_k

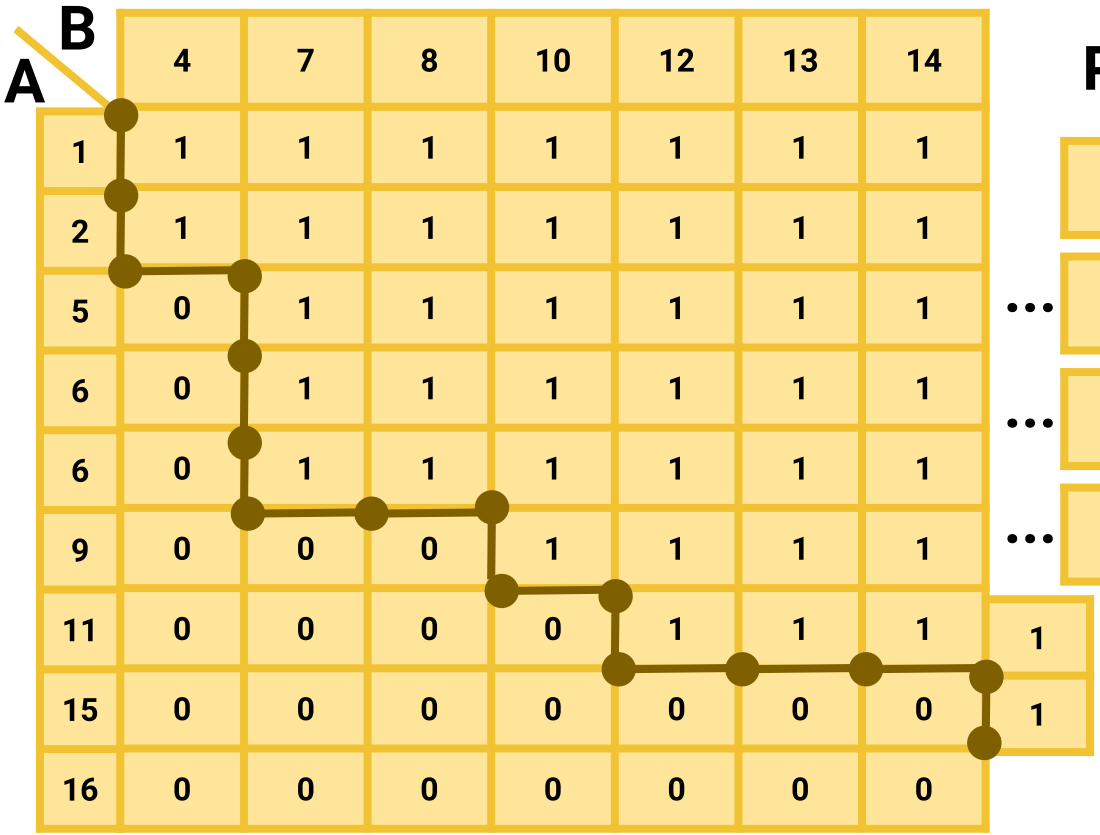


A \ B	4	7	8	10	12	13	14	
1	1	1	1	1	1	1	1	
2	1	1	1	1	1	1	1	
5	0	1	1	1	1	1	1	
6	0	1	1	1	1	1	1	
6	0	1	1	1	1	1	1	
9	0	0	0	1	1	1	1	
11	0	0	0	0	1	1	1	1
15	0	0	0	0	0	0	0	1
16	0	0	0	0	0	0	0	

Path

	1	1	0	1	1
…	1	0	0	1	0
…	1	0	0	0	1
…	1				

ALGO pathBig_k



B A	4	7	8	10	12	13	14	
1	1	1	1	1	1	1	1	
2	1	1	1	1	1	1	1	
5	0	1	1	1	1	1	1	
6	0	1	1	1	1	1	1	
6	0	1	1	1	1	1	1	
9	0	0	0	1	1	1	1	
11	0	0	0	0	1	1	1	1
15	0	0	0	0	0	0	0	1
16	0	0	0	0	0	0	0	

Path

	1	1	0	1	1
…	1	0	0	1	0
…	1	0	0	0	1
…	1				

ALGO mergeBig_k

A

1	2	5	6	6	9	11	15	16

B

4	7	8	10	12	13	14

Path

1	1	0	1	1	1	0	0	1	0	1	0	0	0	1	1

ALGO mergeBig_k

A

1	2	5	6	6	9	11	15	16

B

4	7	8	10	12	13	14

Path

1	1	0	1	1	1	0	0	1	0	1	0	0	0	1	1

M

1

ALGO mergeBig_k

A

1	2	5	6	6	9	11	15	16

B

4	7	8	10	12	13	14

Path

1	1	0	1	1	1	0	0	1	0	1	0	0	0	1	1

M

1	2

ALGO mergeBig_k

A

1	2	5	6	6	9	11	15	16

B

4	7	8	10	12	13	14

Path

1	1	0	1	1	1	0	0	1	0	1	0	0	0	1	1

M

1	2	4

ALGO mergeBig_k

A

1	2	5	6	6	9	11	15	16

B

4	7	8	10	12	13	14

Path

1	1	0	1	1	1	0	0	1	0	1	0	0	0	1	1

M

1	2	4	5

ALGO mergeBig_k

A

1	2	5	6	6	9	11	15	16

B

4	7	8	10	12	13	14

Path

1	1	0	1	1	1	0	0	1	0	1	0	0	0	1	1

M

1	2	4	5	6

ALGO mergeBig_k

A

1	2	5	6	6	9	11	15	16

B

4	7	8	10	12	13	14

Path

1	1	0	1	1	1	0	0	1	0	1	0	0	0	1	1

M

1	2	4	5	6	6

ALGO mergeBig_k

A

1	2	5	6	6	9	11	15	16

B

4	7	8	10	12	13	14

Path

1	1	0	1	1	1	0	0	1	0	1	0	0	0	1	1

M

1	2	4	5	6	6	7

ALGO mergeBig_k

A

1	2	5	6	6	9	11	15	16

B

4	7	8	10	12	13	14

Path

1	1	0	1	1	1	0	0	1	0	1	0	0	0	1	1

M

1	2	4	5	6	6	7	8

ALGO mergeBig_k

A

1	2	5	6	6	9	11	15	16

B

4	7	8	10	12	13	14

Path

1	1	0	1	1	1	0	0	1	0	1	0	0	0	1	1

M

1	2	4	5	6	6	7	8	9

ALGO mergeBig_k

A

1	2	5	6	6	9	11	15	16

B

4	7	8	10	12	13	14

Path

1	1	0	1	1	1	0	0	1	0	1	0	0	0	1	1

M

1	2	4	5	6	6	7	8	9	10

ALGO mergeBig_k

A

1	2	5	6	6	9	11	15	16

B

4	7	8	10	12	13	14

Path

1	1	0	1	1	1	0	0	1	0	1	0	0	0	1	1

M

1	2	4	5	6	6	7	8	9	10	11

ALGO mergeBig_k

A

1	2	5	6	6	9	11	15	16

B

4	7	8	10	12	13	14

Path

1	1	0	1	1	1	0	0	1	0	1	0	0	0	1	1

M

1	2	4	5	6	6	7	8	9	10	11	12

ALGO mergeBig_k

A

1	2	5	6	6	9	11	15	16

B

4	7	8	10	12	13	14

Path

1	1	0	1	1	1	0	0	1	0	1	0	0	0	1	1

M

1	2	4	5	6	6	7	8	9	10	11	12	13

ALGO mergeBig_k

A

1	2	5	6	6	9	11	15	16

B

4	7	8	10	12	13	14

Path

1	1	0	1	1	1	0	0	1	0	1	0	0	0	1	1

M

1	2	4	5	6	6	7	8	9	10	11	12	13	14

ALGO mergeBig_k

A

1	2	5	6	6	9	11	15	16

B

4	7	8	10	12	13	14

Path

1	1	0	1	1	1	0	0	1	0	1	0	0	0	1	1

M

1	2	4	5	6	6	7	8	9	10	11	12	13	14	15

ALGO mergeBig_k

A

1	2	5	6	6	9	11	15	16

B

4	7	8	10	12	13	14

Path

1	1	0	1	1	1	0	0	1	0	1	0	0	0	1	1

M

1	2	4	5	6	6	7	8	9	10	11	12	13	14	15	16

ALGO mergeBig_k

A

1	2	5	6	6	9	11	15	16

B

4	7	8	10	12	13	14

Path

1	1	0	1	1	1	0	0	1	0	1	0	0	0	1	1

M

1	2	4	5	6	6	7	8	9	10	11	12	13	14	15	16

ALGO mergeBig_k

A

1	2	5	6	6	9	11	15	16

B

4	7	8	10	12	13	14

Path

1	1	0	1	1	1	0	0	1	0	1	0	0	0	1	1

M

1	2	4	5	6	6	7	8	9	10	11	12	13	14	15	16

ALGO mergeBig_k

A

1	2	5	6	6	9	11	15	16

B

4	7	8	10	12	13	14

Path

1	1	0	1	1	1	0	0	1	0	1	0	0	0	1	1

ALGO mergeBig_k

A

1	2	5	6	6	9	11	15	16

B

4	7	8	10	12	13	14

Path

1	1	0	1	1	1	0	0	1	0	1	0	0	0	1	1
0	1	0	2	4	5	1	2	6	3	7	4	5	6	8	9

ALGO mergeBig_k

A

1	2	5	6	6	9	11	15	16

B

4	7	8	10	12	13	14

Path

1	1	0	1	1	1	0	0	1	0	1	0	0	0	1	1
0	1	0	2	4	5	1	2	6	3	7	4	5	6	8	9

M

ALGO mergeBig_k

A

1	2	5	6	6	9	11	15	16

B

4	7	8	10	12	13	14

Path

1	1	0	1	1	1	0	0	1	0	1	0	0	0	1	1
0	1	0	2	4	5	1	2	6	3	7	4	5	6	8	9

M

ALGO mergeBig_k

A

1	2	5	6	6	9	11	15	16

B

4	7	8	10	12	13	14

Path

1	1	0	1	1	1	0	0	1	0	1	0	0	0	1	1
0	1	0	2	4	5	1	2	6	3	7	4	5	6	8	9

M

ALGO mergeBig_k

A

1	2	5	6	6	9	11	15	16

B

4	7	8	10	12	13	14

Path

1	1	0	1	1	1	0	0	1	0	1	0	0	0	1	1
0	1	0	2	4	5	1	2	6	3	7	4	5	6	8	9

M

ALGO mergeBig_k

A

1	2	5	6	6	9	11	15	16

B

4	7	8	10	12	13	14

Path

1	1	0	1	1	1	0	0	1	0	1	0	0	0	1	1
0	1	0	2	4	5	1	2	6	3	7	4	5	6	8	9

M

ALGO mergeBig_k

A

1	2	5	6	6	9	11	15	16

B

4	7	8	10	12	13	14

Path

1	1	0	1	1	1	0	0	1	0	1	0	0	0	1	1
0	1	0	2	4	5	1	2	6	3	7	4	5	6	8	9

M

1	2	4	5	6	6	7	8	9	10	11	12	13	14	15	16

ALGO mergeBig_k

A

1	2	5	6	6	9	11	15	16

B

4	7	8	10	12	13	14

Path

1	1	0	1	1	1	0	0	1	0	1	0	0	0	1	1
0	1	0	2	4	5	1	2	6	3	7	4	5	6	8	9

M

1	2	4	5	6	6	7	8	9	10	11	12	13	14	15	16

03

“MERGE SORT” ON GPU

MERGE SORT ON CPU

8	4	12	2	2	1	5	7

MERGE SORT ON CPU

8	4	12	2	2	1	5	7

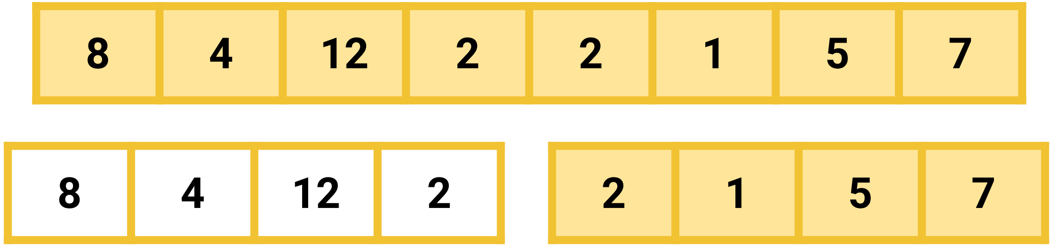
MERGE SORT ON CPU

8	4	12	2	2	1	5	7

8	4	12	2

2	1	5	7

MERGE SORT ON CPU

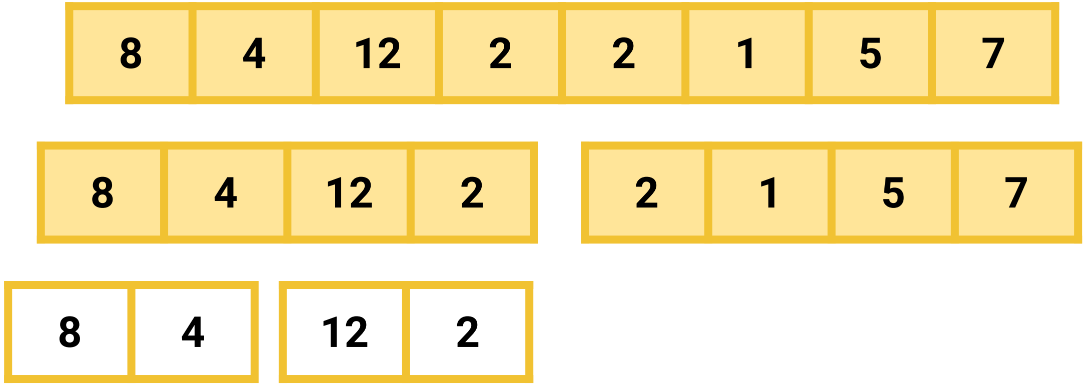


8	4	12	2	2	1	5	7

8	4	12	2

2	1	5	7

MERGE SORT ON CPU

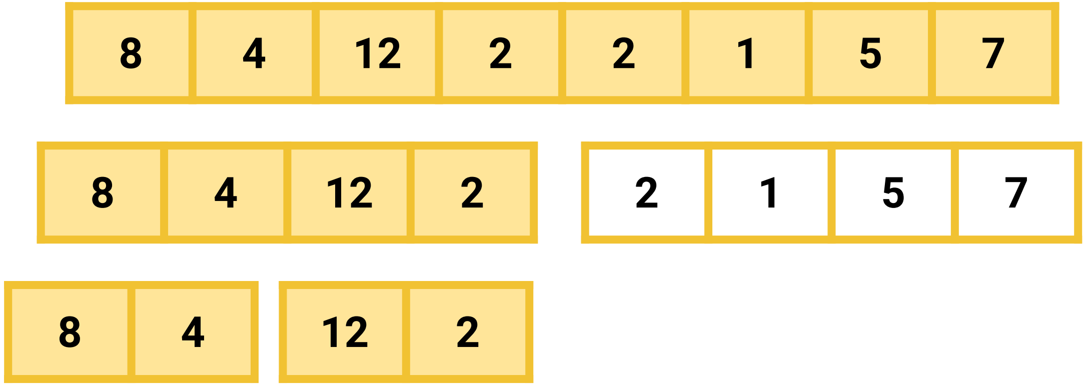


8 4 12 2 2 1 5 7

8 4 12 2   2 1 5 7

8 4   12 2

MERGE SORT ON CPU



8	4	12	2	2	1	5	7

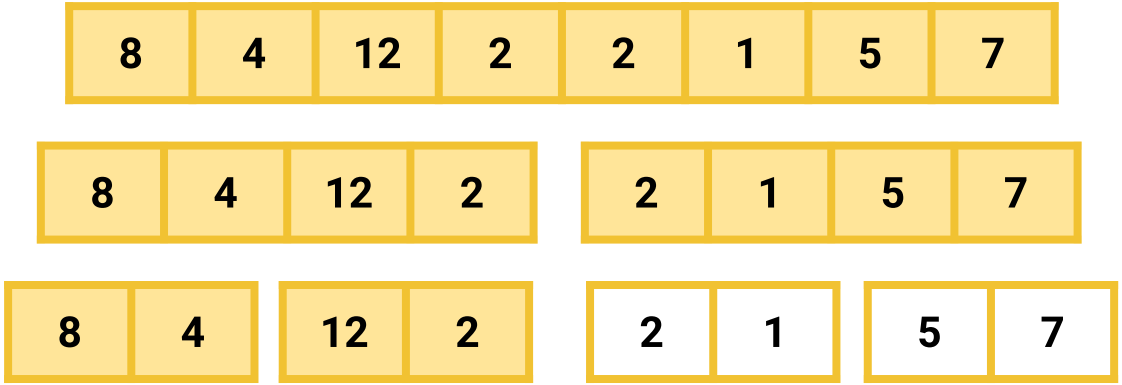
8	4	12	2

2	1	5	7

8	4

12	2

MERGE SORT ON CPU

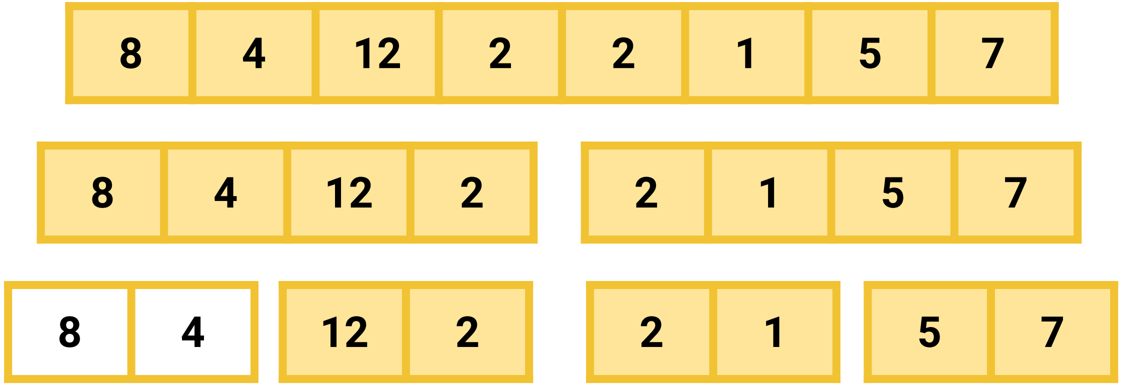


8 4 12 2 2 1 5 7

8 4 12 2 | 2 1 5 7

8 4 | 12 2 | 2 1 | 5 7

MERGE SORT ON CPU

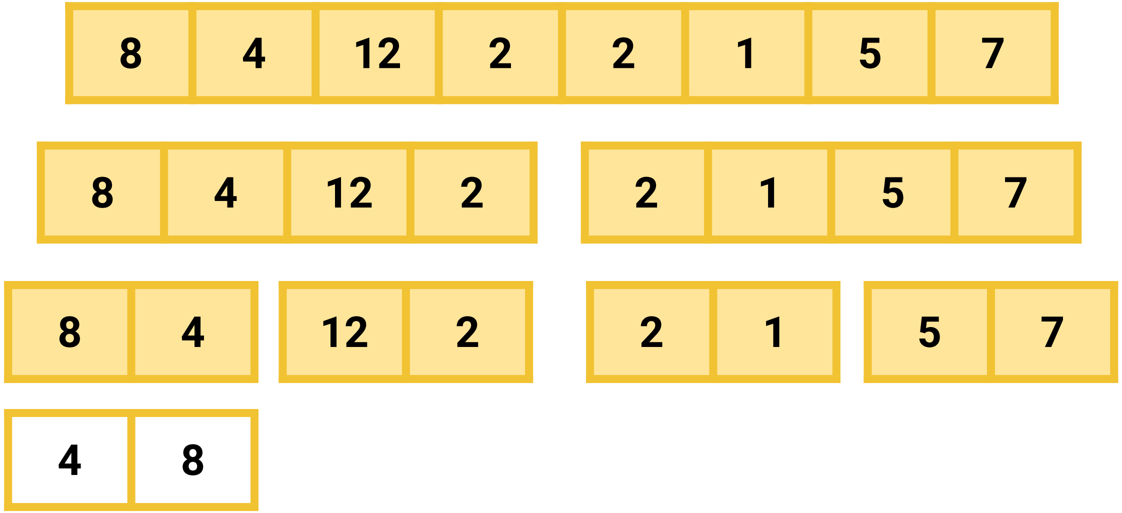


8 4 12 2 2 1 5 7

8 4 12 2 | 2 1 5 7

8 4 | 12 2 | 2 1 | 5 7

MERGE SORT ON CPU



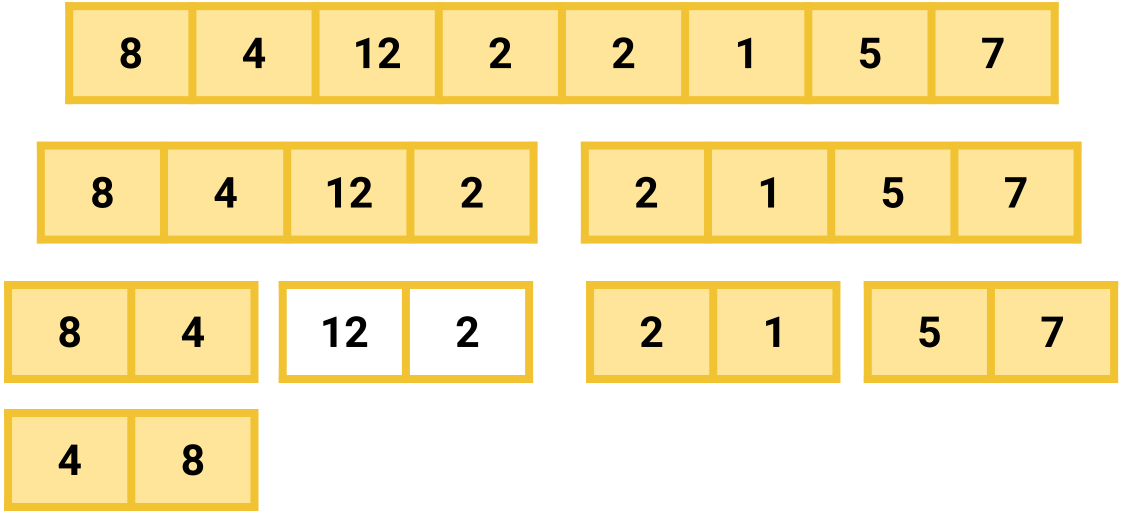
8 4 12 2 2 1 5 7

8 4 12 2 | 2 1 5 7

8 4 | 12 2 | 2 1 | 5 7

4 8

MERGE SORT ON CPU



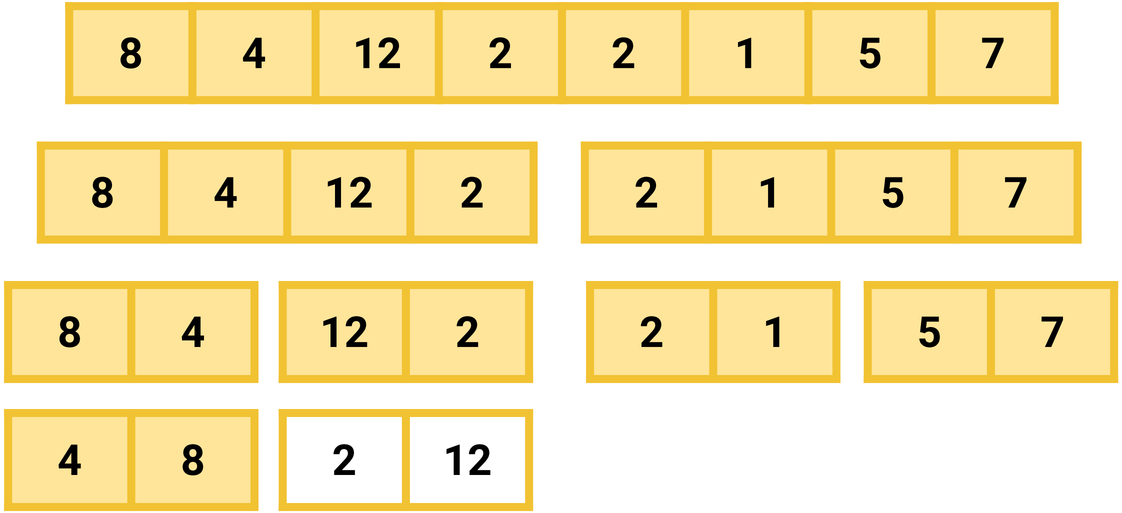
8 4 12 2 2 1 5 7

8 4 12 2 | 2 1 5 7

8 4 | 12 2 | 2 1 | 5 7

4 8

MERGE SORT ON CPU



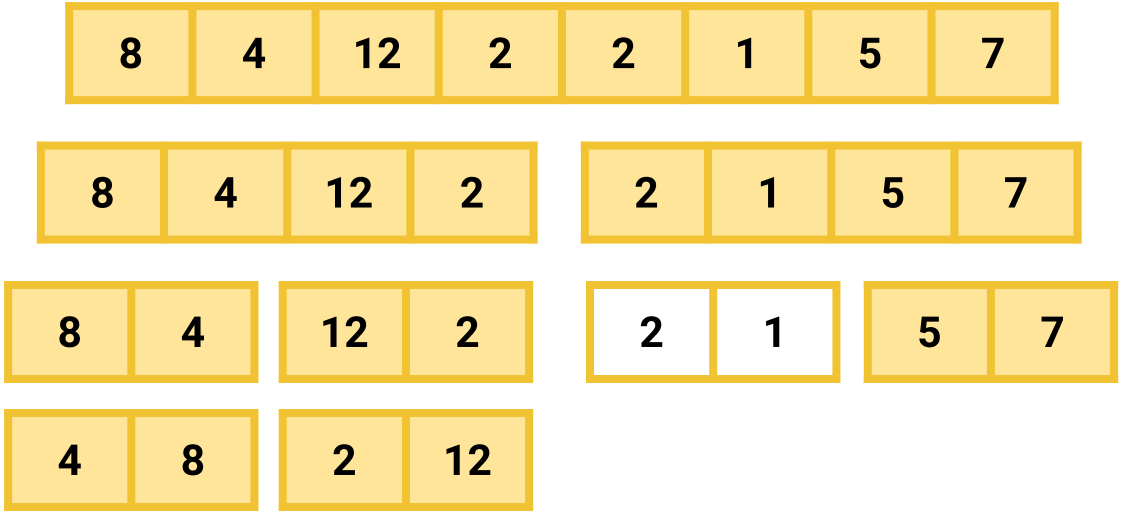
8 4 12 2 2 1 5 7

8 4 12 2 | 2 1 5 7

8 4 | 12 2 | 2 1 | 5 7

4 8 | 2 12

MERGE SORT ON CPU



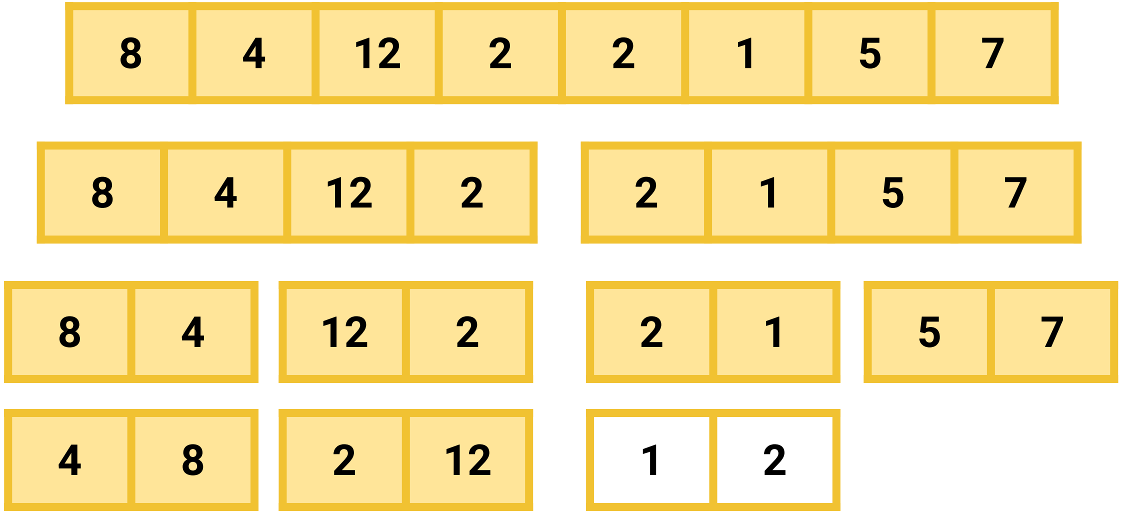
8 4 12 2 2 1 5 7

8 4 12 2 | 2 1 5 7

8 4 | 12 2 | 2 1 | 5 7

4 8 | 2 12

MERGE SORT ON CPU



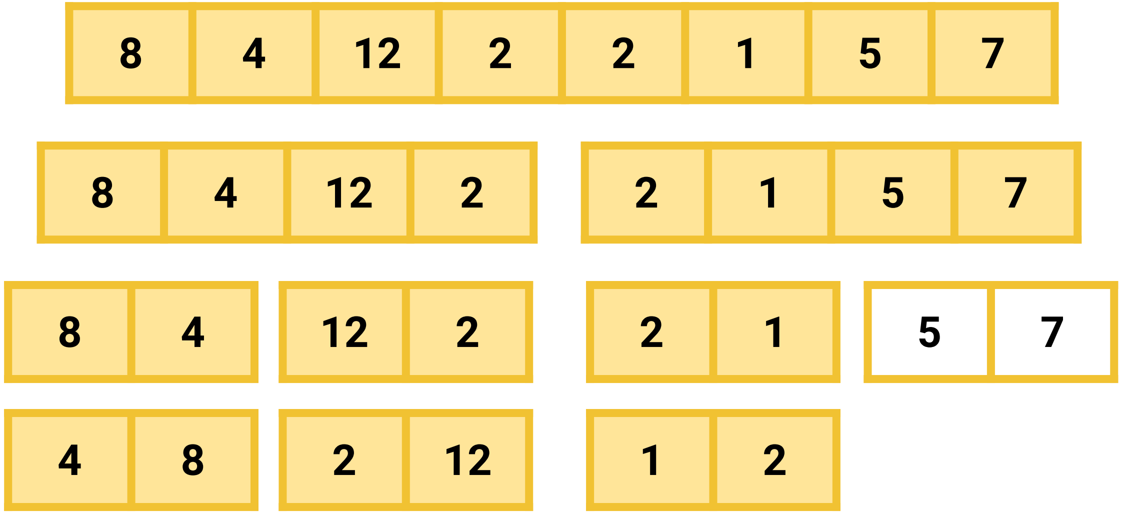
8 4 12 2 2 1 5 7

8 4 12 2 | 2 1 5 7

8 4 | 12 2 | 2 1 | 5 7

4 8 | 2 12 | 1 2

MERGE SORT ON CPU



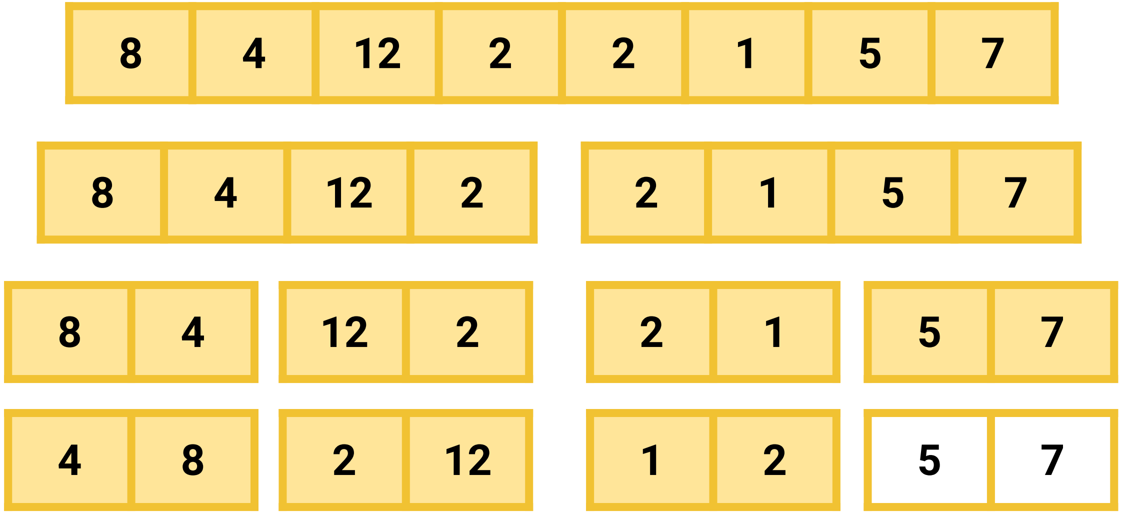
8 4 12 2 2 1 5 7

8 4 12 2 | 2 1 5 7

8 4 | 12 2 | 2 1 | 5 7

4 8 | 2 12 | 1 2

MERGE SORT ON CPU



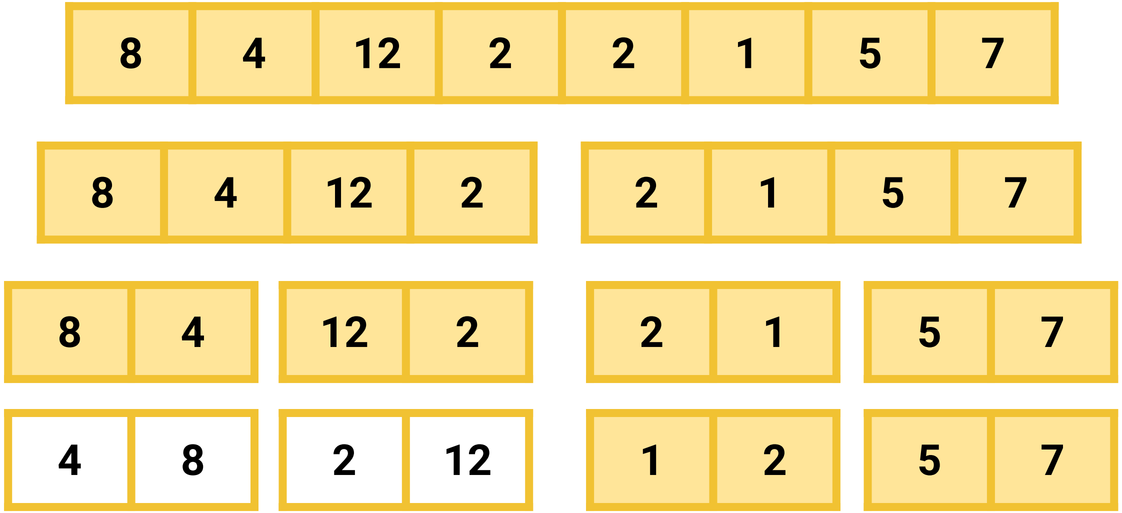
8 4 12 2 2 1 5 7

8 4 12 2 | 2 1 5 7

8 4 | 12 2 | 2 1 | 5 7

4 8 | 2 12 | 1 2 | 5 7

MERGE SORT ON CPU



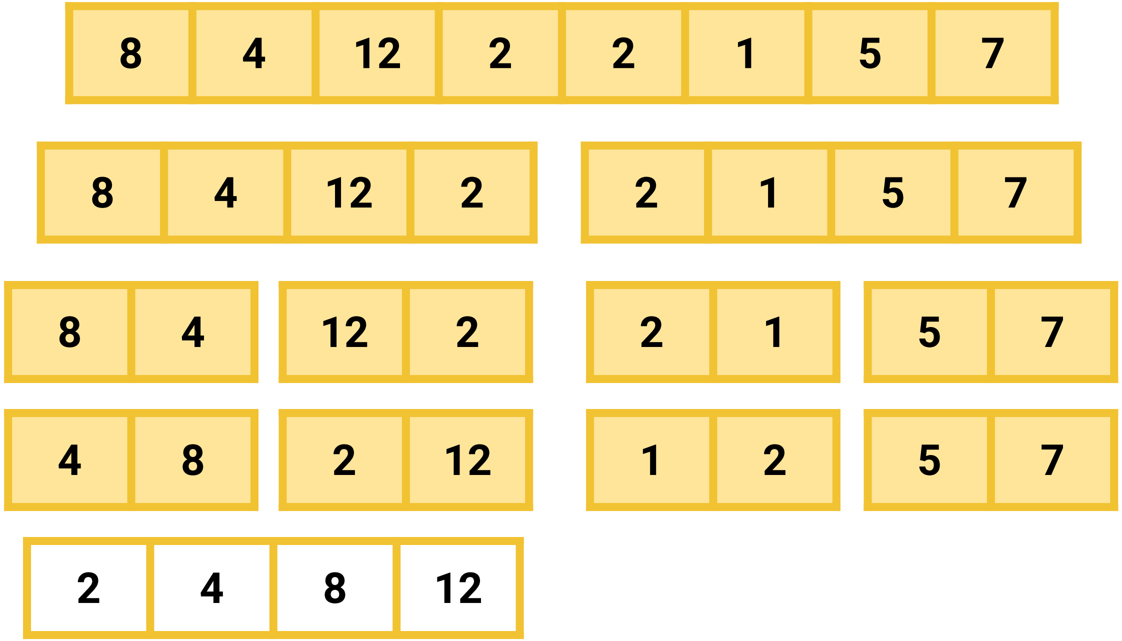
8 4 12 2 2 1 5 7

8 4 12 2 | 2 1 5 7

8 4 | 12 2 | 2 1 | 5 7

4 8 | 2 12 | 1 2 | 5 7

MERGE SORT ON CPU



8 4 12 2 2 1 5 7

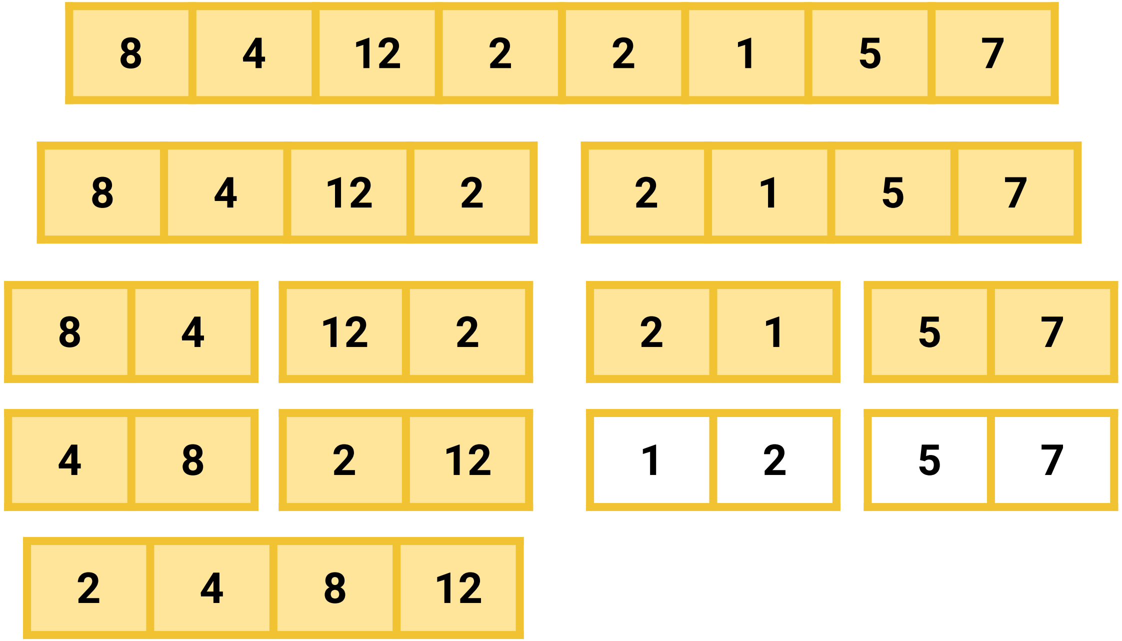
8 4 12 2 | 2 1 5 7

8 4 | 12 2 | 2 1 | 5 7

4 8 | 2 12 | 1 2 | 5 7

2 4 8 12

MERGE SORT ON CPU



8 4 12 2 2 1 5 7

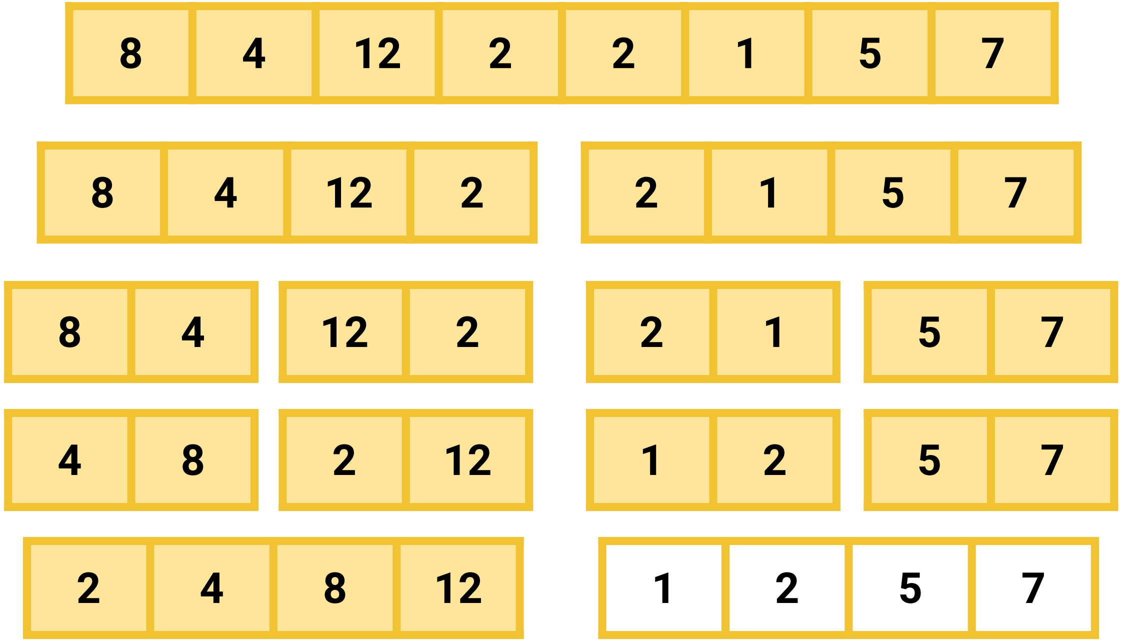
8 4 12 2 | 2 1 5 7

8 4 | 12 2 | 2 1 | 5 7

4 8 | 2 12 | 1 2 | 5 7

2 4 8 12

MERGE SORT ON CPU



8 4 12 2 2 1 5 7

8 4 12 2

2 1 5 7

8 4

12 2

2 1

5 7

4 8

2 12

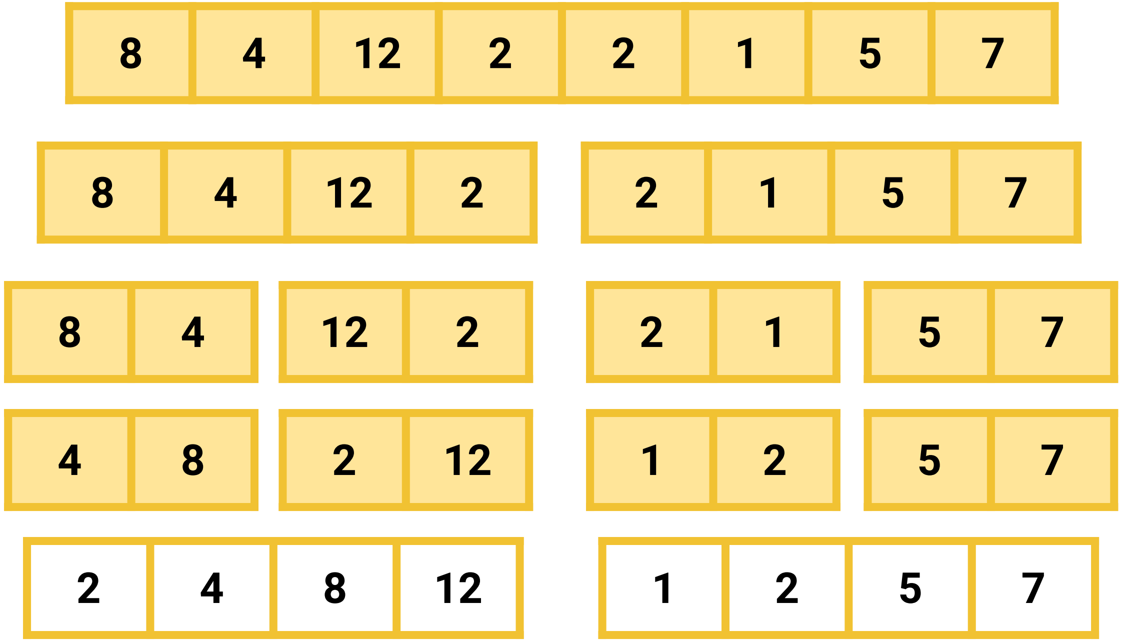
1 2

5 7

2 4 8 12

1 2 5 7

MERGE SORT ON CPU



8 4 12 2 2 1 5 7

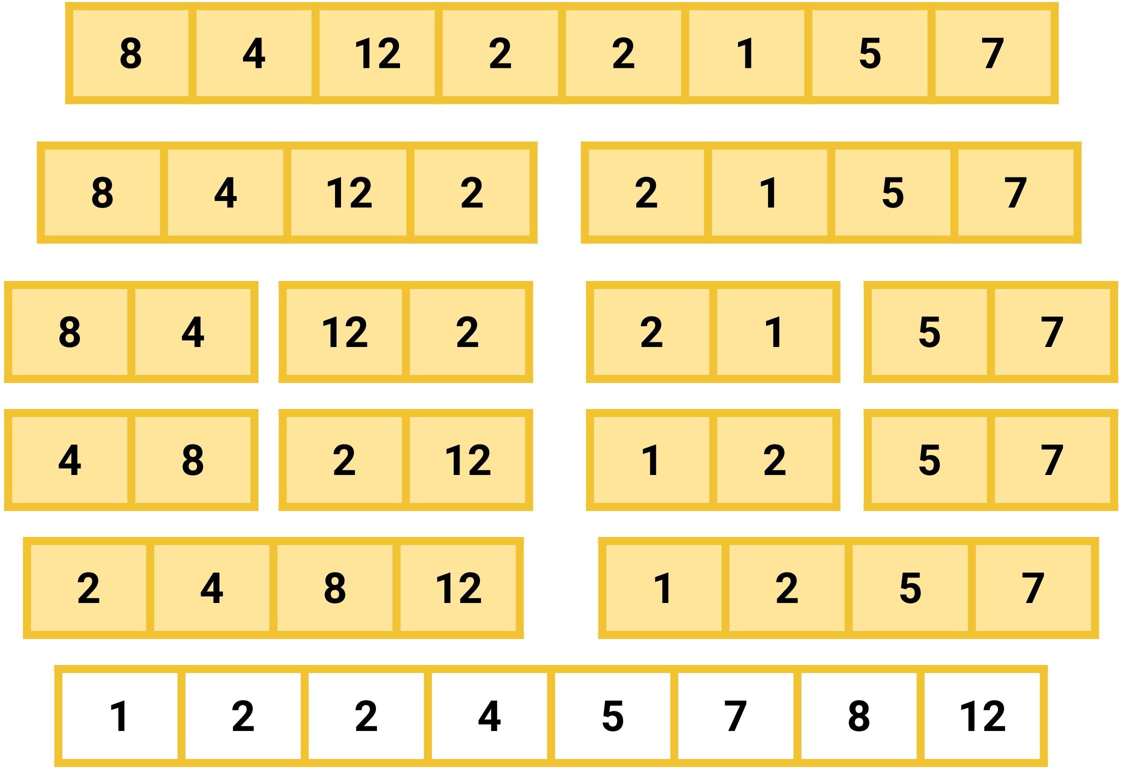
8 4 12 2 | 2 1 5 7

8 4 | 12 2 | 2 1 | 5 7

4 8 | 2 12 | 1 2 | 5 7

2 4 8 12 | 1 2 5 7

MERGE SORT ON CPU



8 4 12 2 2 1 5 7

8 4 12 2 | 2 1 5 7

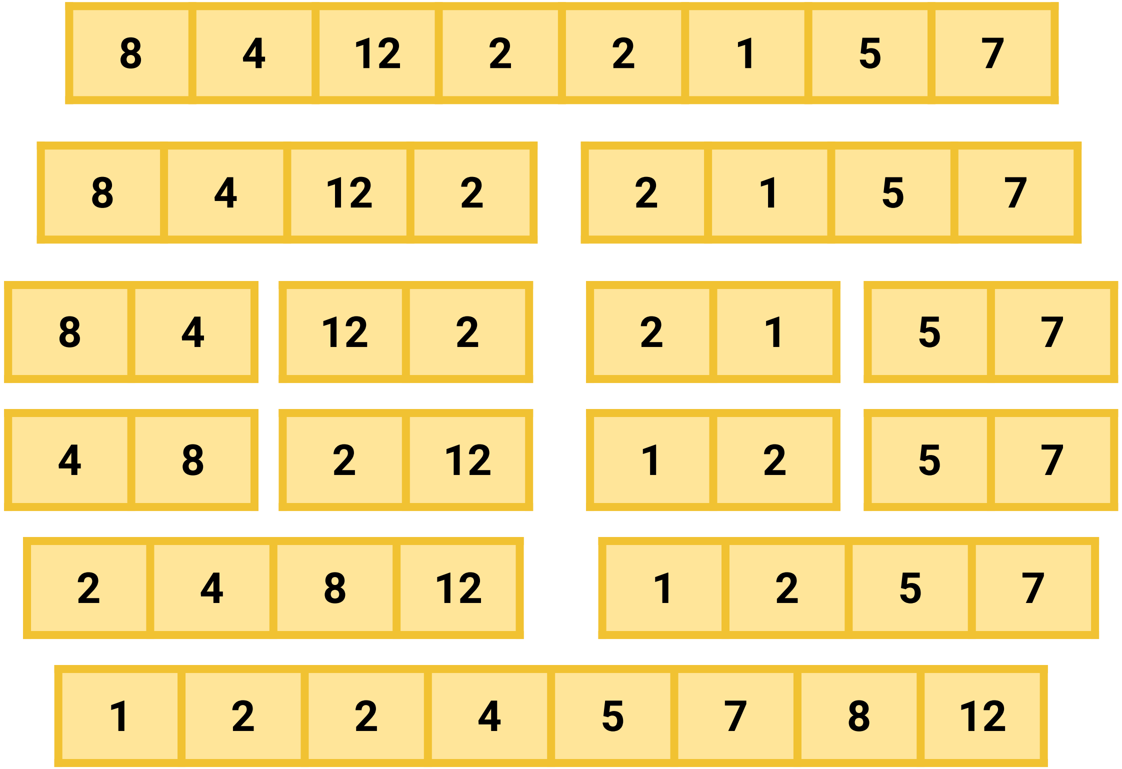
8 4 | 12 2 | 2 1 | 5 7

4 8 | 2 12 | 1 2 | 5 7

2 4 8 12 | 1 2 5 7

1 2 2 4 5 7 8 12

MERGE SORT ON CPU



8 4 12 2 2 1 5 7

8 4 12 2 | 2 1 5 7

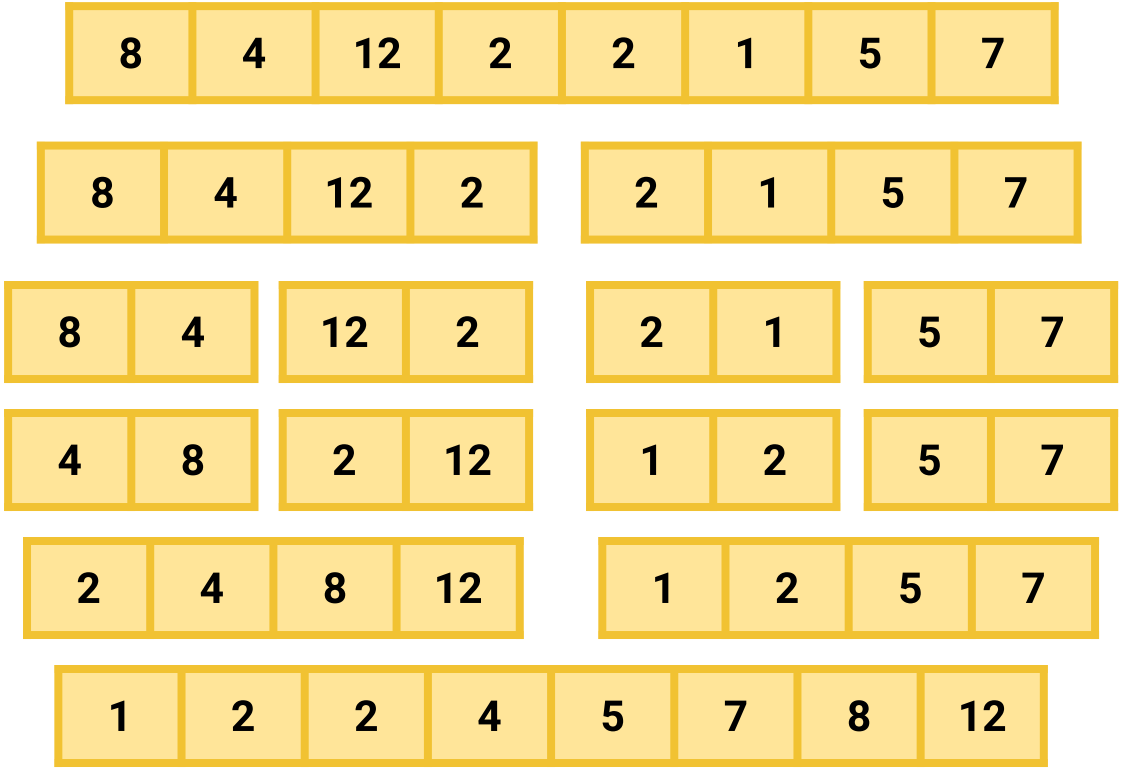
8 4 | 12 2 | 2 1 | 5 7

4 8 | 2 12 | 1 2 | 5 7

2 4 8 12 | 1 2 5 7

1 2 2 4 5 7 8 12

“MERGE SORT” ON GPU



8 4 12 2 2 1 5 7

8 4 12 2 | 2 1 5 7

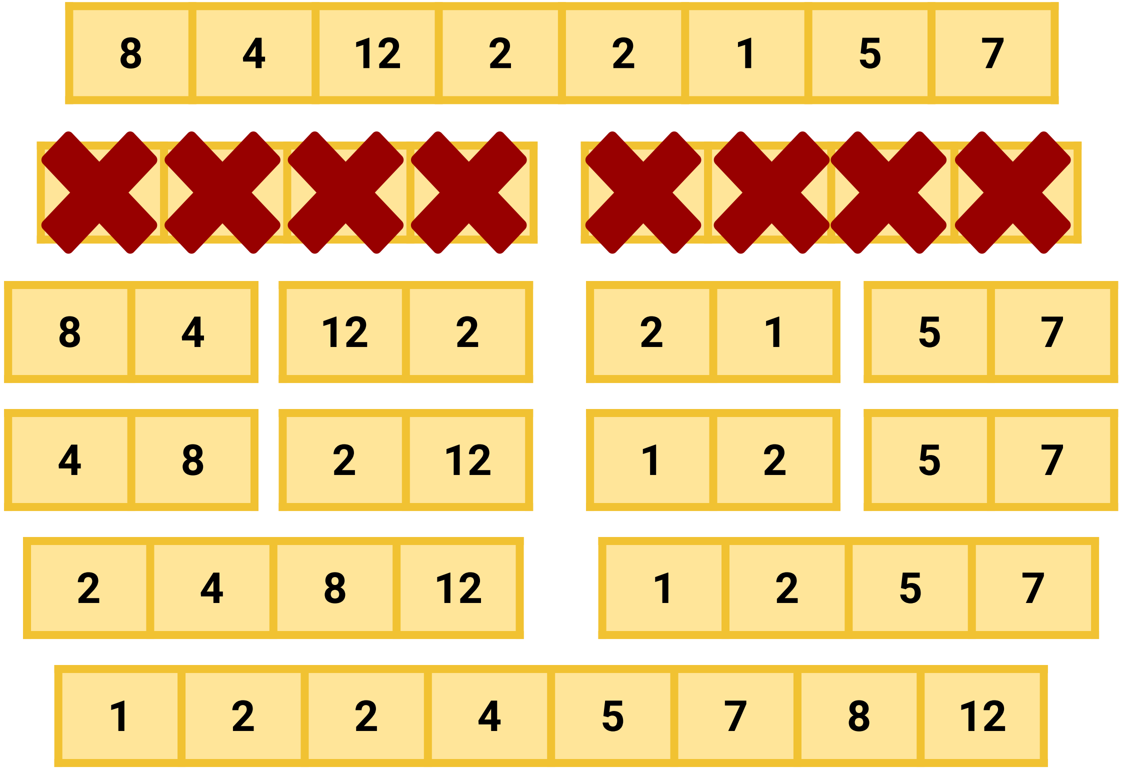
8 4 | 12 2 | 2 1 | 5 7

4 8 | 2 12 | 1 2 | 5 7

2 4 8 12 | 1 2 5 7

1 2 2 4 5 7 8 12

“MERGE SORT” ON GPU



8 4 12 2 2 1 5 7

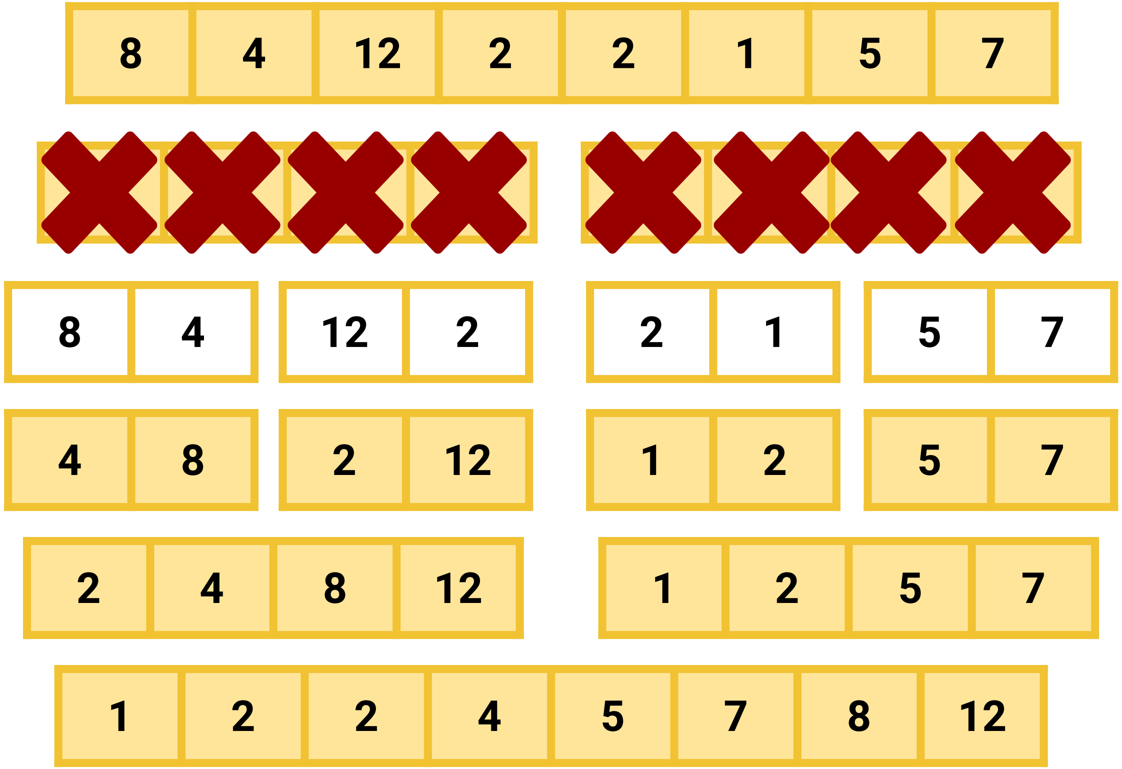
8 4 | 12 2 | 2 1 | 5 7

4 8 | 2 12 | 1 2 | 5 7

2 4 8 12 | 1 2 5 7

1 2 2 4 5 7 8 12

“MERGE SORT” ON GPU



8 4 12 2 2 1 5 7

8 4 | 12 2 | 2 1 | 5 7

4 8 | 2 12 | 1 2 | 5 7

2 4 8 12 | 1 2 5 7

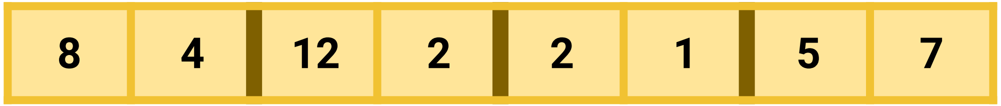
1 2 2 4 5 7 8 12

“MERGE SORT” ON GPU

8	4	12	2	2	1	5	7

“MERGE SORT” ON GPU

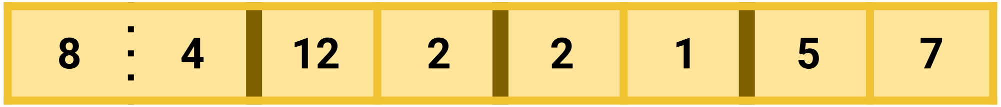
slice_size = 2
number_of_slices = 4



8	4	12	2	2	1	5	7

slice_size = 2
number_of_slices = 4

“MERGE SORT” ON GPU



8	4	12	2	2	1	5	7

NOTATIONS

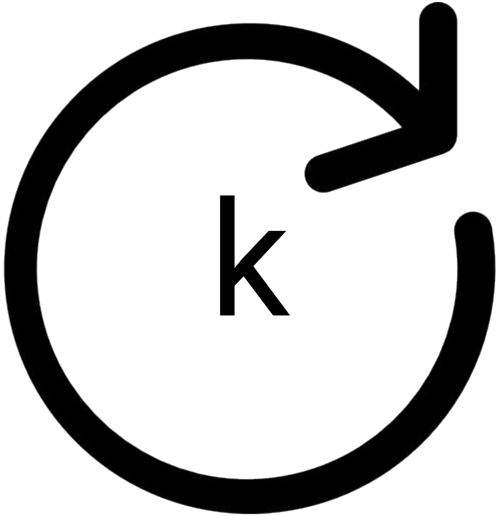
NOTATIONS

pathBig_k(A,B)

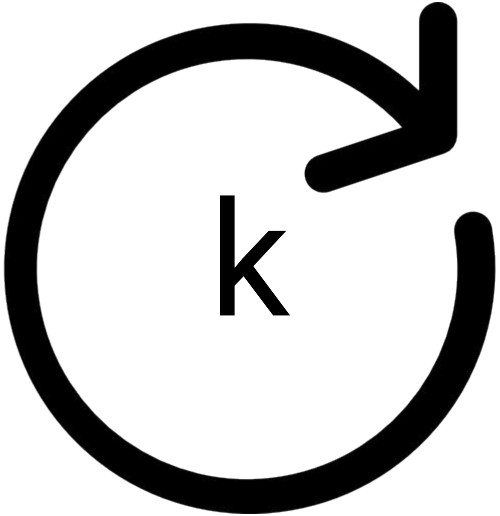
+

mergeBig_k(A,B)

NOTATIONS



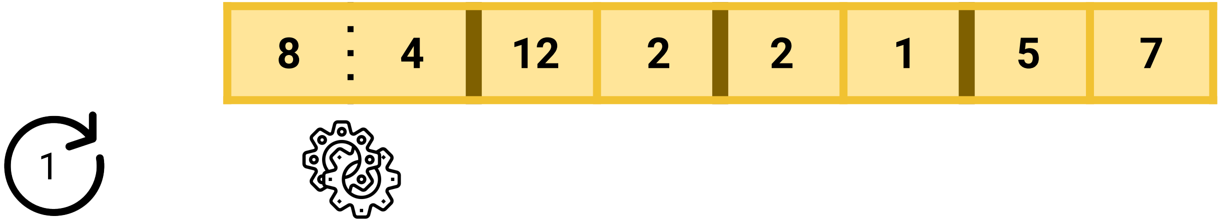
NOTATIONS



k loop tour

slice_size = 2
number_of_slices = 4

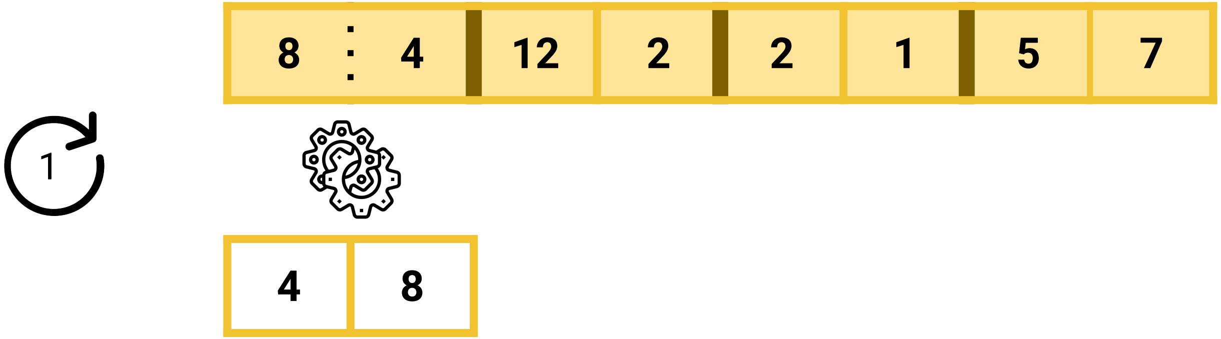
"MERGE SORT" ON GPU



8	4	12	2	2	1	5	7

slice_size = 2
number_of_slices = 4

“MERGE SORT” ON GPU

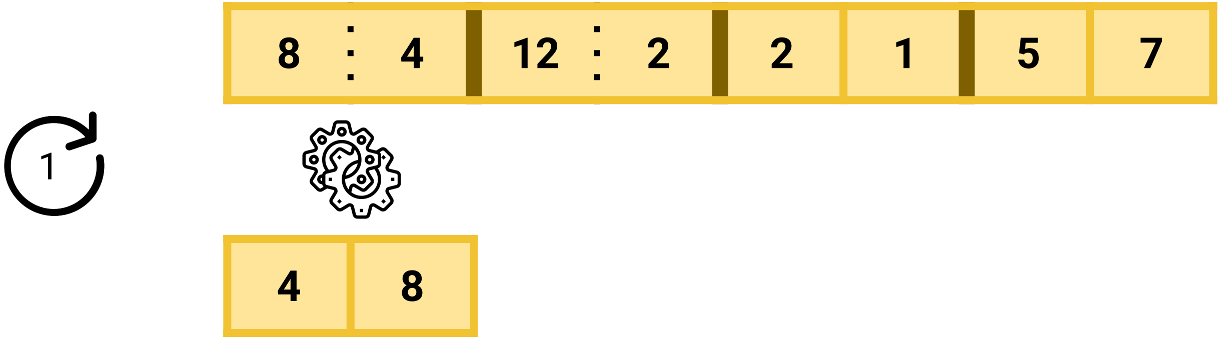


8	4	12	2	2	1	5	7

4	8

“MERGE SORT” ON GPU

slice_size = 2
number_of_slices = 4

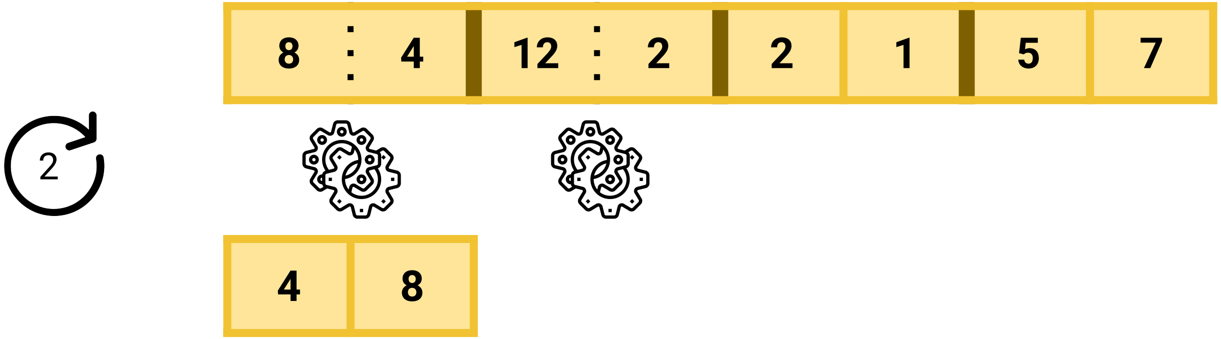

1

8 4 12 2 2 1 5 7

4 8

slice_size = 2
number_of_slices = 4

“MERGE SORT” ON GPU



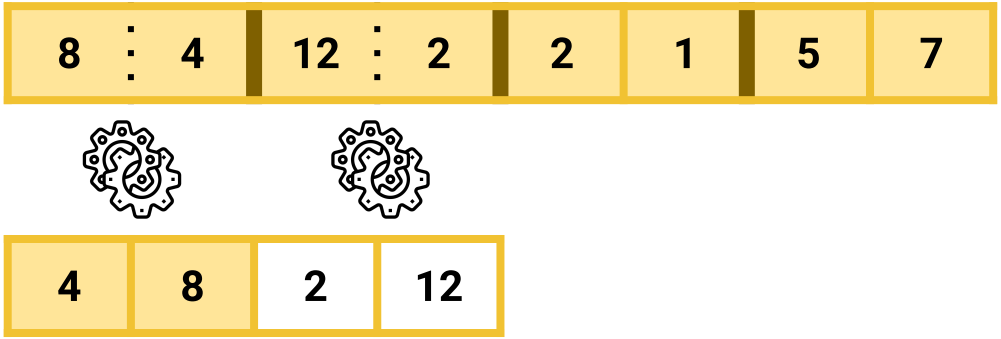
8	4	12	2	2	1	5	7

4	8

slice_size = 2
number_of_slices = 4

“MERGE SORT” ON GPU

2

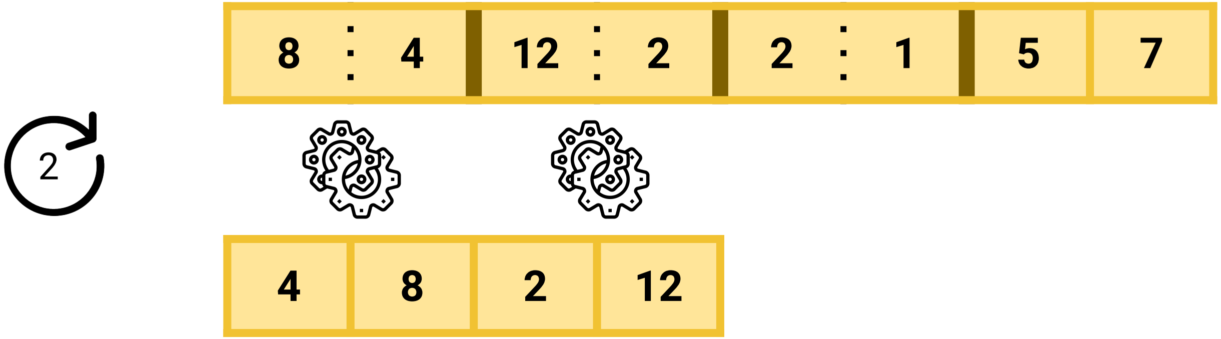


8	4	12	2	2	1	5	7

4	8	2	12

slice_size = 2
number_of_slices = 4

"MERGE SORT" ON GPU

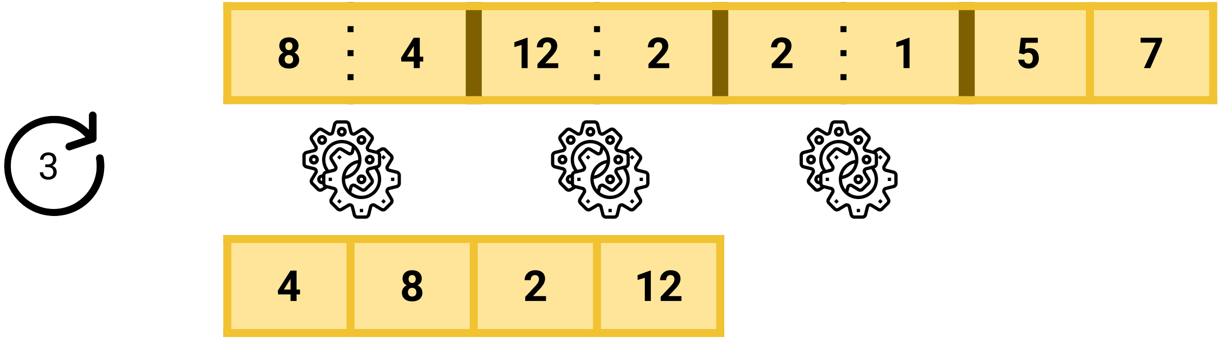


8	4	12	2	2	1	5	7

4	8	2	12

slice_size = 2
number_of_slices = 4

“MERGE SORT” ON GPU

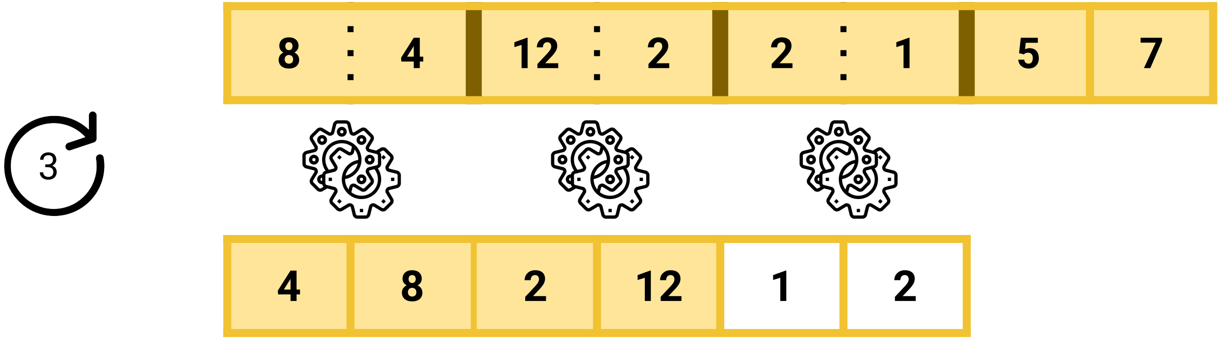


8	4	12	2	2	1	5	7

4	8	2	12

“MERGE SORT” ON GPU

slice_size = 2
number_of_slices = 4

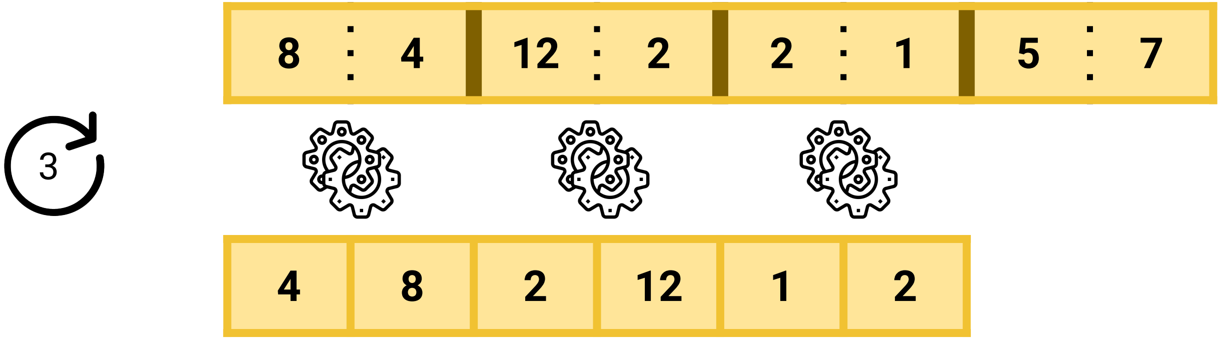


8	4	12	2	2	1	5	7

4	8	2	12	1	2

slice_size = 2
number_of_slices = 4

“MERGE SORT” ON GPU

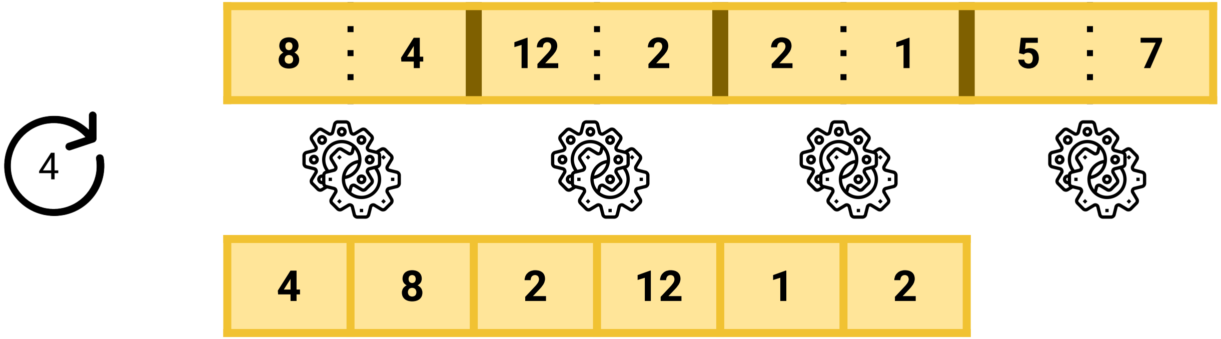


8	4	12	2	2	1	5	7

4	8	2	12	1	2

slice_size = 2
number_of_slices = 4

“MERGE SORT” ON GPU

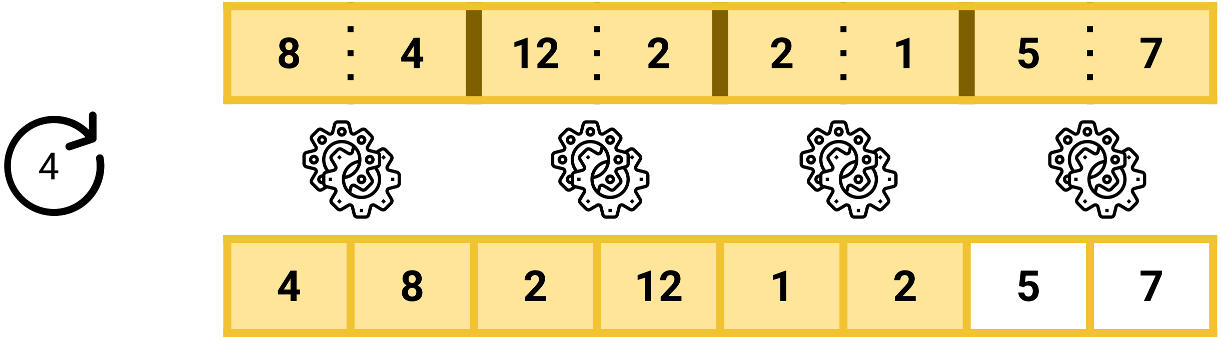


8 : 4 | 12 : 2 | 2 : 1 | 5 : 7

4	8	2	12	1	2

slice_size = 2
number_of_slices = 4

“MERGE SORT” ON GPU

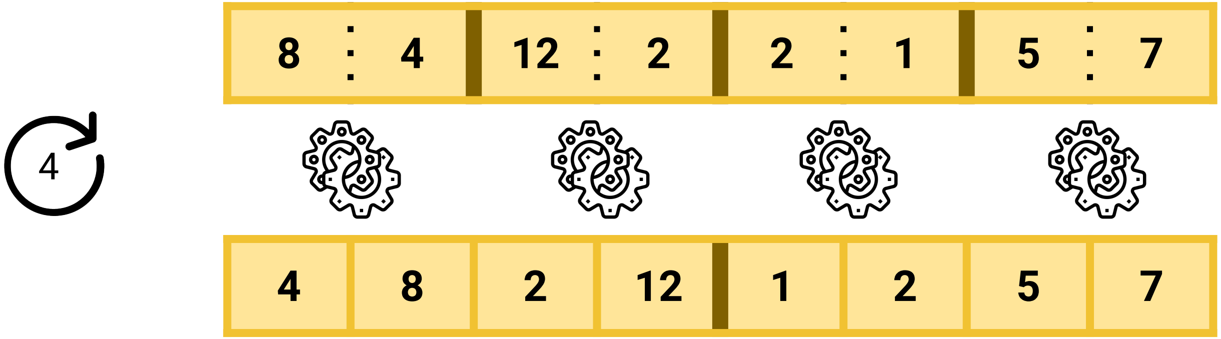


8	4	12	2	2	1	5	7

4	8	2	12	1	2	5	7

“MERGE SORT” ON GPU

slice_size = 4
number_of_slices = 2

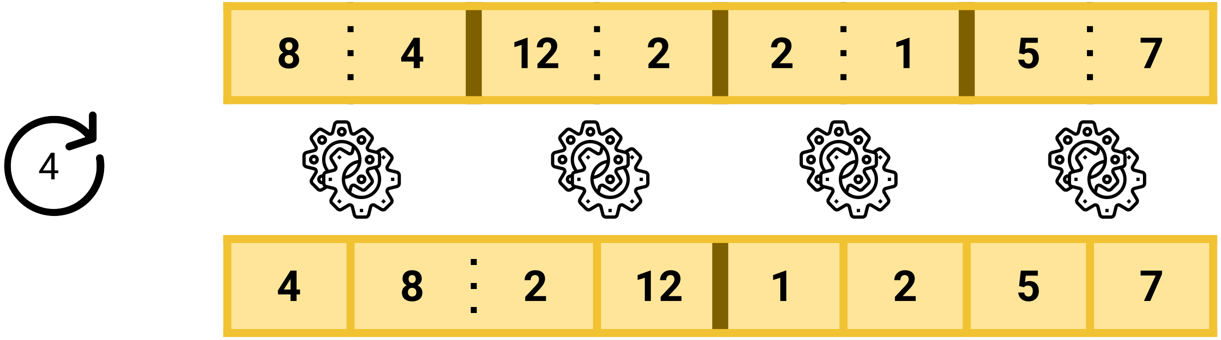


8	4	12	2	2	1	5	7

4	8	2	12	1	2	5	7

slice_size = 4
number_of_slices = 2

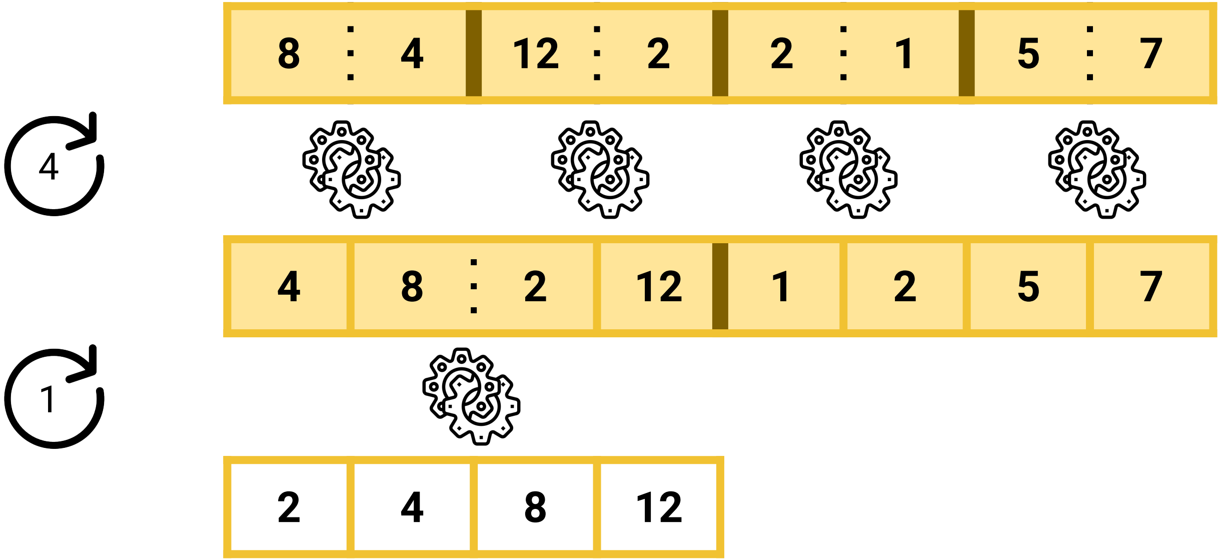
“MERGE SORT” ON GPU



8	4	12	2	2	1	5	7
4	8	2	12	1	2	5	7

slice_size = 4
number_of_slices = 2

“MERGE SORT” ON GPU



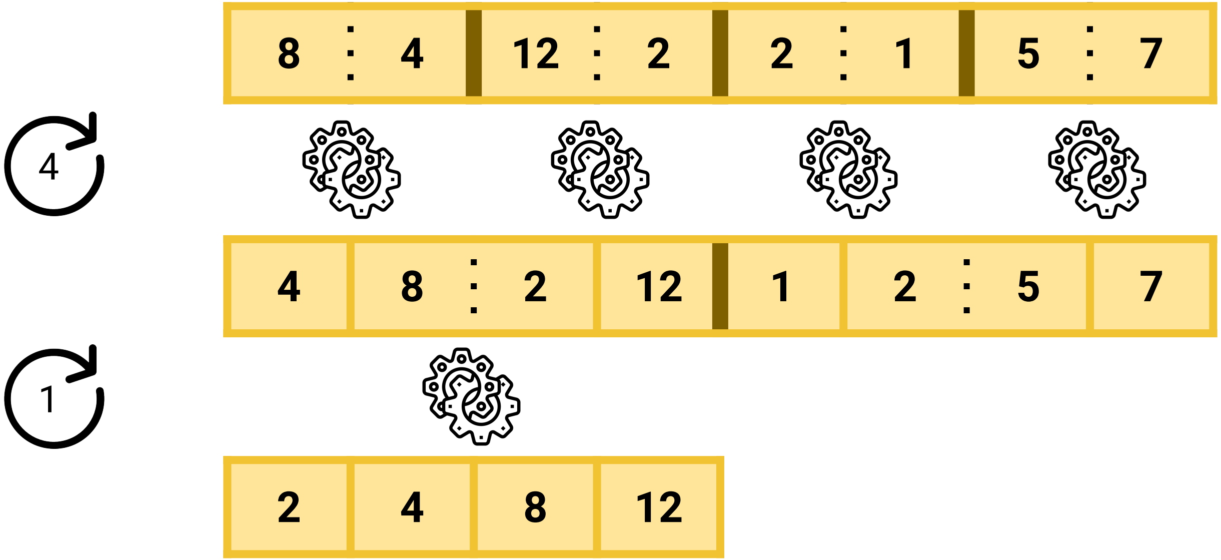
8 : 4 | 12 : 2 | 2 : 1 | 5 : 7

4 | 4 | 8 : 2 | 12 | 1 | 2 | 5 | 7

1 | 2 | 4 | 8 | 12

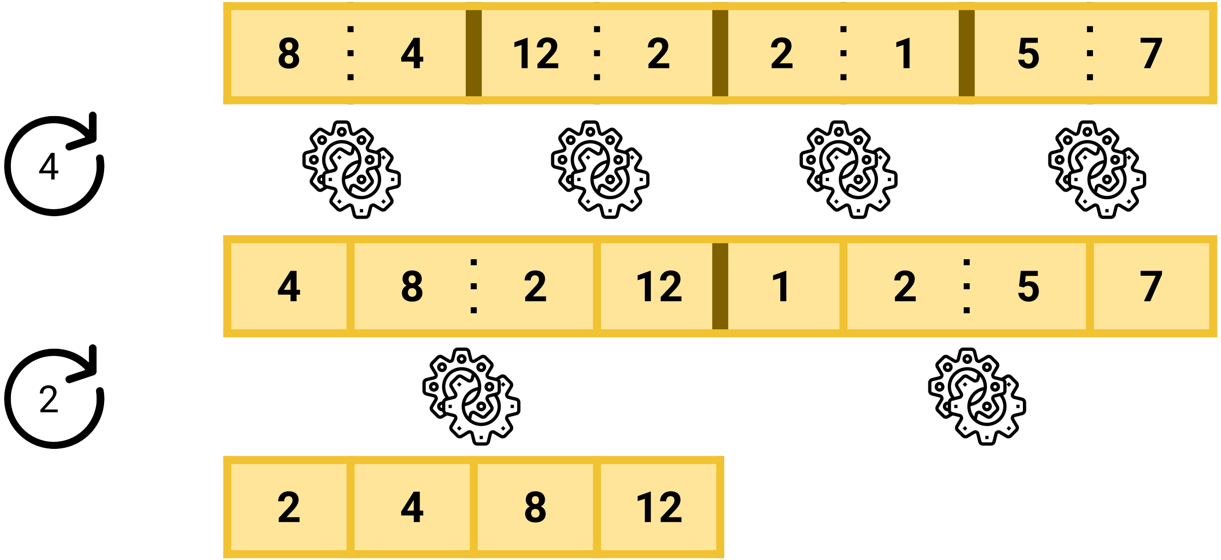
“MERGE SORT” ON GPU

slice_size = 4
number_of_slices = 2



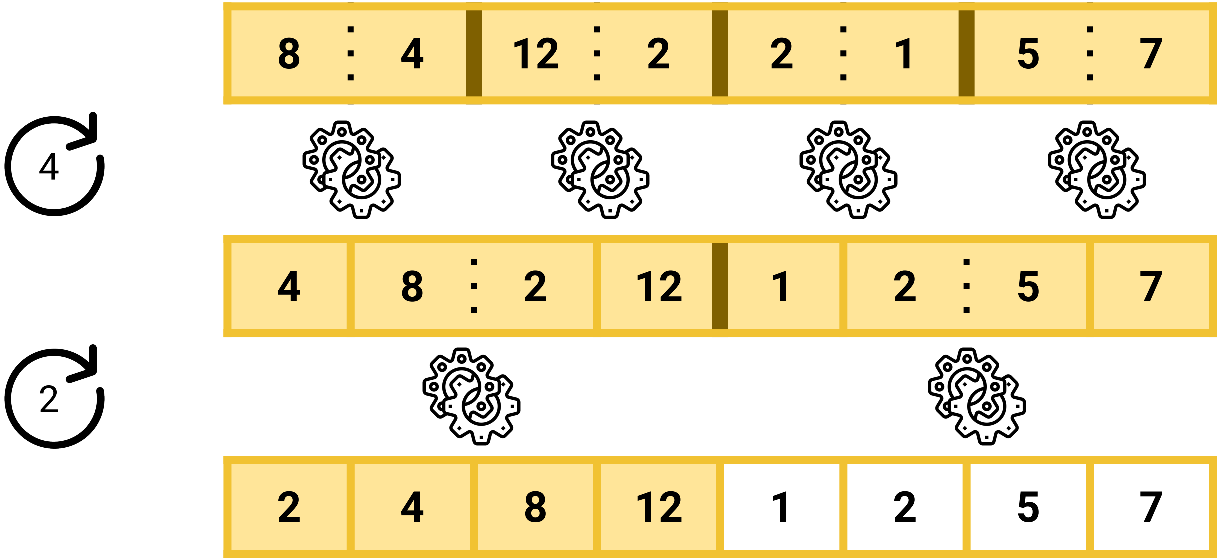
“MERGE SORT” ON GPU

slice_size = 4
number_of_slices = 2



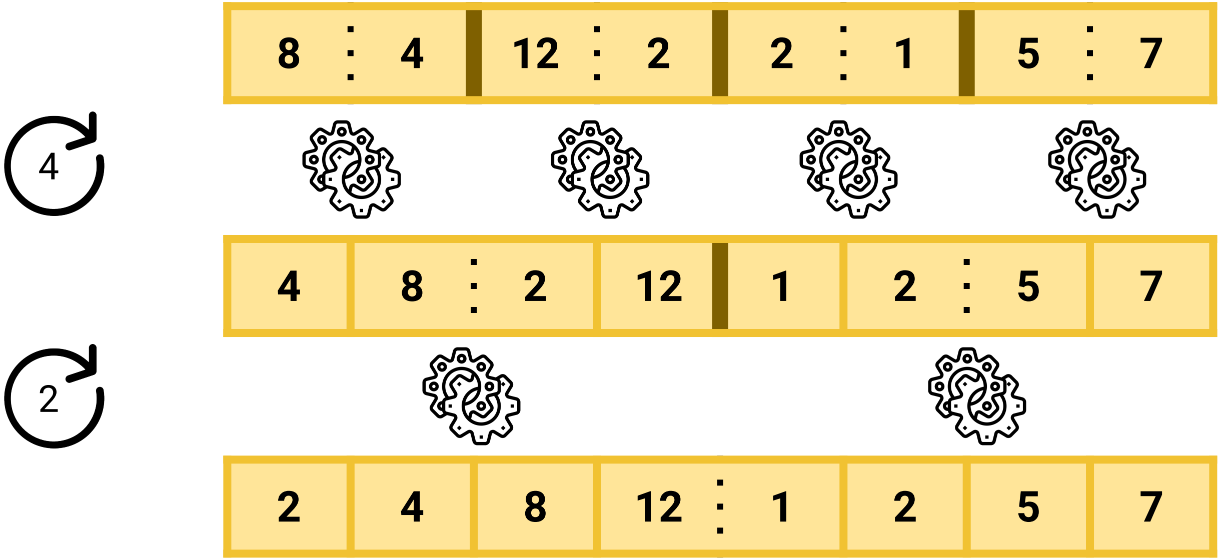
slice_size = 4
number_of_slices = 2

“MERGE SORT” ON GPU



slice_size = 8
number_of_slices = 1

“MERGE SORT” ON GPU



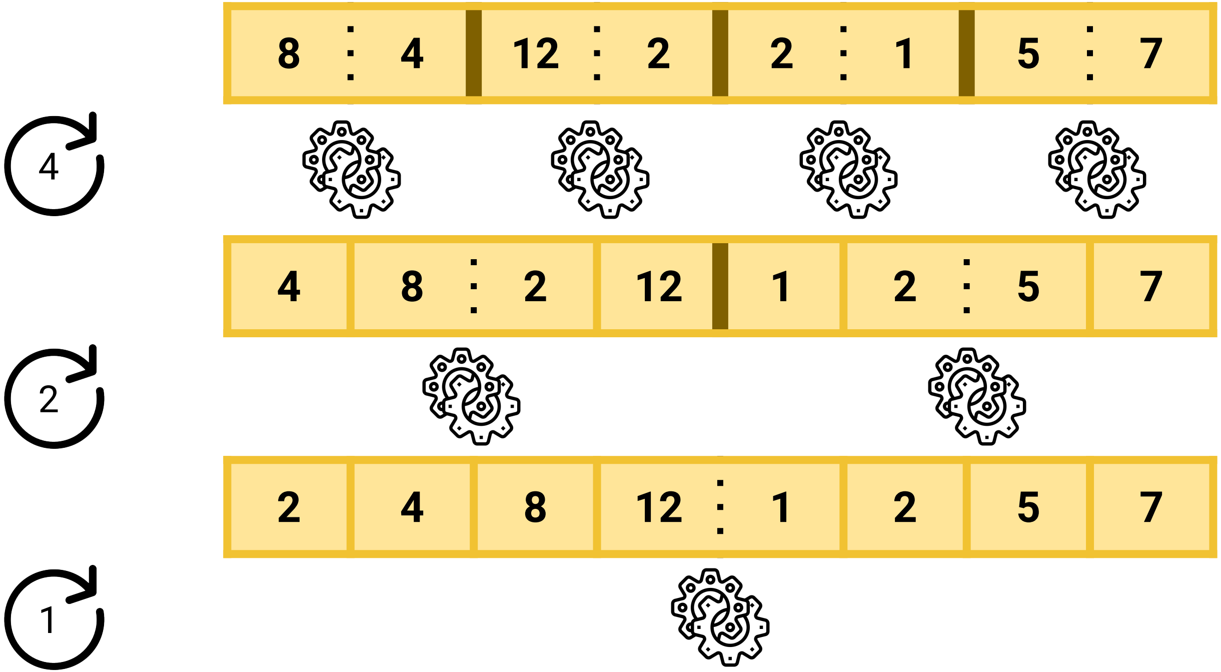
8 : 4 | 12 : 2 | 2 : 1 | 5 : 7

4 | 8 : 2 | 12 | 1 | 2 : 5 | 7

2 | 4 | 8 | 12 : 1 | 2 | 5 | 7

“MERGE SORT” ON GPU

slice_size = 8
number_of_slices = 1



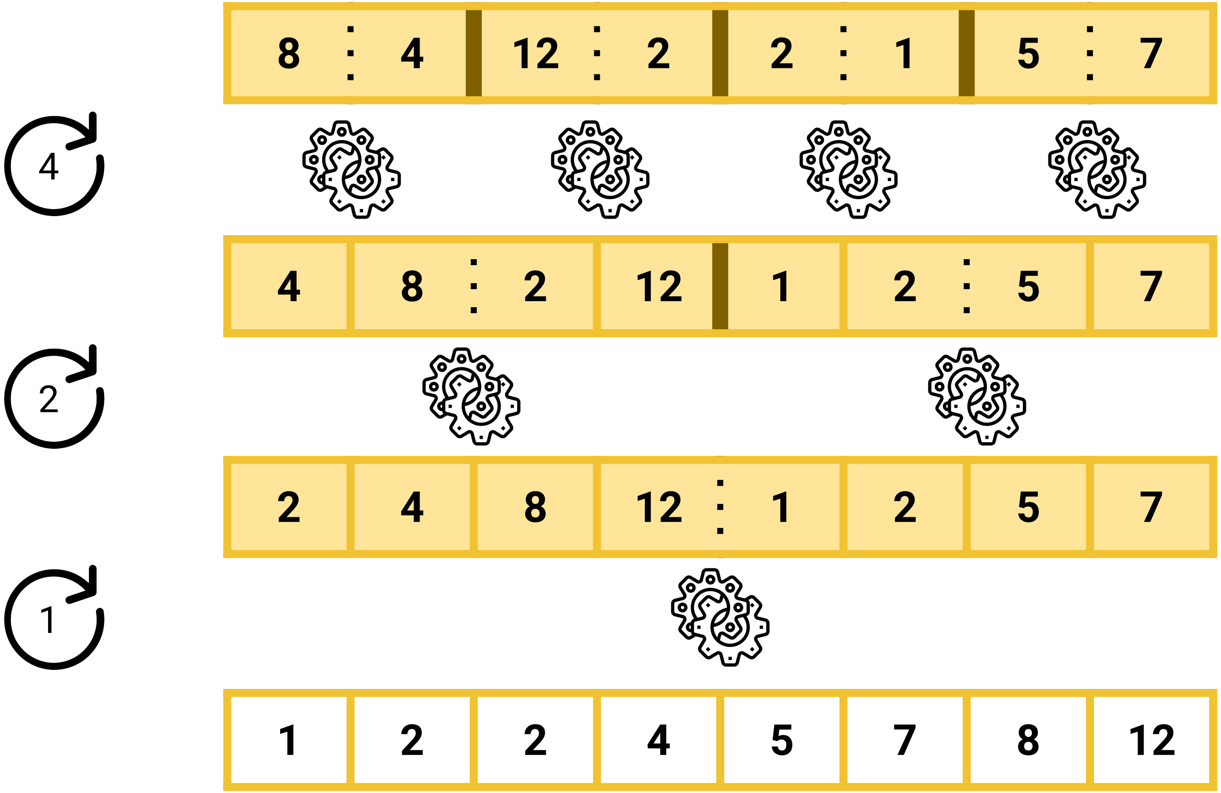
8 : 4 | 12 : 2 | 2 : 1 | 5 : 7

4 | 8 : 2 | 12 | 1 | 2 : 5 | 7

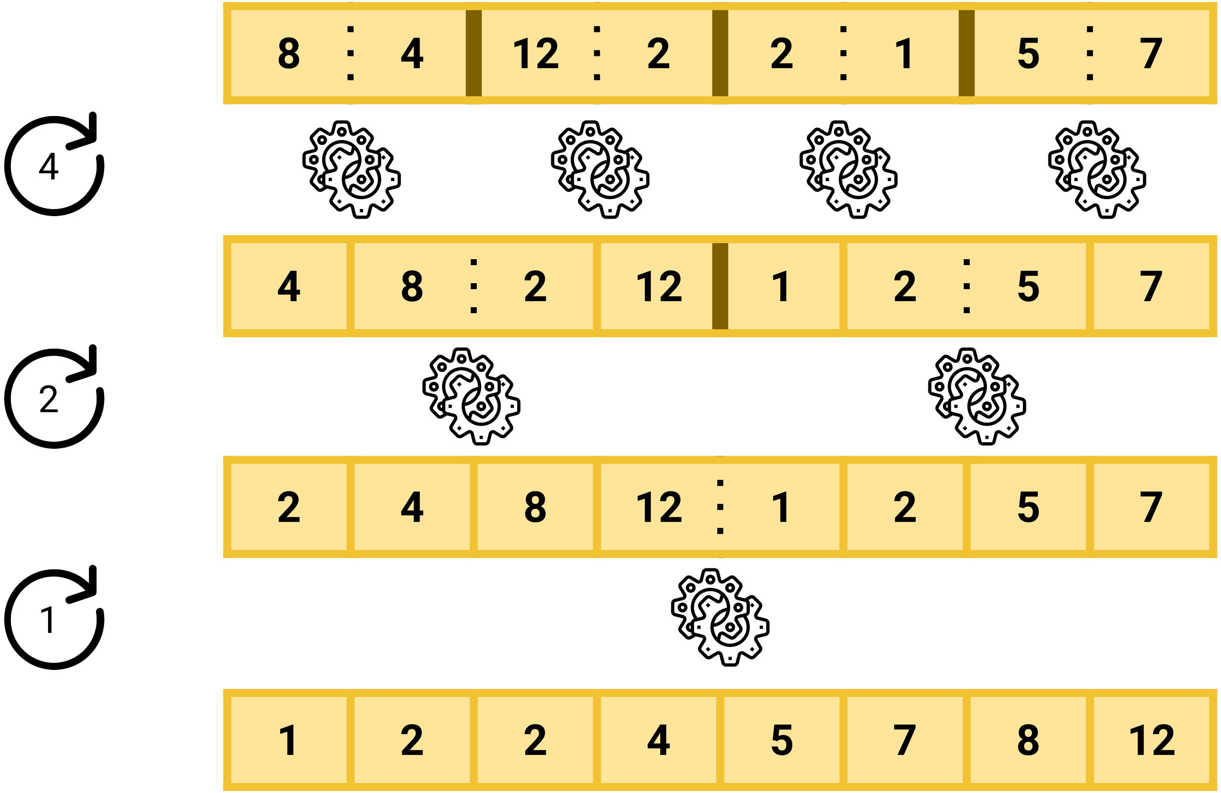
2 | 4 | 8 | 12 : 1 | 2 | 5 | 7

“MERGE SORT” ON GPU

slice_size = 8
number_of_slices = 1



"MERGE SORT" ON GPU



8 4 12 2 2 1 5 7

4 8 2 12 1 2 5 7

2 4 8 12 1 2 5 7

1 2 2 4 5 7 8 12

Ideal Case

Size of M is an exponent of 2

Every slices are full

More Often

M as any size

Some slices are not full

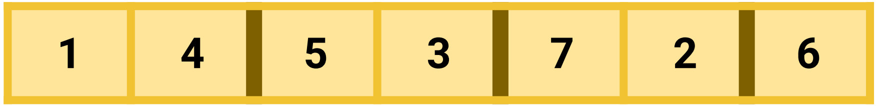
“MERGE SORT” ON GPU

1	4	5	3	7	2	6

“MERGE SORT” ON GPU

slice_size = 2

number_of_slices = 4



1	4	5	3	7	2	6

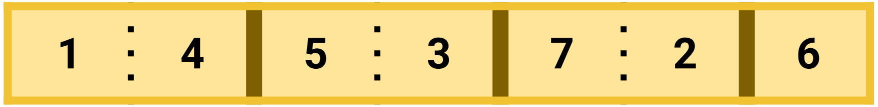
“MERGE SORT” ON GPU

slice_size = 2
number_of_slices = 4

1	4	5	3	7	2	6

“MERGE SORT” ON GPU

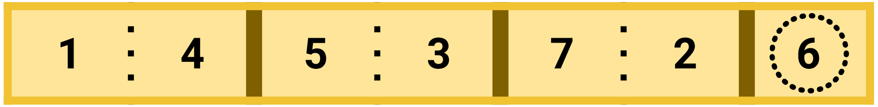
slice_size = 2
number_of_slices = **3**



1	4	5	3	7	2	6

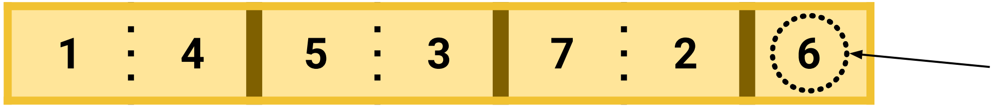
“MERGE SORT” ON GPU

slice_size = 2
number_of_slices = **3**



slice_size = 2
number_of_slices = **3**

“MERGE SORT” ON GPU



Remainder of 1 element

DEFINITION : REMAINDER

We call “remainder” an incomplete slice in our cutting

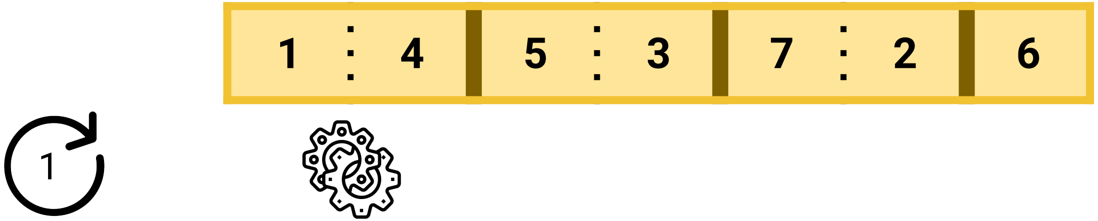
DEFINITION : REMAINDER

We call “remainder” an incomplete slice in our cutting

N.B : If there is a remainder, it’s always and only on the last slice

slice_size = 2
number_of_slices = **3**

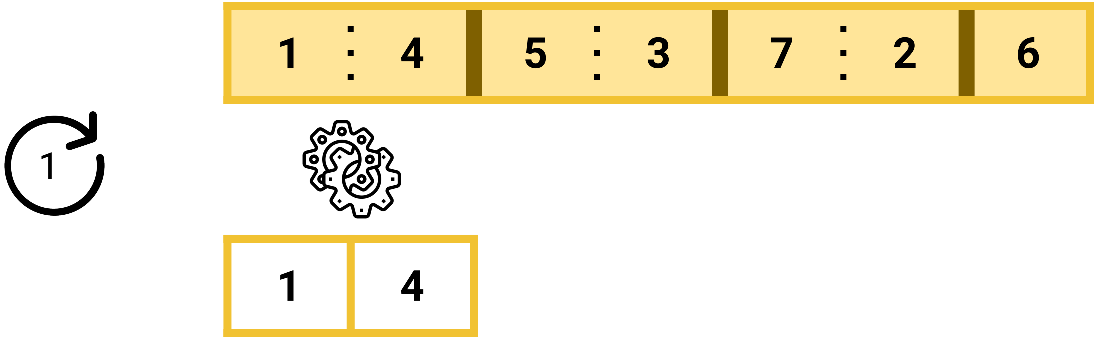
“MERGE SORT” ON GPU



1	4	5	3	7	2	6

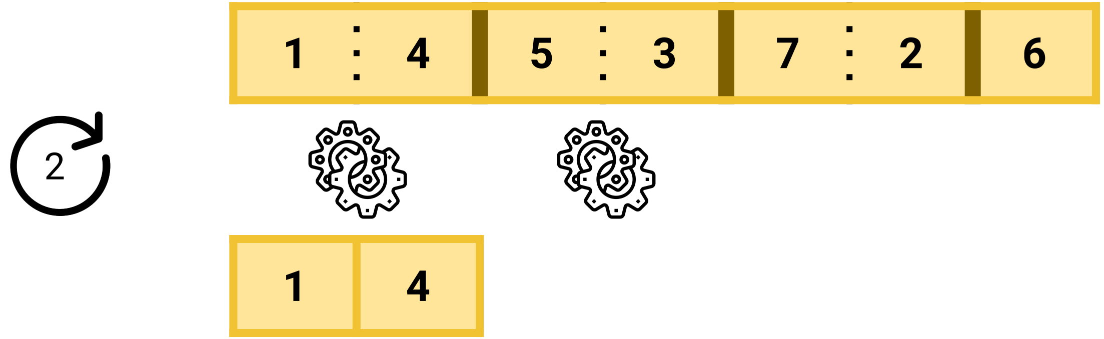
“MERGE SORT” ON GPU

slice_size = 2
number_of_slices = **3**



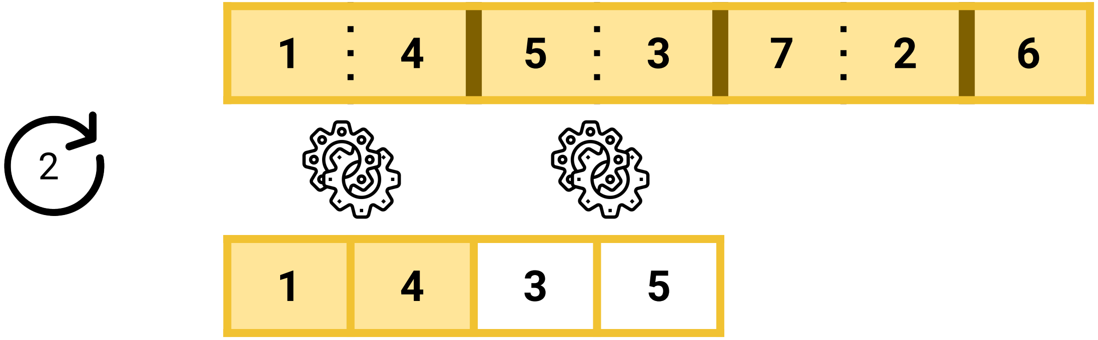
slice_size = 2
number_of_slices = **3**

"MERGE SORT" ON GPU



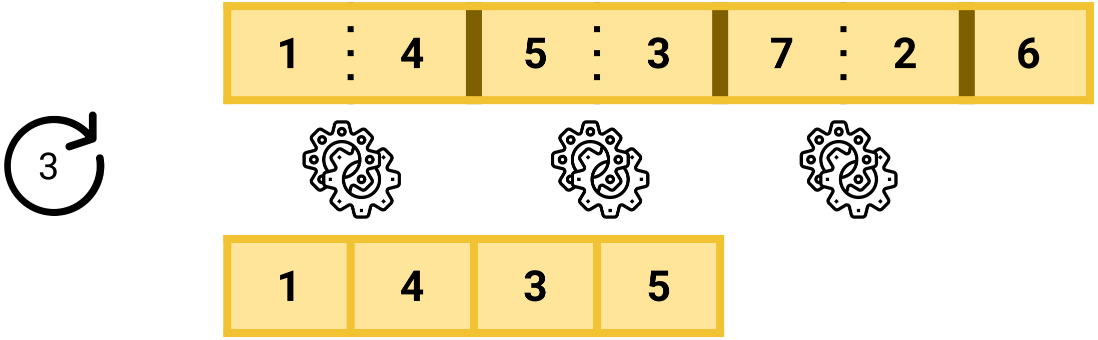
slice_size = 2
number_of_slices = **3**

“MERGE SORT” ON GPU



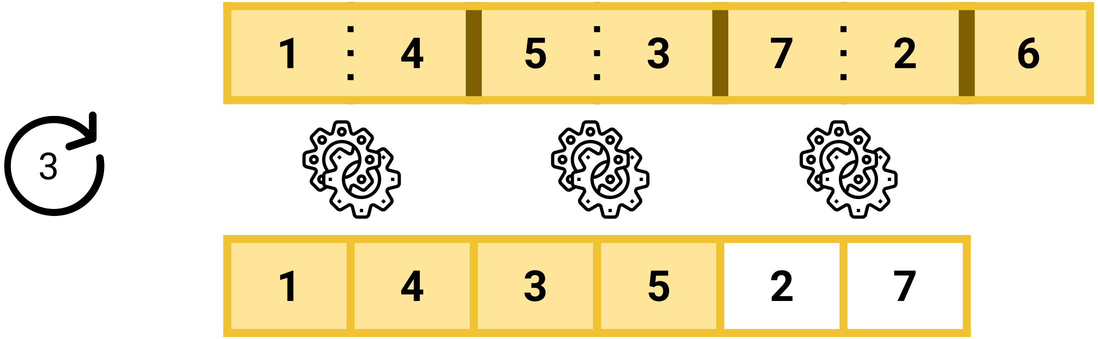
slice_size = 2
number_of_slices = **3**

“MERGE SORT” ON GPU



slice_size = 2
number_of_slices = **3**

“MERGE SORT” ON GPU

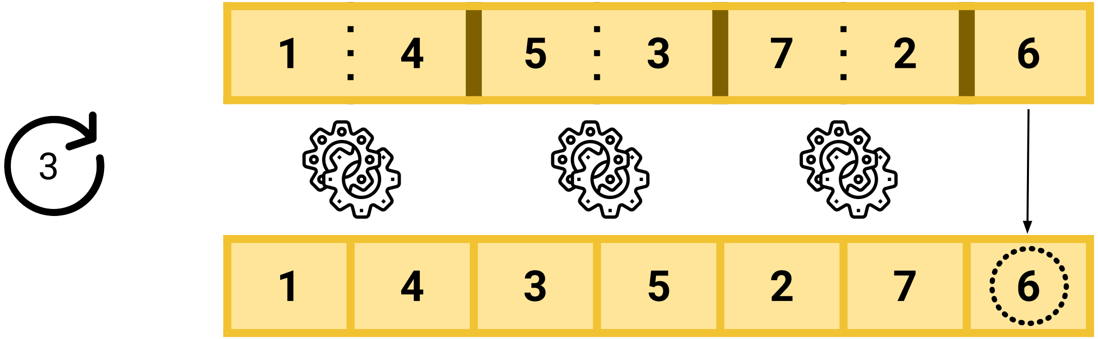


1	4	5	3	7	2	6

1	4	3	5	2	7

slice_size = 2
number_of_slices = **3**

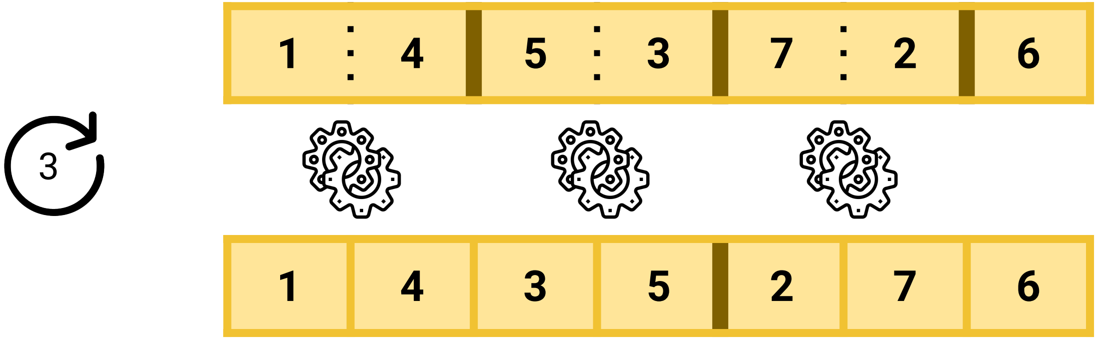
“MERGE SORT” ON GPU



1	4	5	3	7	2	6
1	4	3	5	2	7	6

slice_size = 4
number_of_slices = 2

“MERGE SORT” ON GPU

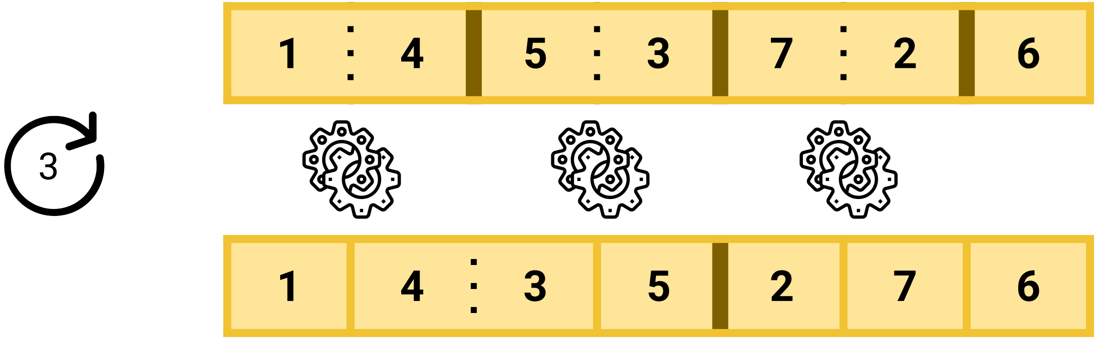


1	4	5	3	7	2	6

1	4	3	5	2	7	6

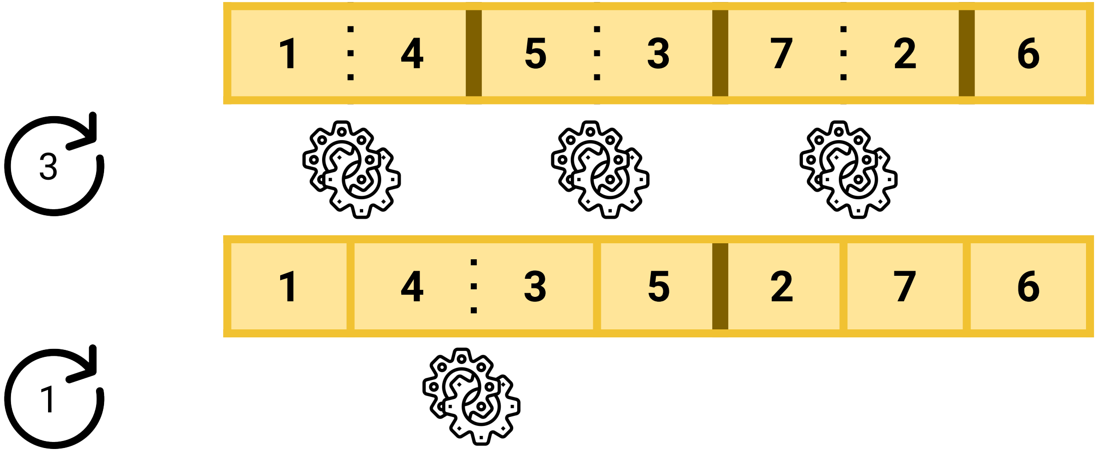
slice_size = 4
number_of_slices = **1**

"MERGE SORT" ON GPU



slice_size = 4
number_of_slices = **1**

“MERGE SORT” ON GPU



3

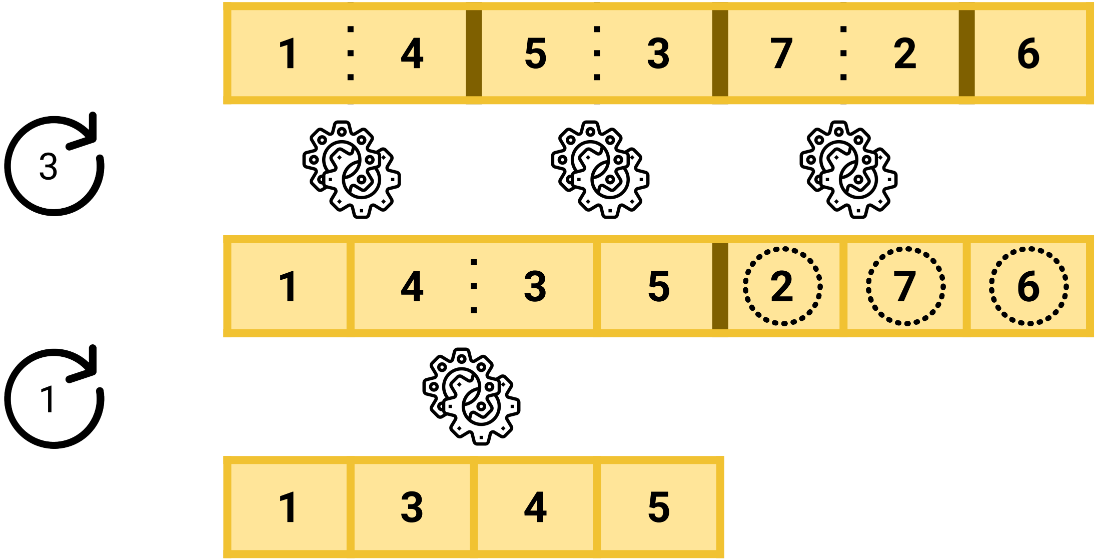
1 : 4 | 5 : 3 | 7 : 2 | 6

1

1 | 4 : 3 | 5 | 2 | 7 | 6

slice_size = 4
number_of_slices = **1**

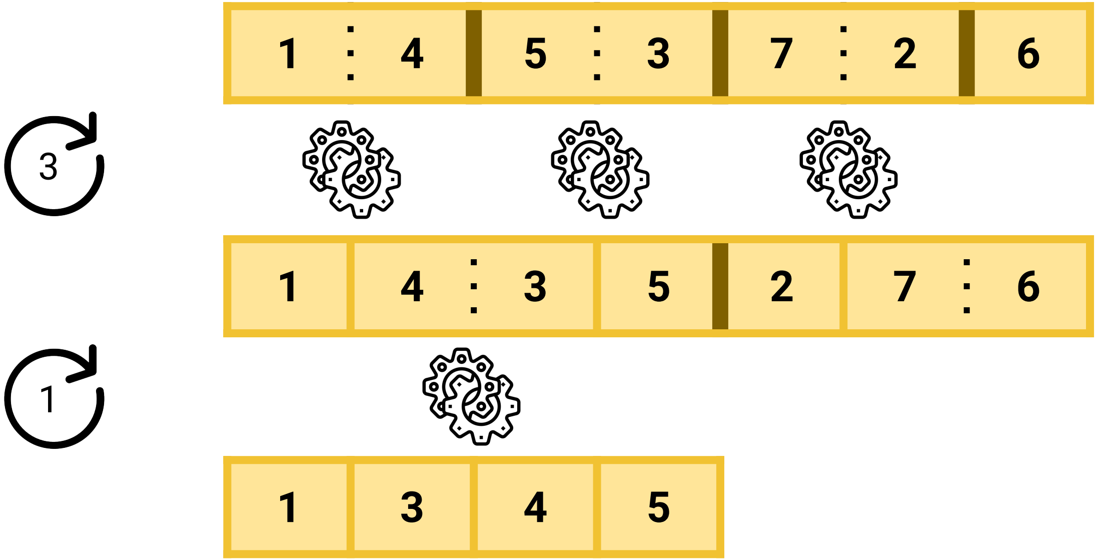
"MERGE SORT" ON GPU



Remainder of 3 elements

“MERGE SORT” ON GPU

slice_size = 4
number_of_slices = **1**



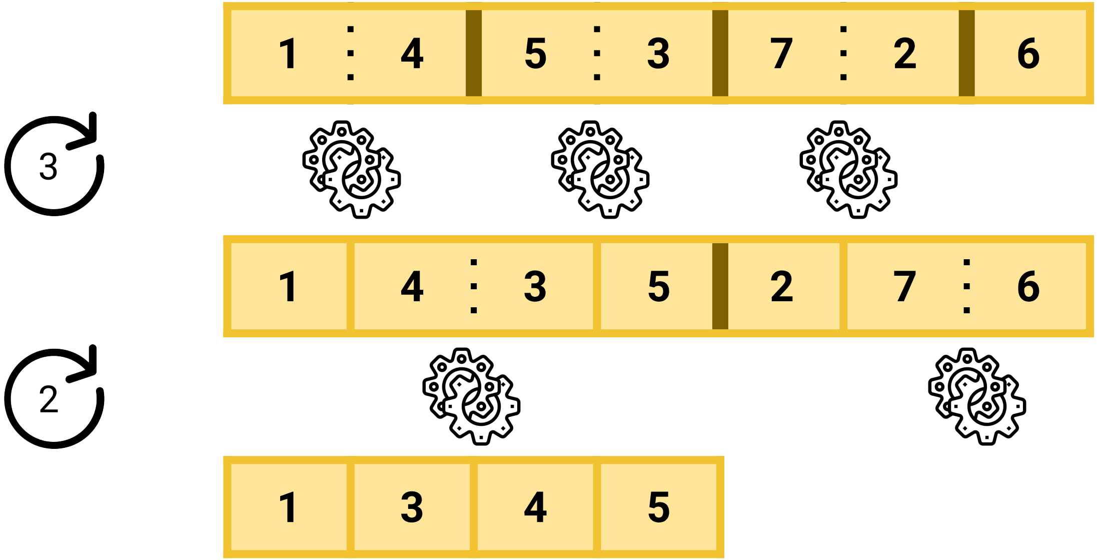
1 4 5 3 7 2 6

1 4 3 5 2 7 6

1 3 4 5

slice_size = 4
number_of_slices = **1**

“MERGE SORT” ON GPU



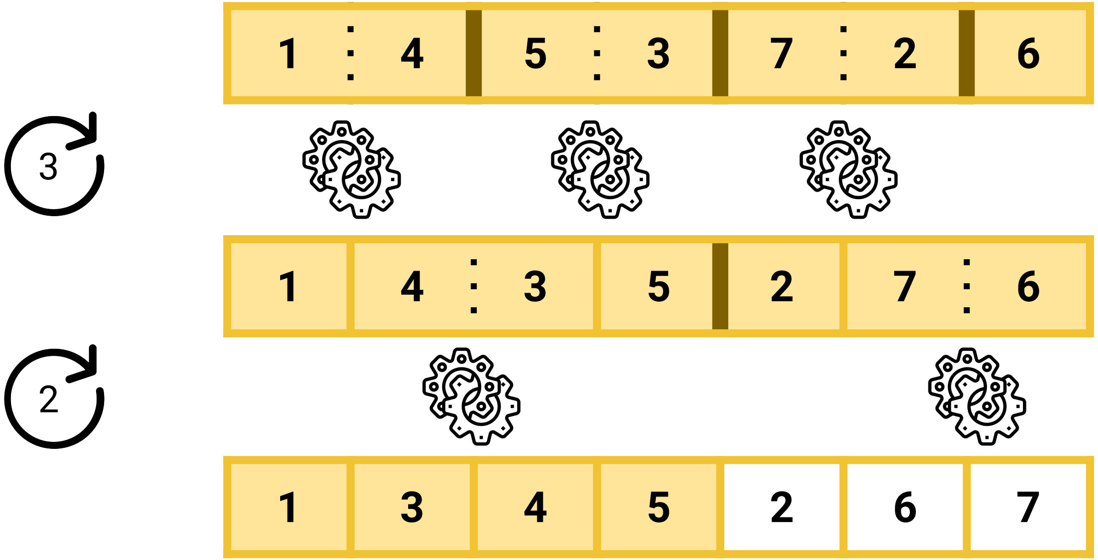
1	4	5	3	7	2	6

1	4	3	5	2	7	6

1	3	4	5

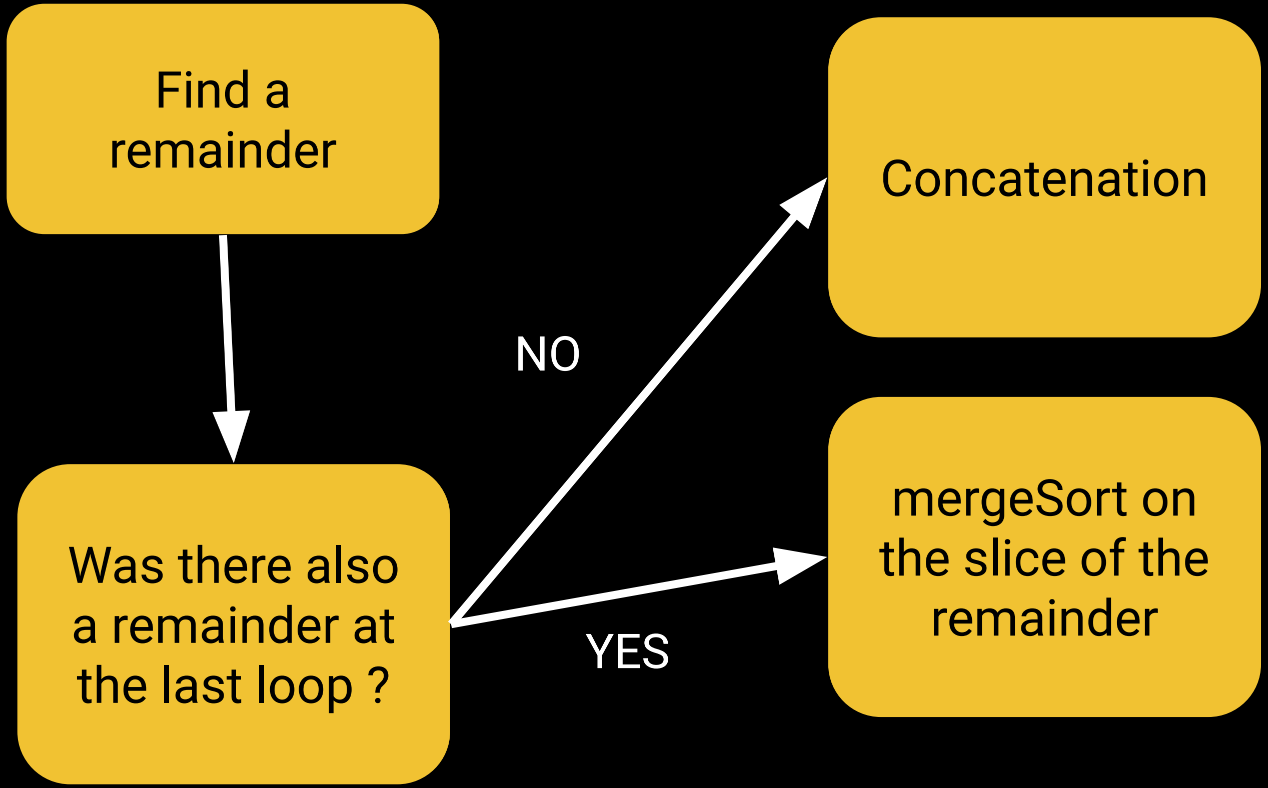
“MERGE SORT” ON GPU

slice_size = 4
number_of_slices = **1**



1	4	5	3	7	2	6
1	4	3	5	2	7	6
1	3	4	5	2	6	7

MANAGING THE REMAINDERS



Find a remainder

Was there also a remainder at the last loop ?

NO

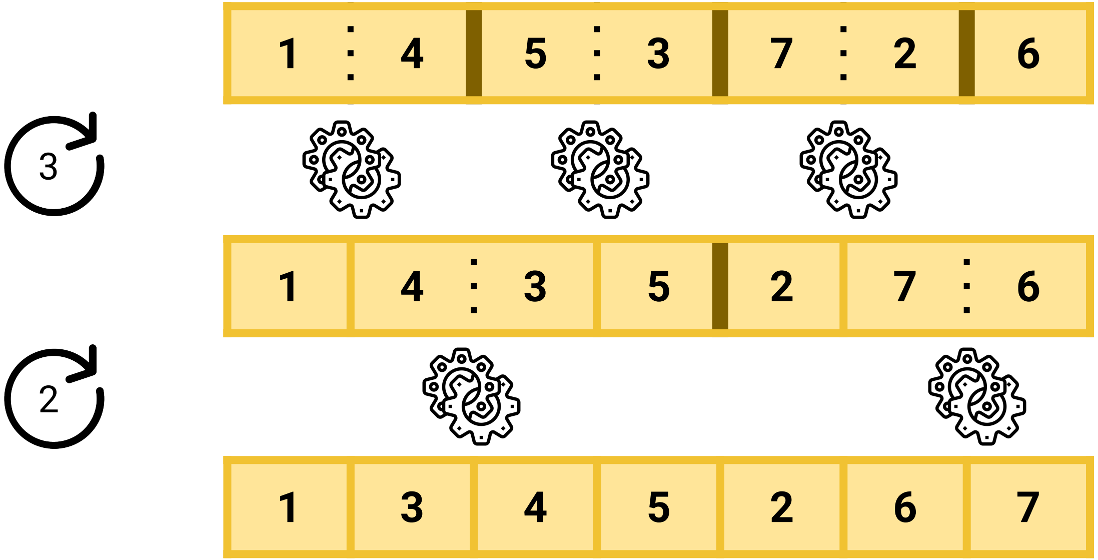
Concatenation

YES

mergeSort on the slice of the remainder

“MERGE SORT” ON GPU

slice_size = 8
number_of_slices =



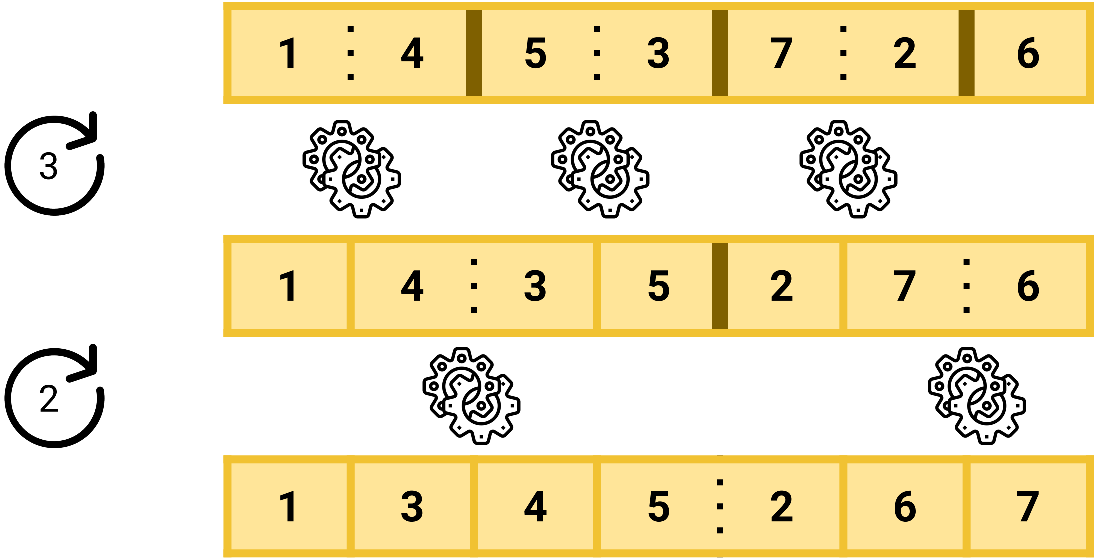
1 4 | 5 3 | 7 2 | 6

1 4 3 5 | 2 7 6

1 3 4 5 2 6 7

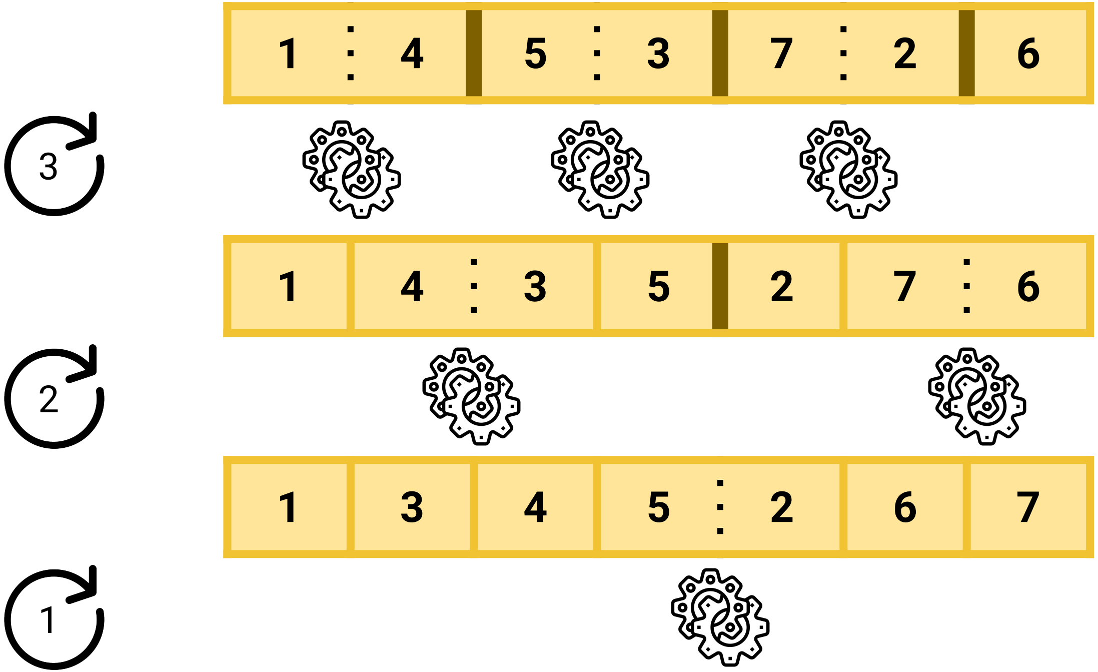
“MERGE SORT” ON GPU

slice_size = 8
number_of_slices = **1**



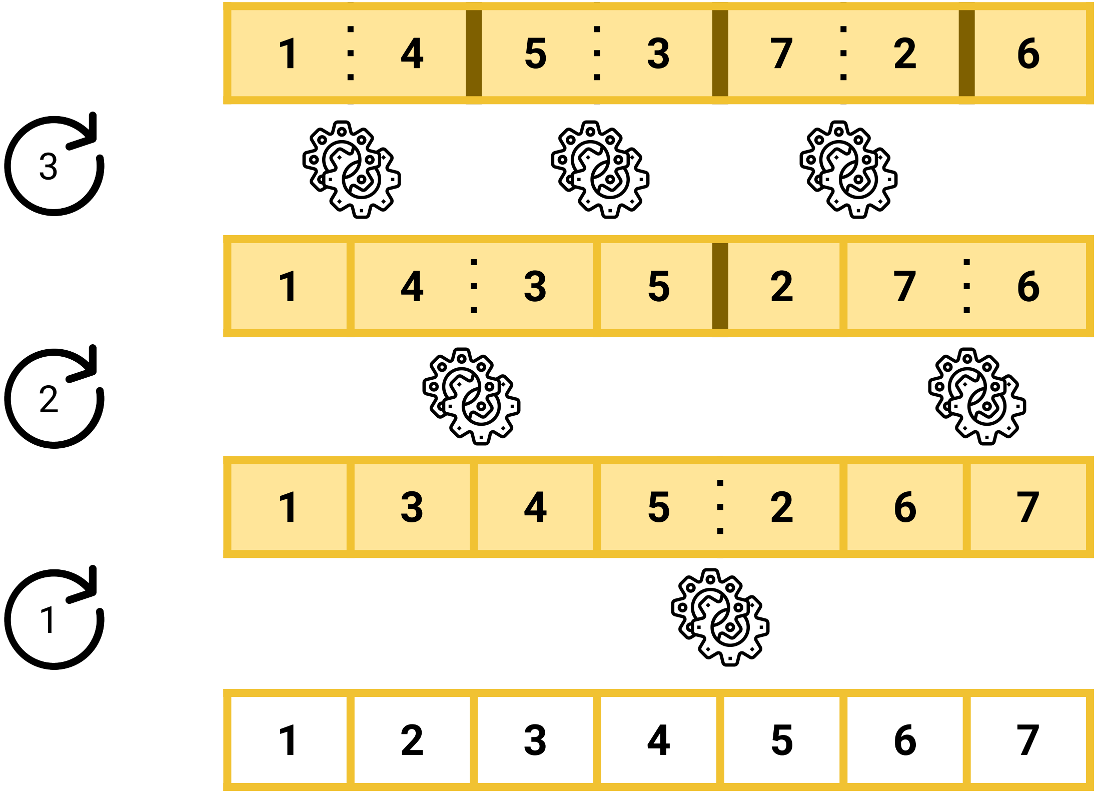
slice_size = 8
number_of_slices = **1**

“MERGE SORT” ON GPU



“MERGE SORT” ON GPU

slice_size = 8
number_of_slices = **1**



1 4 5 3 7 2 6

3

1 4 3 5 2 7 6

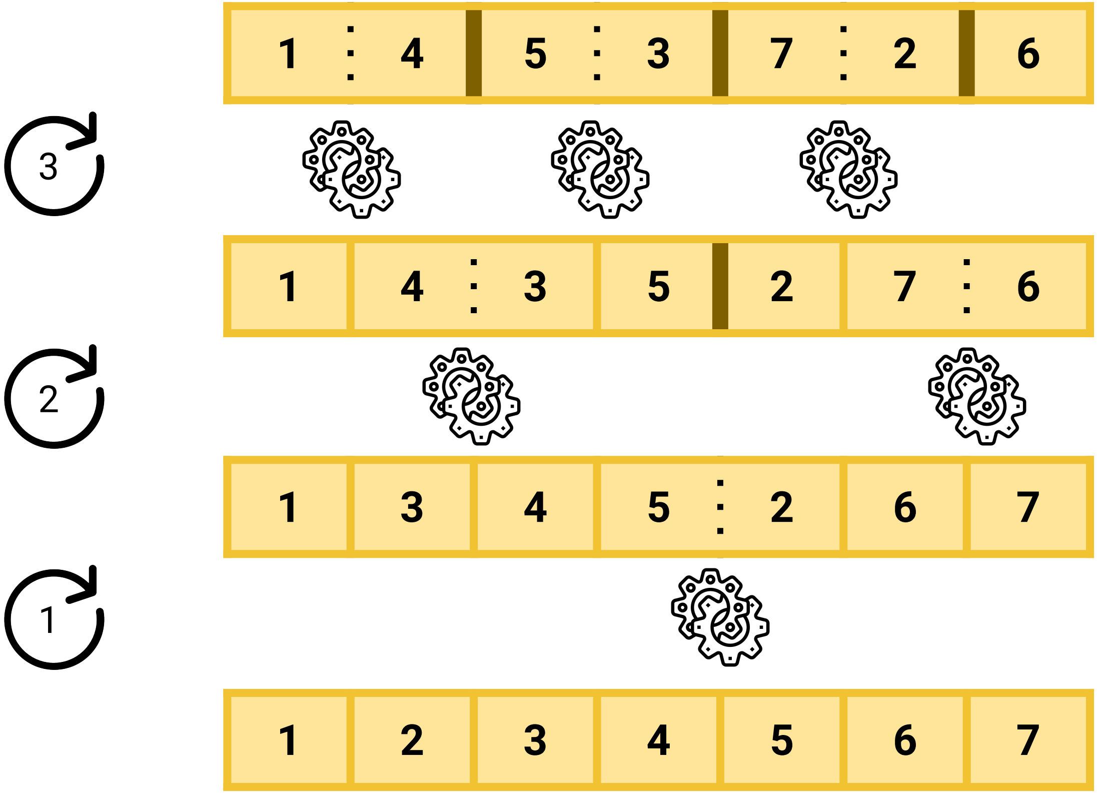
2

1 3 4 5 2 6 7

1

1 2 3 4 5 6 7

“MERGE SORT” ON GPU



“MERGE SORT” ON GPU

```
slice_size = 1
number_of_slices = 1
reste_prec_size = 0
reste_size = 0
```

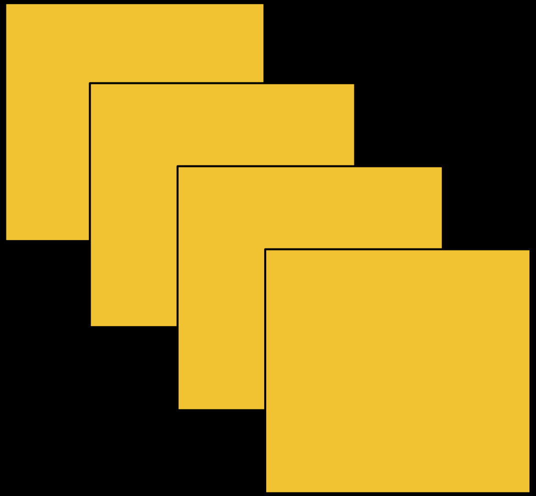
```
while (number_of_slices > 0)
    slice_size = 2*slice_size
    number_of_slices = size_M/slice_size
    reste_prec_size = reste_size
    reste_size = size_M%slice_size

    for (i=0, i<number_of_slices, i++)
        sort(M, slice_size, i)

    if (reste_prec_size!=0 && reste_size!=0)

        sort(M, reste_size + reste_prec_size, number_of_slices)
```

04

SORT WITH BATCHES

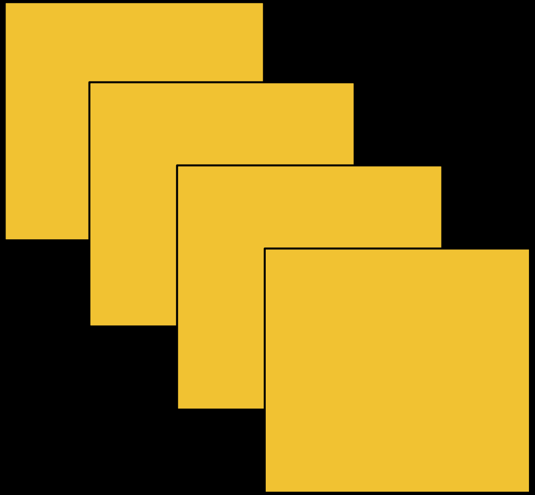
DEFINITION : BATCH

Sub array with at the most 1024 elements



DEFINITION : BATCH

Sub array with at the most 1024 elements
(ie the maximum number of threads on one block)



NOTATIONS

NOTATIONS

mergeSmallBatch_k(A,B)

NOTATIONS

//

NOTATIONS

//

In parallel

SORT ON ONE BATCH

8	4	12	… …	1	5	7

SORT ON ONE BATCH

t1	t2	t3	... ...	t1022	t1023	t1024

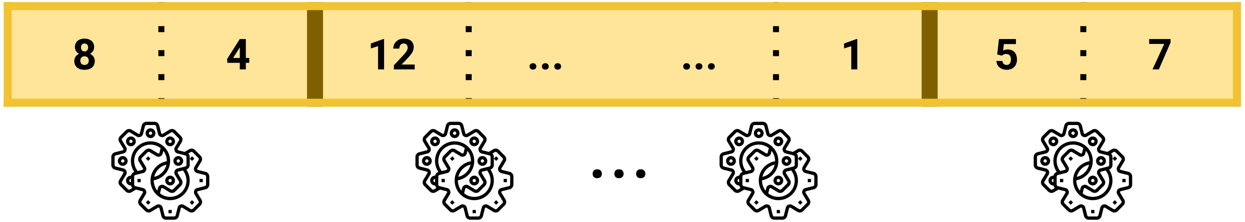
SORT ON ONE BATCH

8	4	12	… …	1	5	7

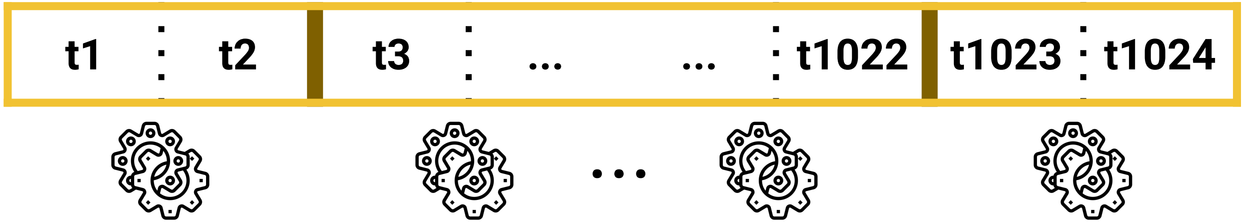
SORT ON ONE BATCH

8	4	12	... ...	1	5	7

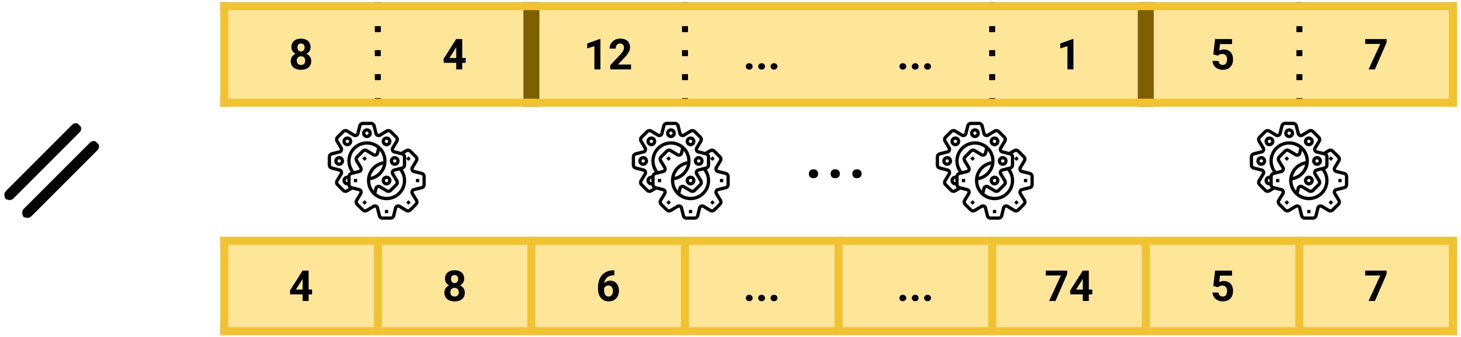
SORT ON ONE BATCH



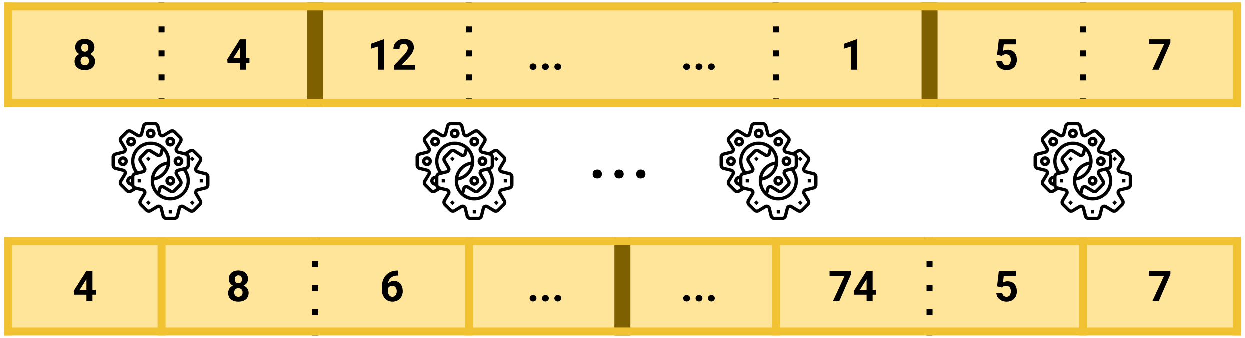
SORT ON ONE BATCH



SORT ON ONE BATCH



SORT ON ONE **BATCH**

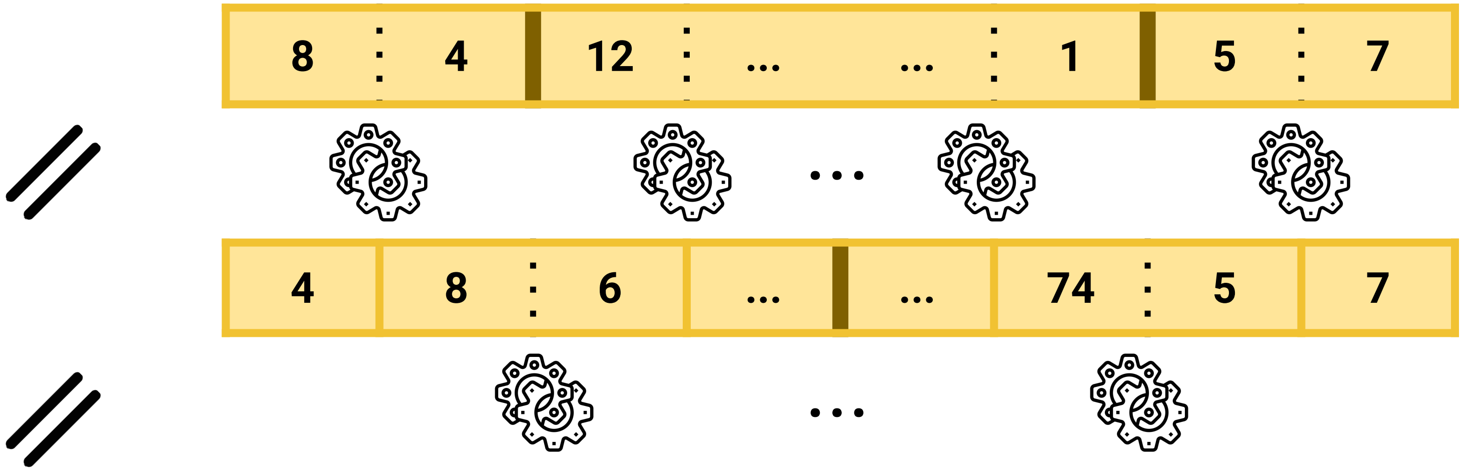


8 4 12 … … 1 5 7

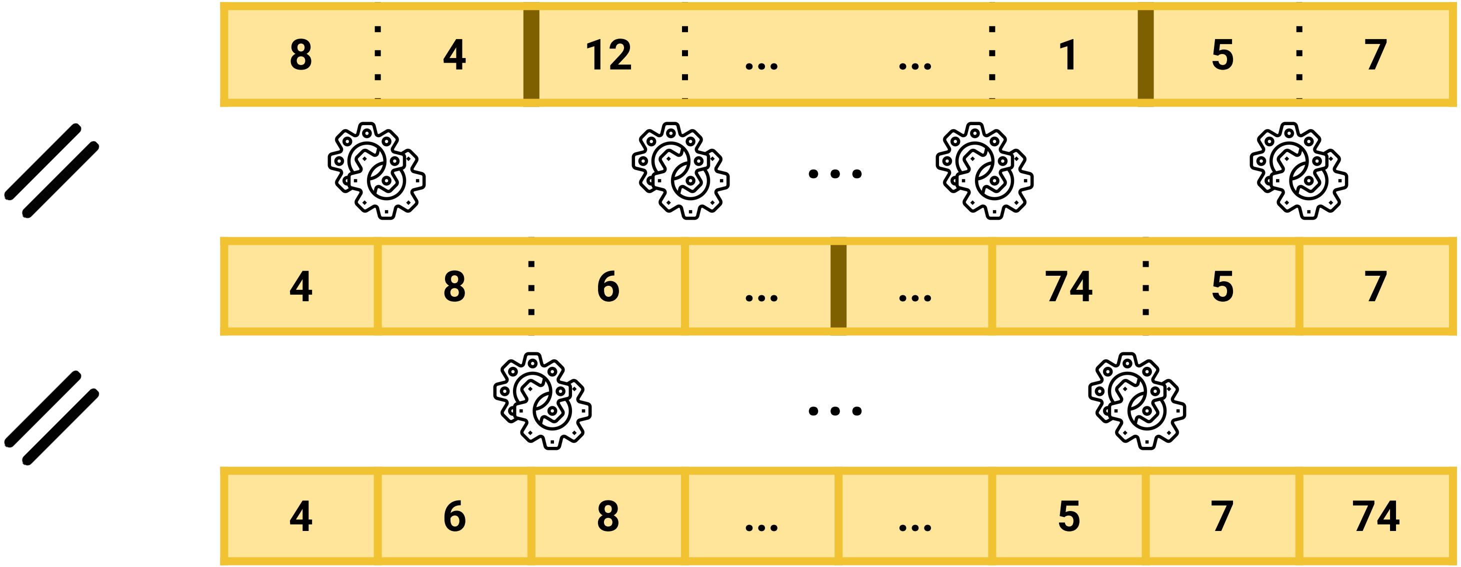
…

4 8 6 … … 74 5 7

SORT ON ONE BATCH



SORT ON ONE BATCH



8 4 12 … … 1 5 7

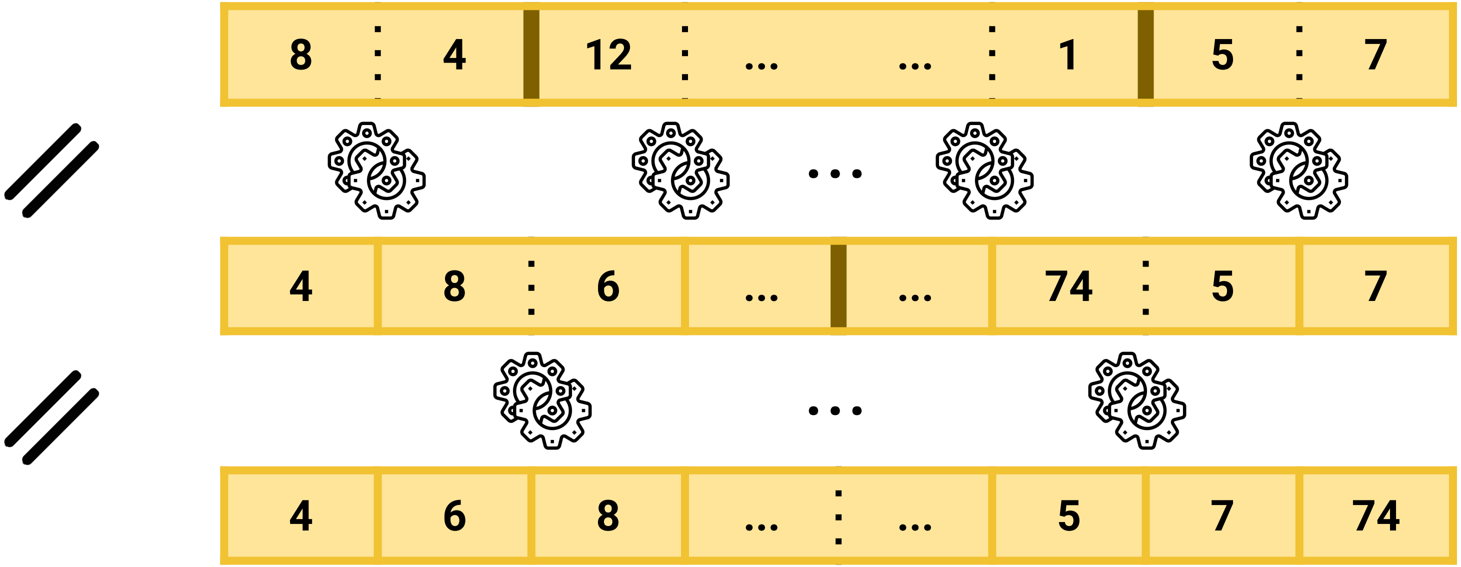
…

4 8 6 … … 74 5 7

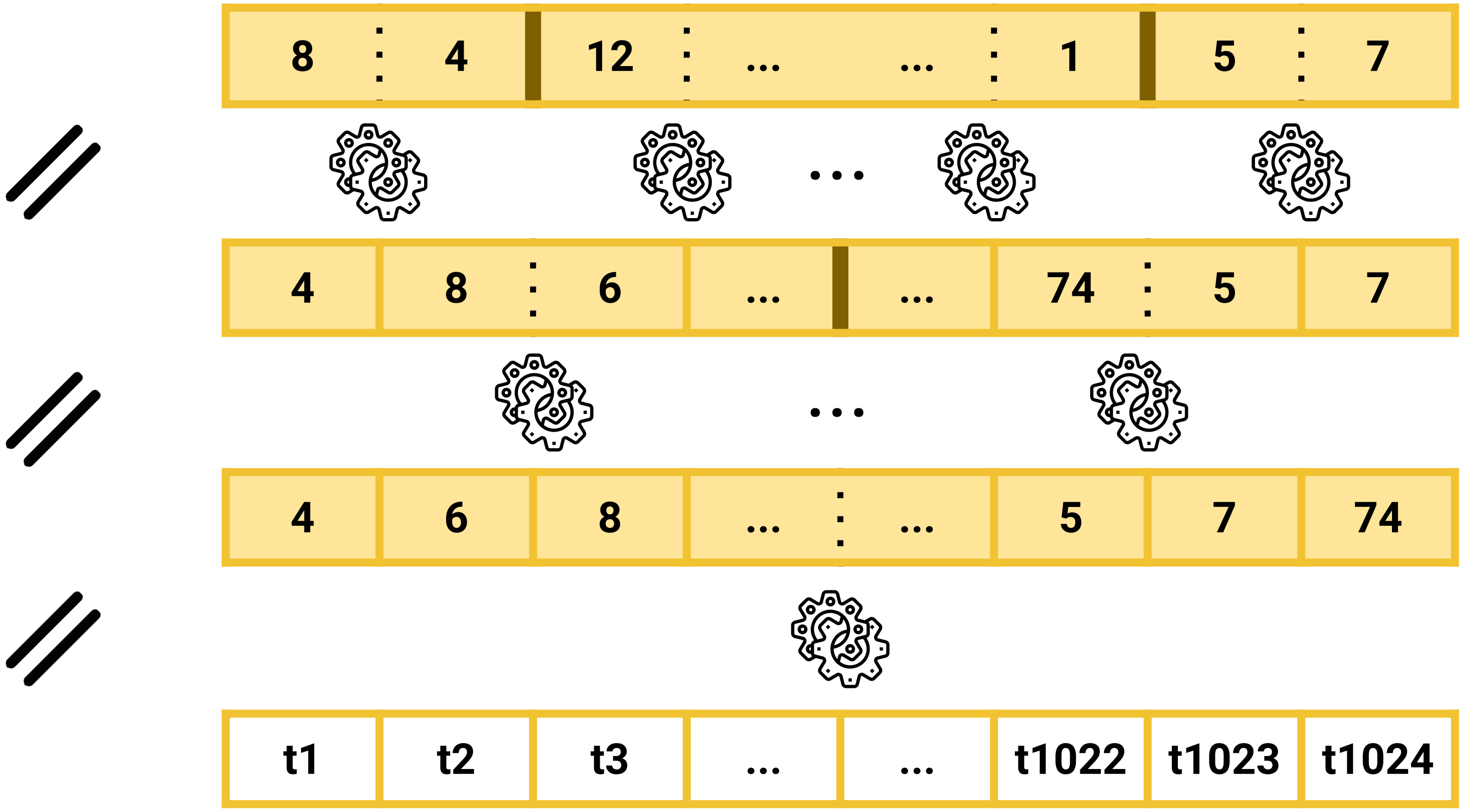
…

4 6 8 … … 5 7 74

SORT ON ONE BATCH



SORT ON ONE BATCH



8 4 12 … … 1 5 7

…

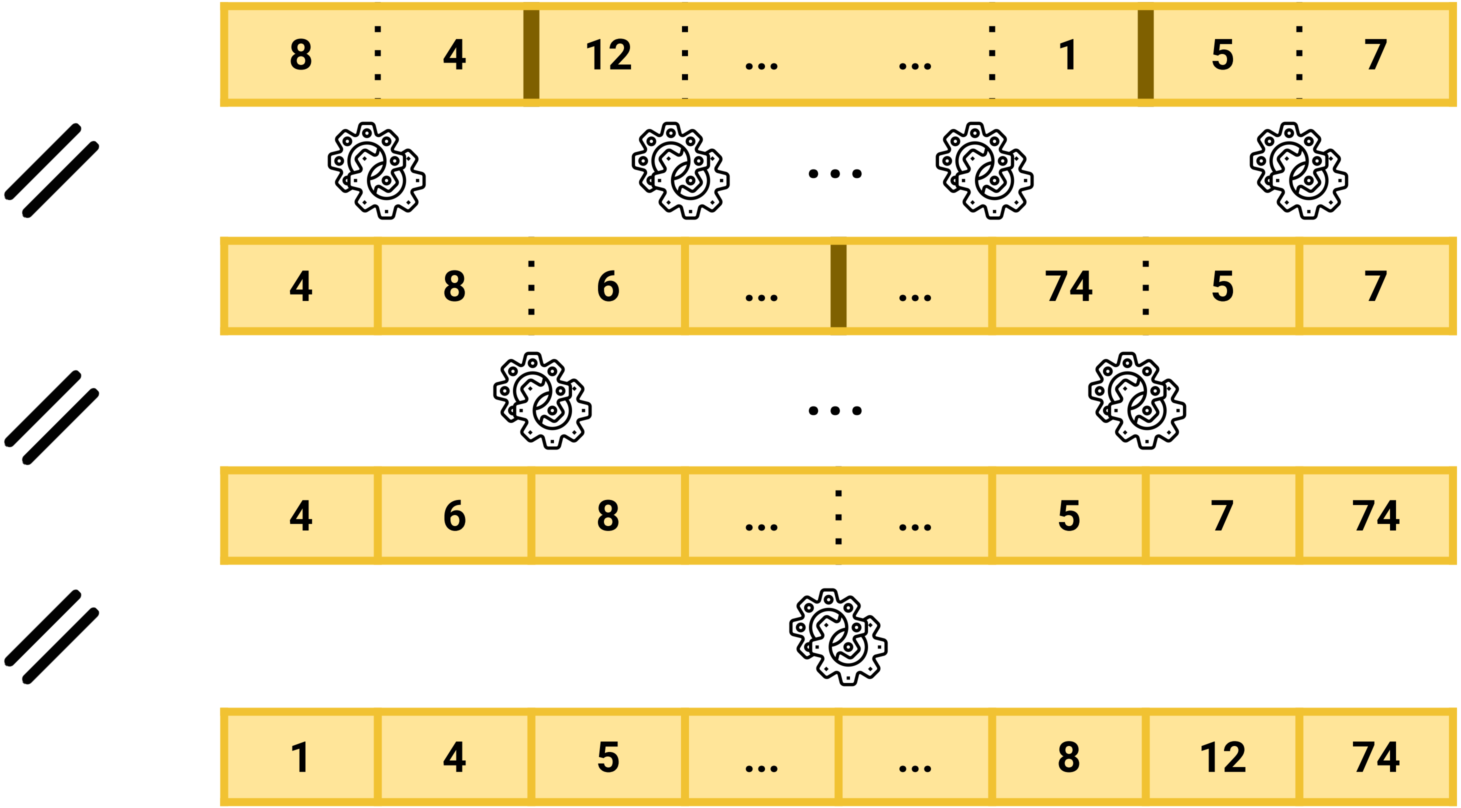
4 8 6 … … 74 5 7

…

4 6 8 … … 5 7 74

t1 t2 t3 … … t1022 t1023 t1024

SORT ON ONE BATCH



8 4 12 … … 1 5 7

…

4 8 6 … … 74 5 7

…

4 6 8 … … 5 7 74

1 4 5 … … 8 12 74

SMALL SORT

M

… … …	4	… … …	26	… … …	17	… … …	2

SMALL SORT

M

… … …	4	… … …	26	… … …	17	… … …	2

Batches of 1024 elements

SMALL SORT

M

… … …	4	… … …	26	… … …	17	… … …	2

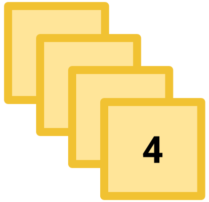
Batches of 1024 elements

SMALL SORT

M

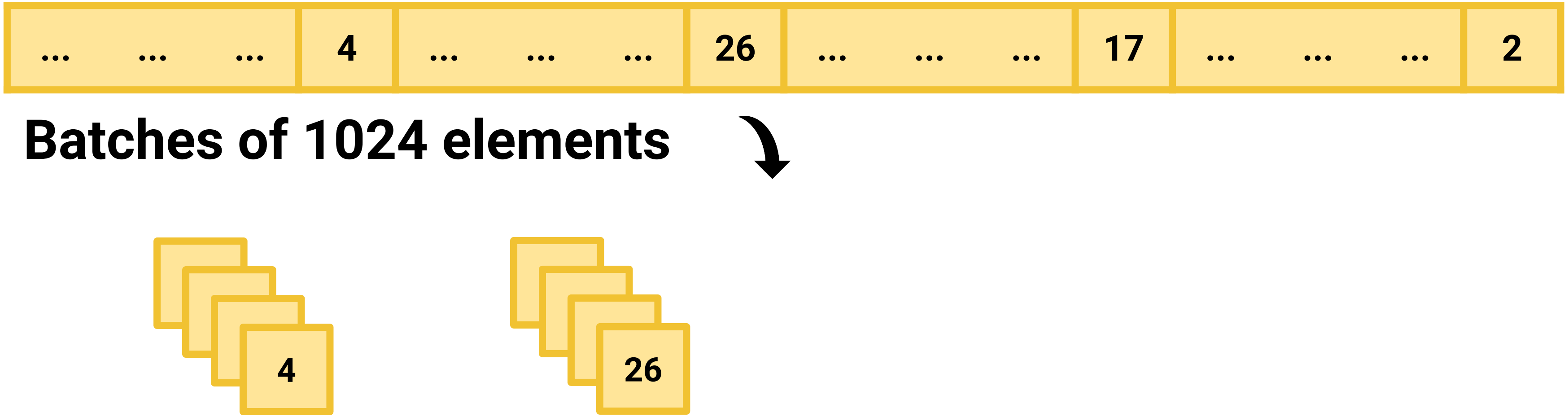
… … …	4	… … …	26	… … …	17	… … …	2

Batches of 1024 elements



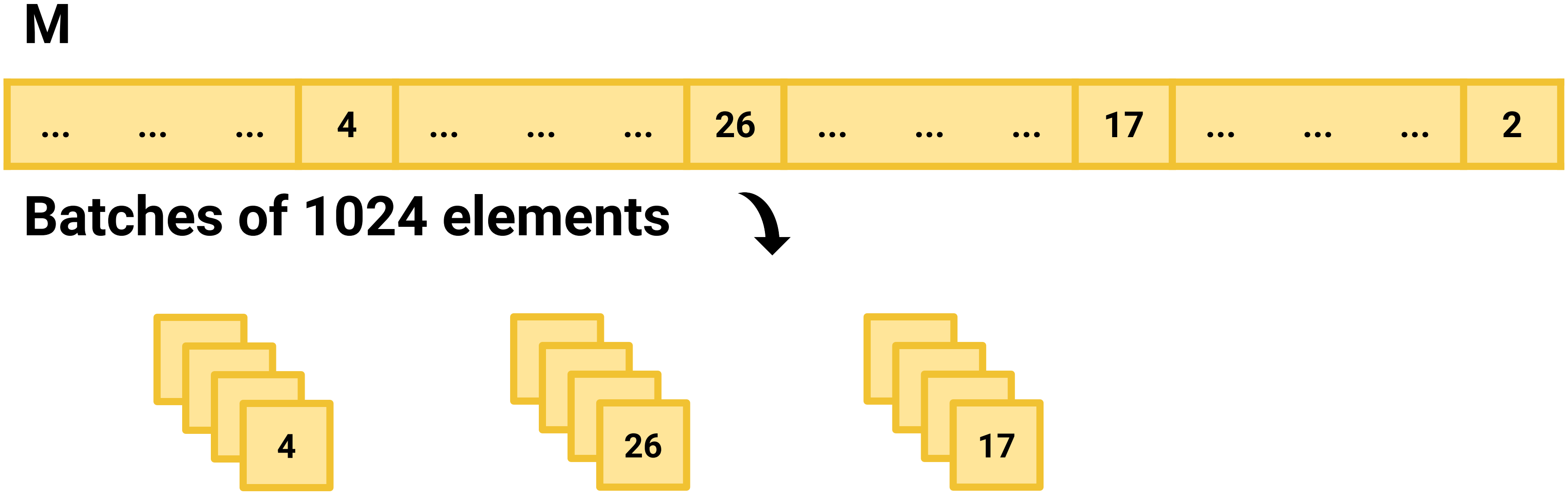
SMALL SORT

M



Batches of 1024 elements

SMALL SORT

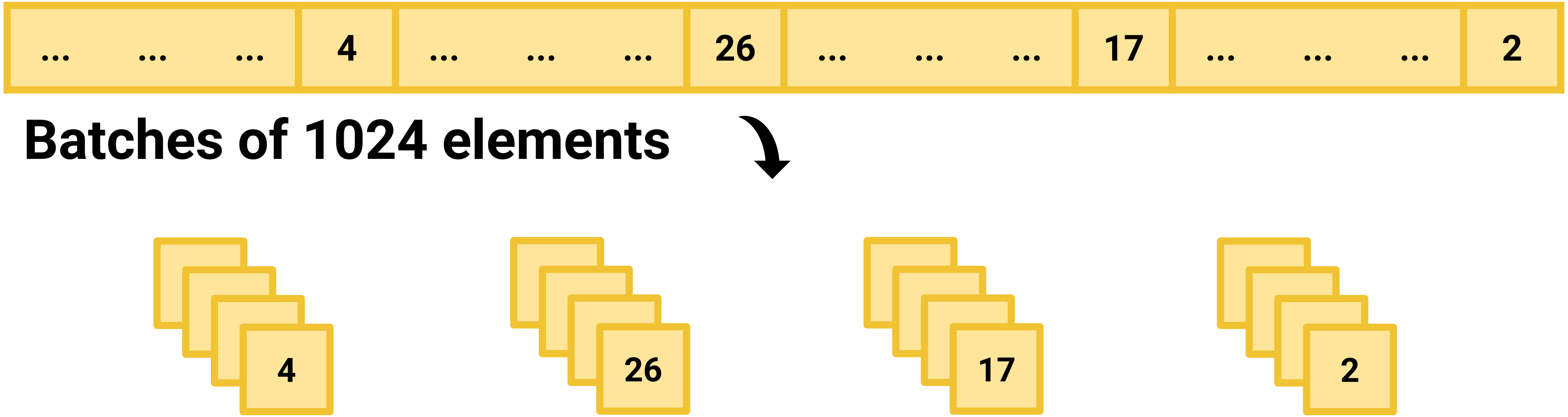


M

Batches of 1024 elements

SMALL SORT

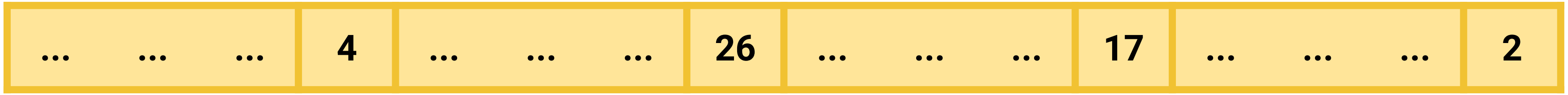
M



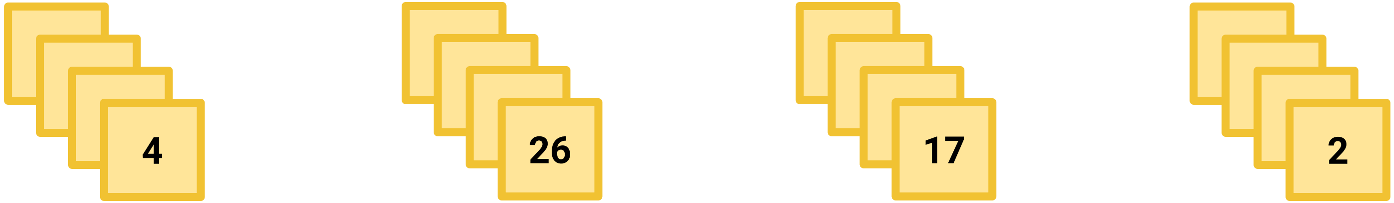
Batches of 1024 elements

SMALL SORT

M

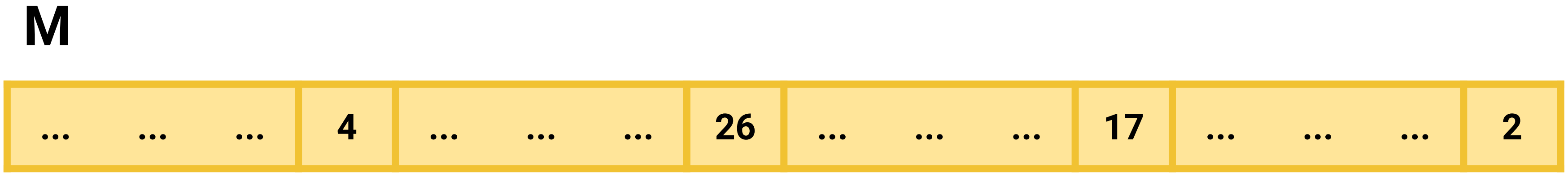


Batches of 1024 elements

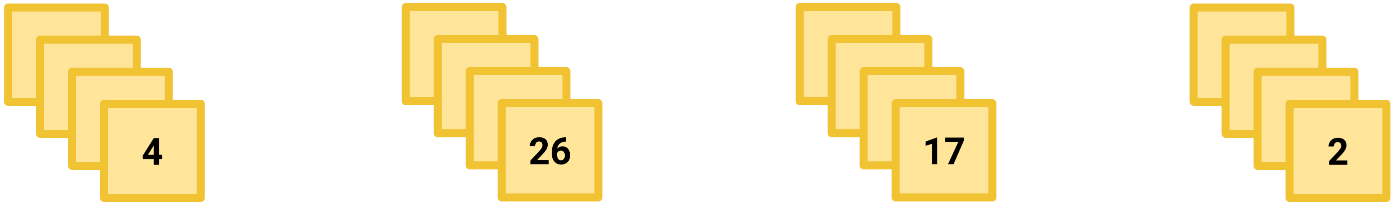


We sort each batch in parallel

SMALL SORT



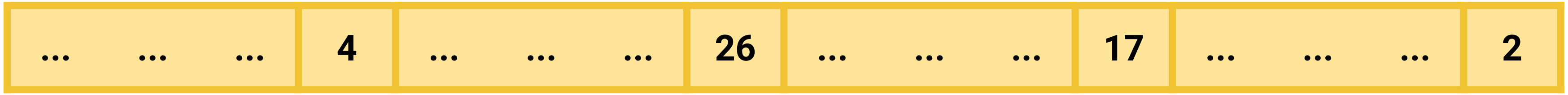
Batches of 1024 elements



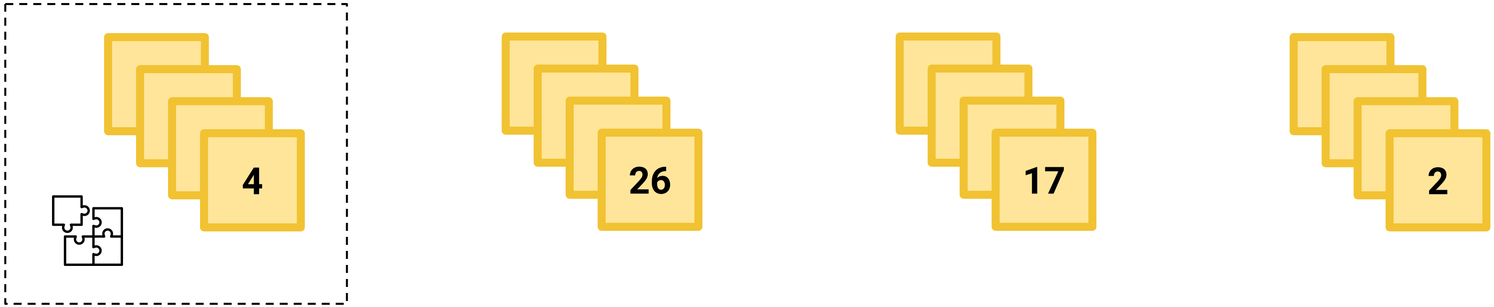
We sort each batch in parallel on one block

SMALL SORT

M



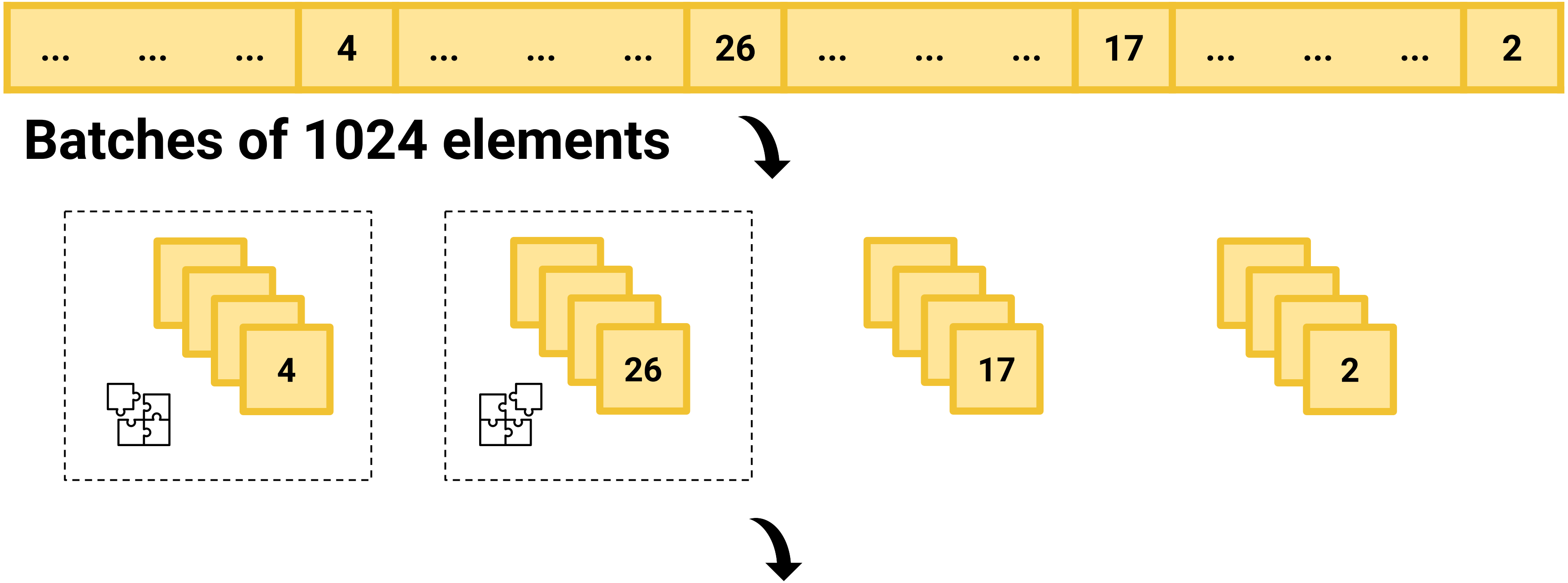
Batches of 1024 elements



We sort each batch in parallel on one block

SMALL SORT

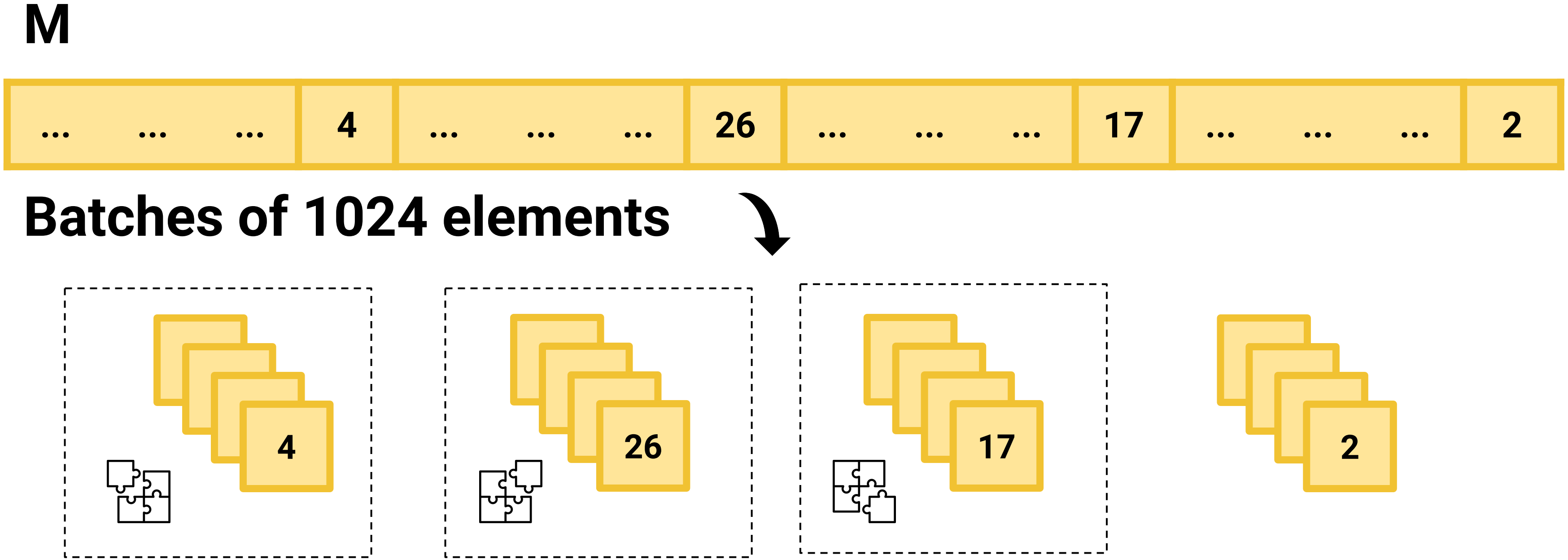
M



Batches of 1024 elements

We sort each batch in parallel on one block

SMALL SORT



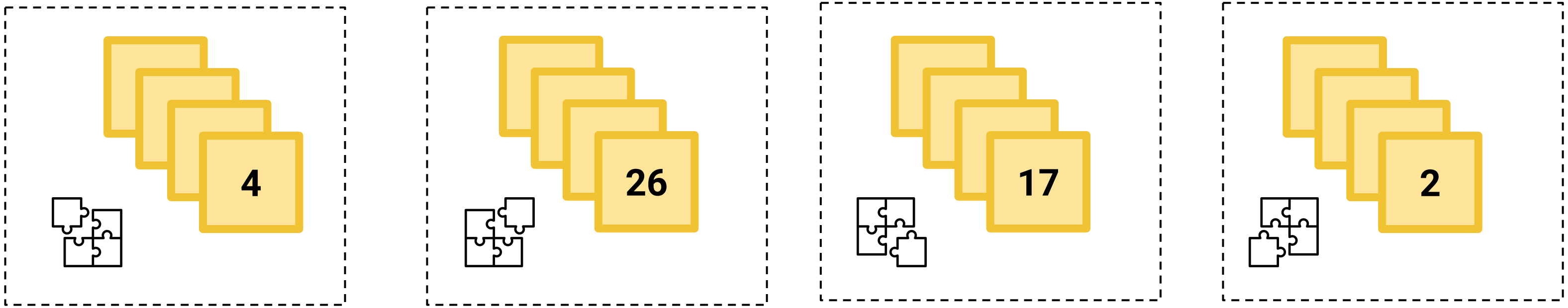
We sort each batch in parallel on one block

SMALL SORT

M

…	…	…	4	…	…	…	26	…	…	…	17	…	…	…	2

Batches of 1024 elements



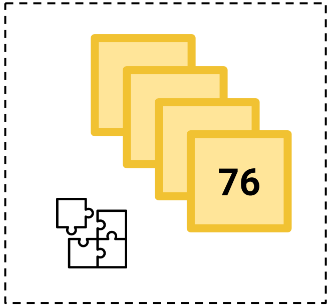
We sort each batch in parallel on one block

BIG SORT

Batches of 1024 sorted elements

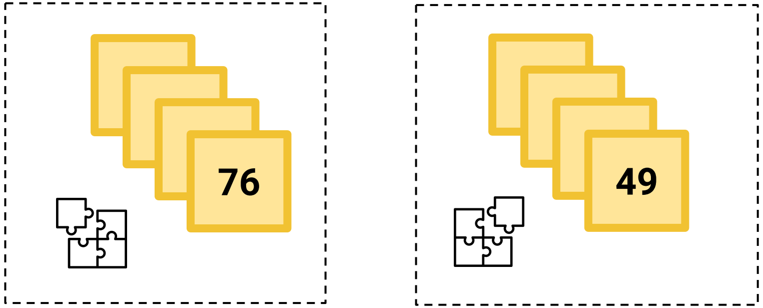
BIG SORT

Batches of 1024 sorted elements



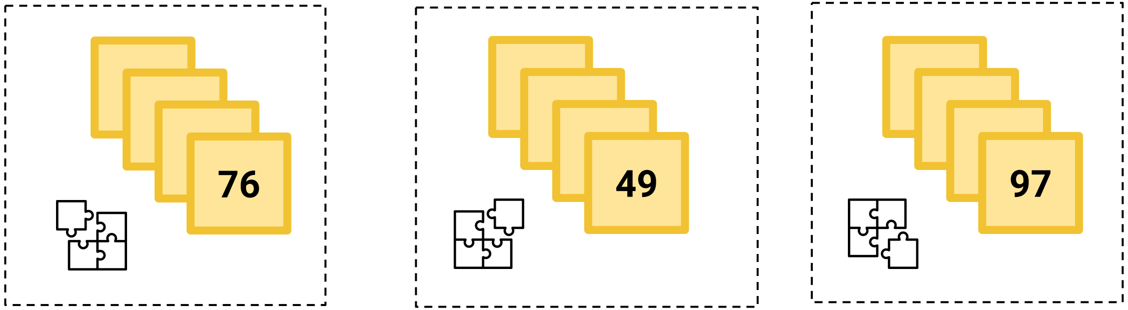
BIG SORT

Batches of 1024 sorted elements



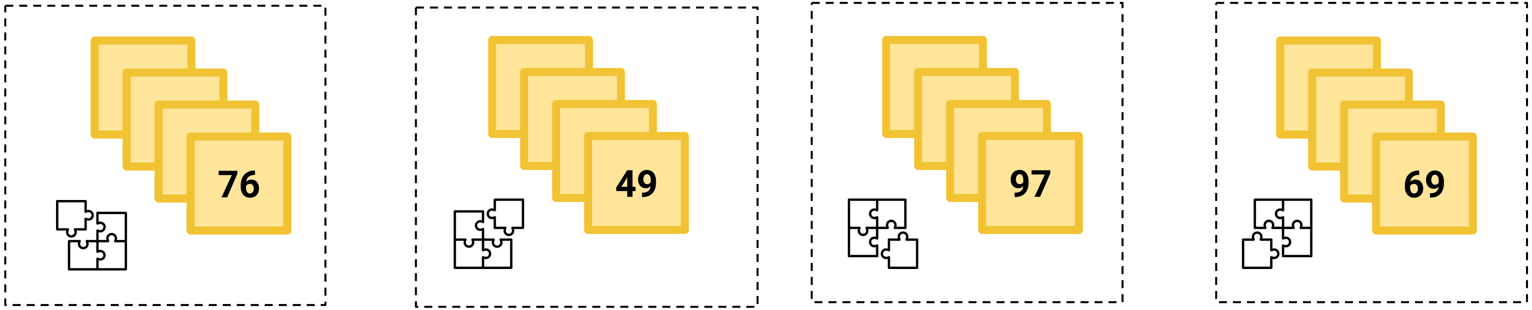
BIG SORT

Batches of 1024 sorted elements



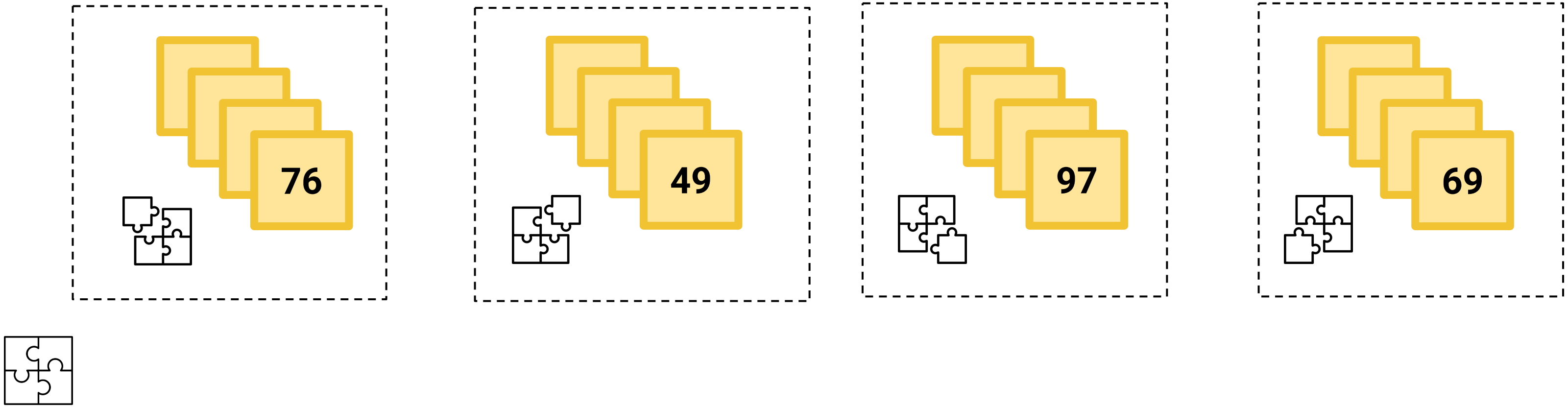
BIG SORT

Batches of 1024 sorted elements



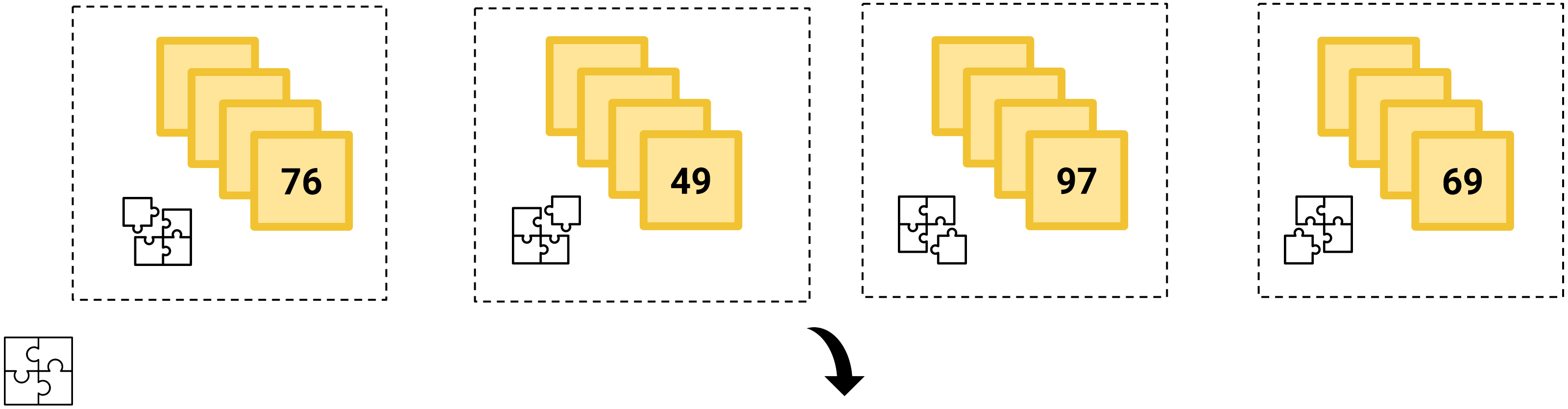
BIG SORT

Batches of 1024 sorted elements



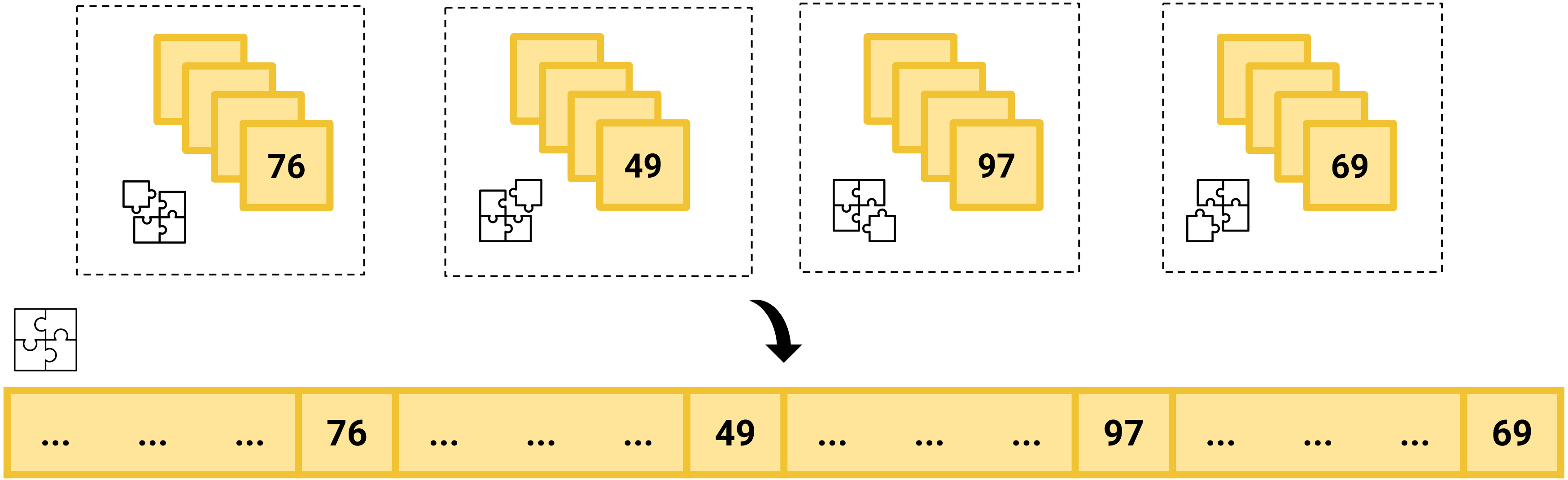
BIG SORT

Batches of 1024 sorted elements



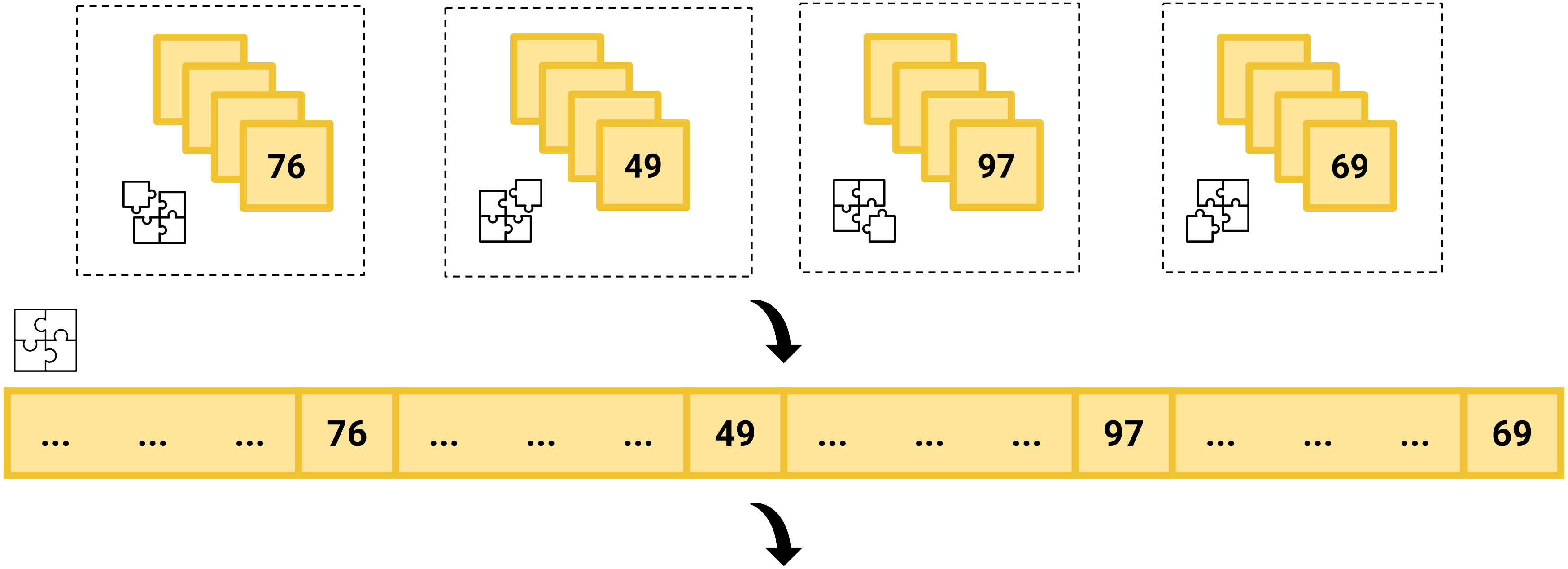
BIG SORT

Batches of 1024 sorted elements



BIG SORT

Batches of 1024 sorted elements



We sort the new reconstitued array

NOTATIONS

NOTATIONS

pathBig_k(A,B)

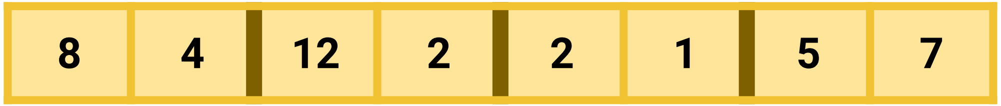
+

mergeBig_k(A,B)

GPU SORT

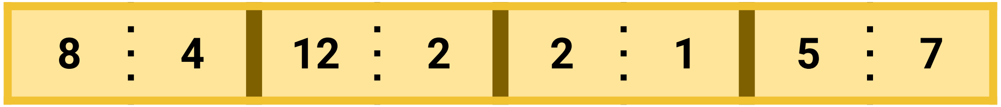
8	4	12	2	2	1	5	7

GPU SORT



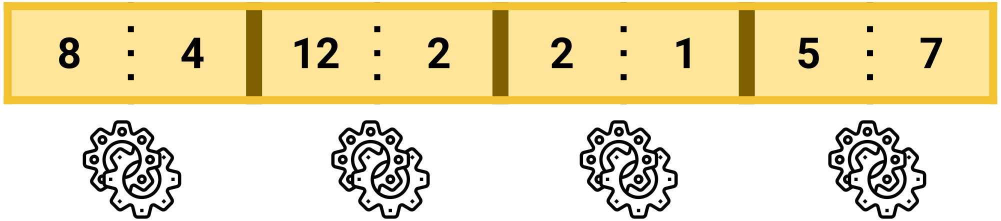
8	4	12	2	2	1	5	7

GPU SORT



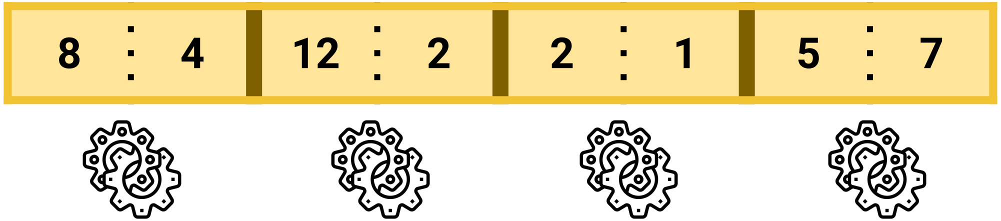
8	4	12	2	2	1	5	7

GPU SORT



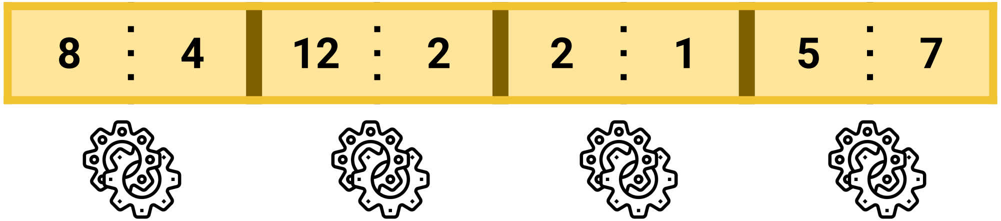
8 : 4 | 12 : 2 | 2 : 1 | 5 : 7

GPU SORT WITH STREAM



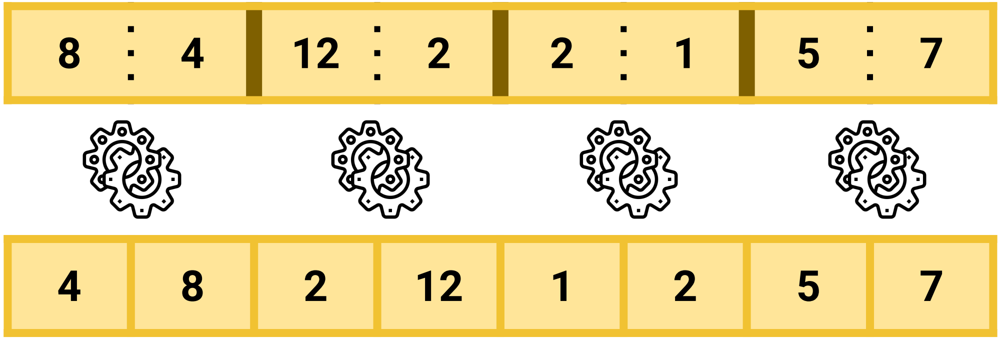
8	4	12	2	2	1	5	7

GPU SORT WITH STREAM

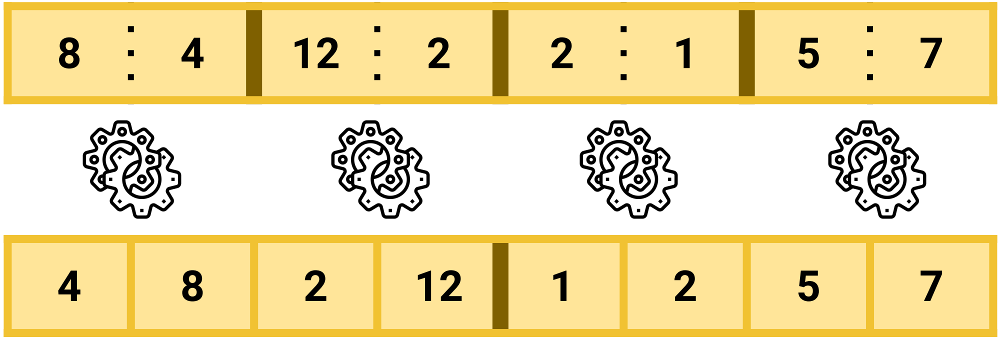


8	4	12	2	2	1	5	7

GPU SORT WITH STREAM



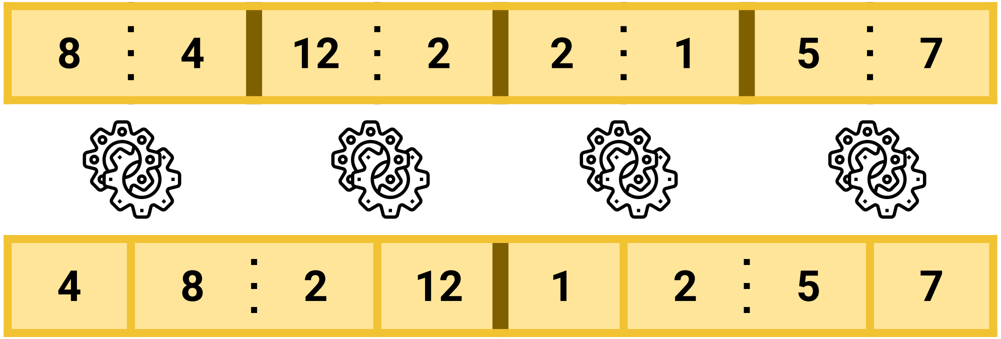
GPU SORT WITH STREAM



8	4	12	2	2	1	5	7

4	8	2	12	1	2	5	7

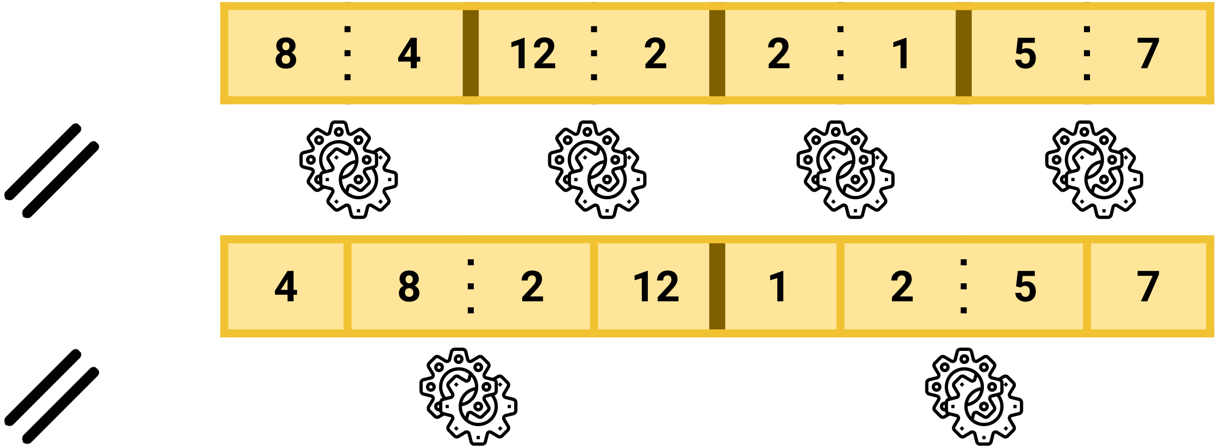
GPU SORT WITH STREAM



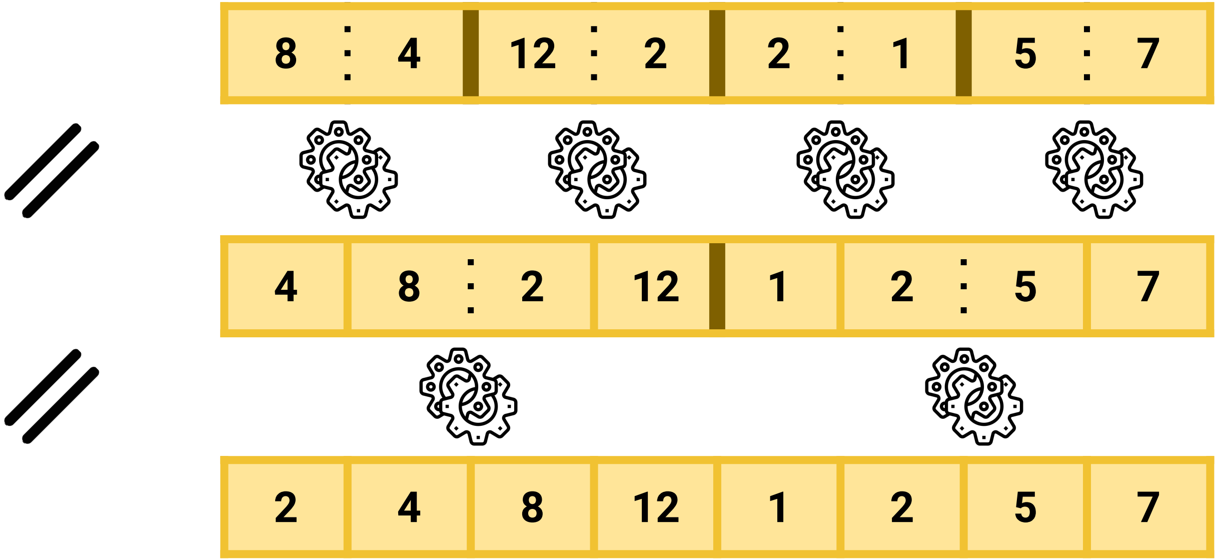
8 4 | 12 2 | 2 1 | 5 7

4 8 2 12 | 1 2 5 7

GPU SORT WITH STREAM



GPU SORT WITH STREAM

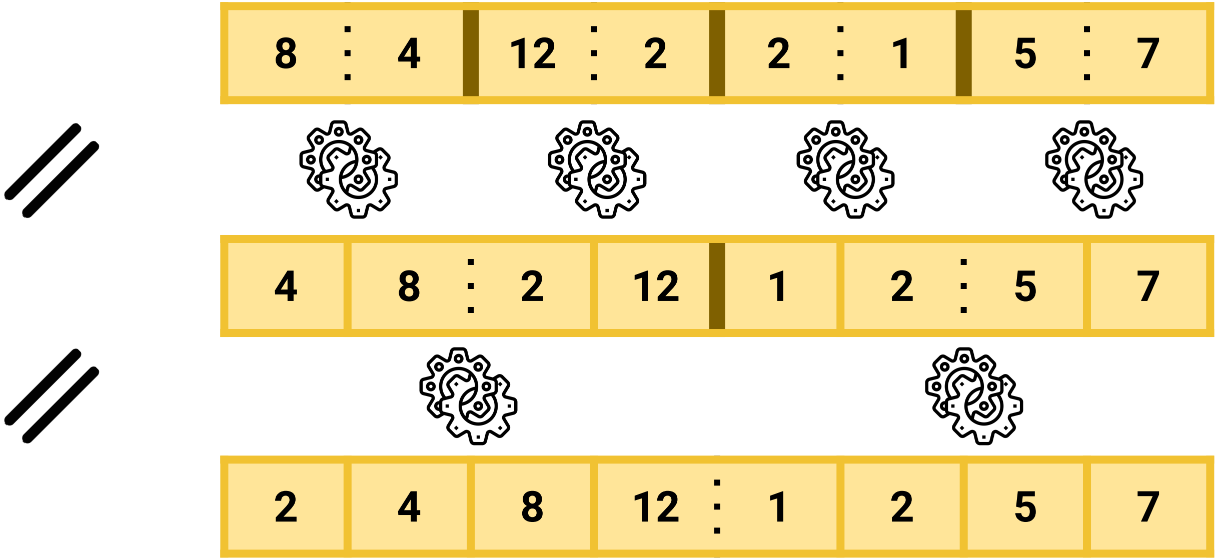


8 : 4 | 12 : 2 | 2 : 1 | 5 : 7

4 | 8 : 2 | 12 | 1 | 2 : 5 | 7

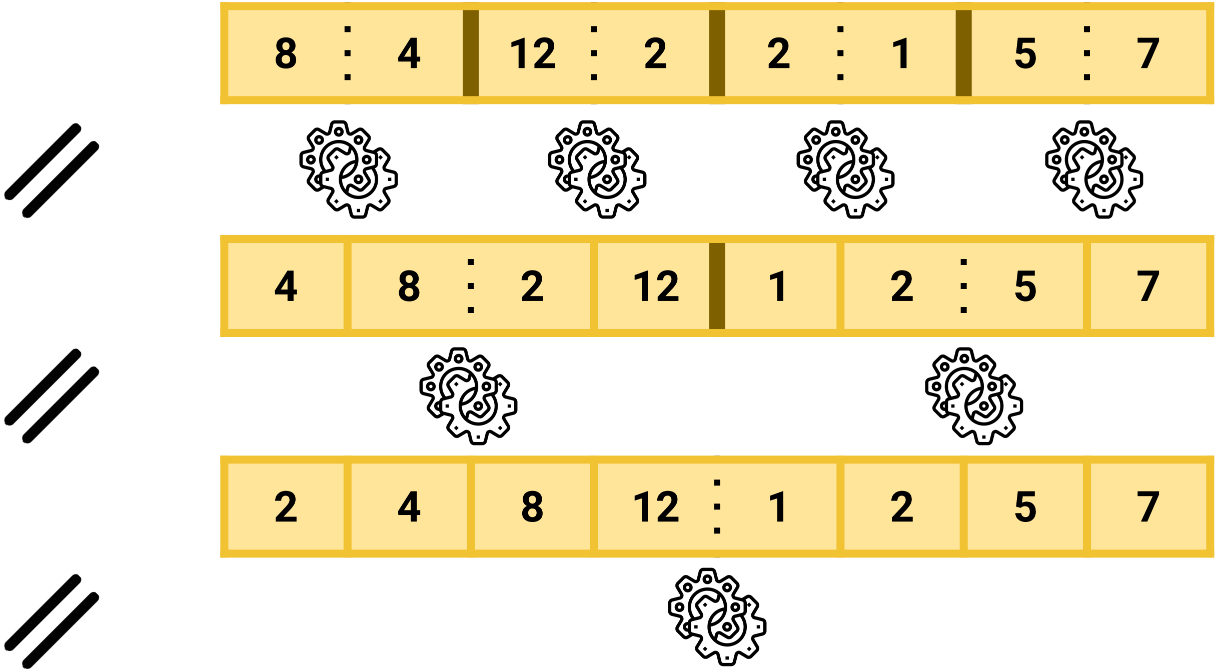
2 | 4 | 8 | 12 | 1 | 2 | 5 | 7

GPU SORT WITH STREAM

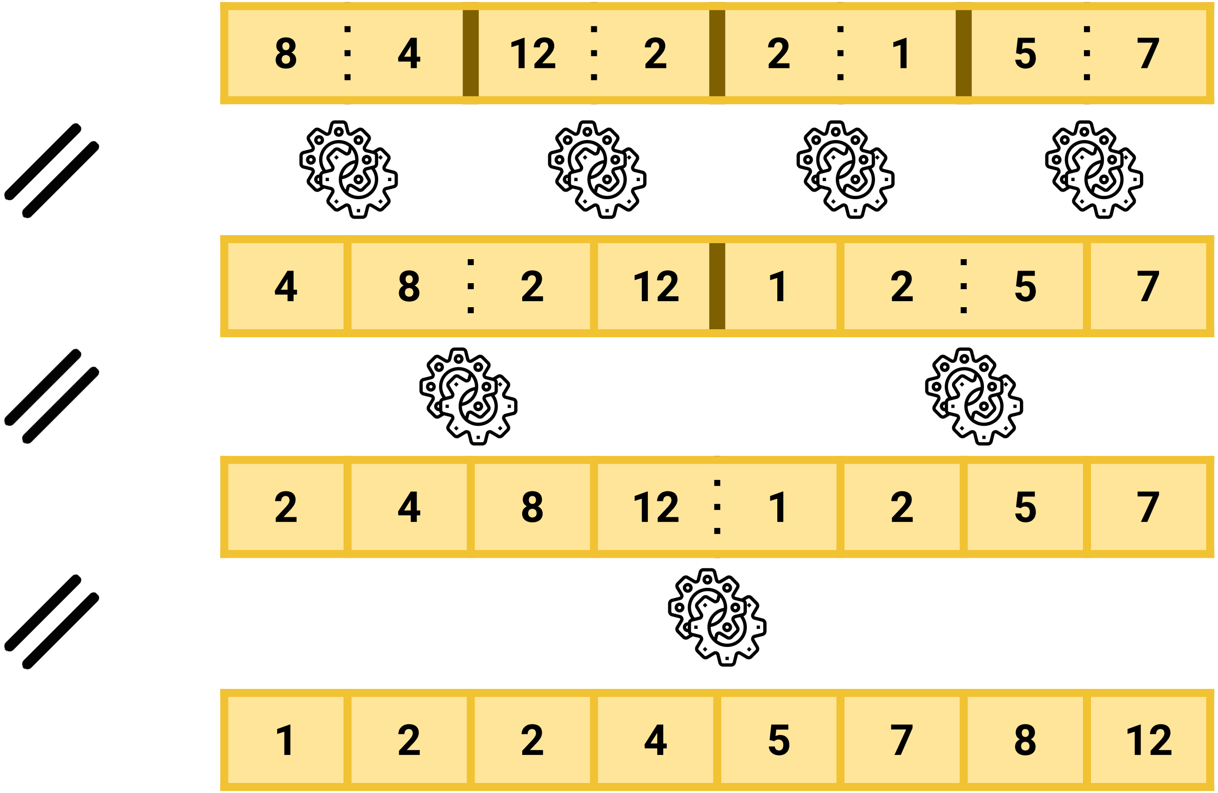


8	4	12	2	2	1	5	7
4	8	2	12	1	2	5	7
2	4	8	12	1	2	5	7

GPU SORT WITH STREAM



GPU SORT WITH STREAM



8 : 4 | 12 : 2 | 2 : 1 | 5 : 7

4 | 8 : 2 | 12 | 1 | 2 : 5 | 7

2 | 4 | 8 | 12 : 1 | 2 | 5 | 7

1 | 2 | 2 | 4 | 5 | 7 | 8 | 12

SMALL SORT

mergeSmall_Batch_k

Batches on shared memory

Very Fast

Impossible if batch_size > 1024

SMALL SORT

mergeSmall_Batch_k

Batches on shared memory

Very Fast

Impossible if batch_size > 1024

BIG SORT

pathBig_k and mergeBig_k

Streams

Global Memory

OUR GITHUB :

https://github.com/saulnierS/main5_hpc.git