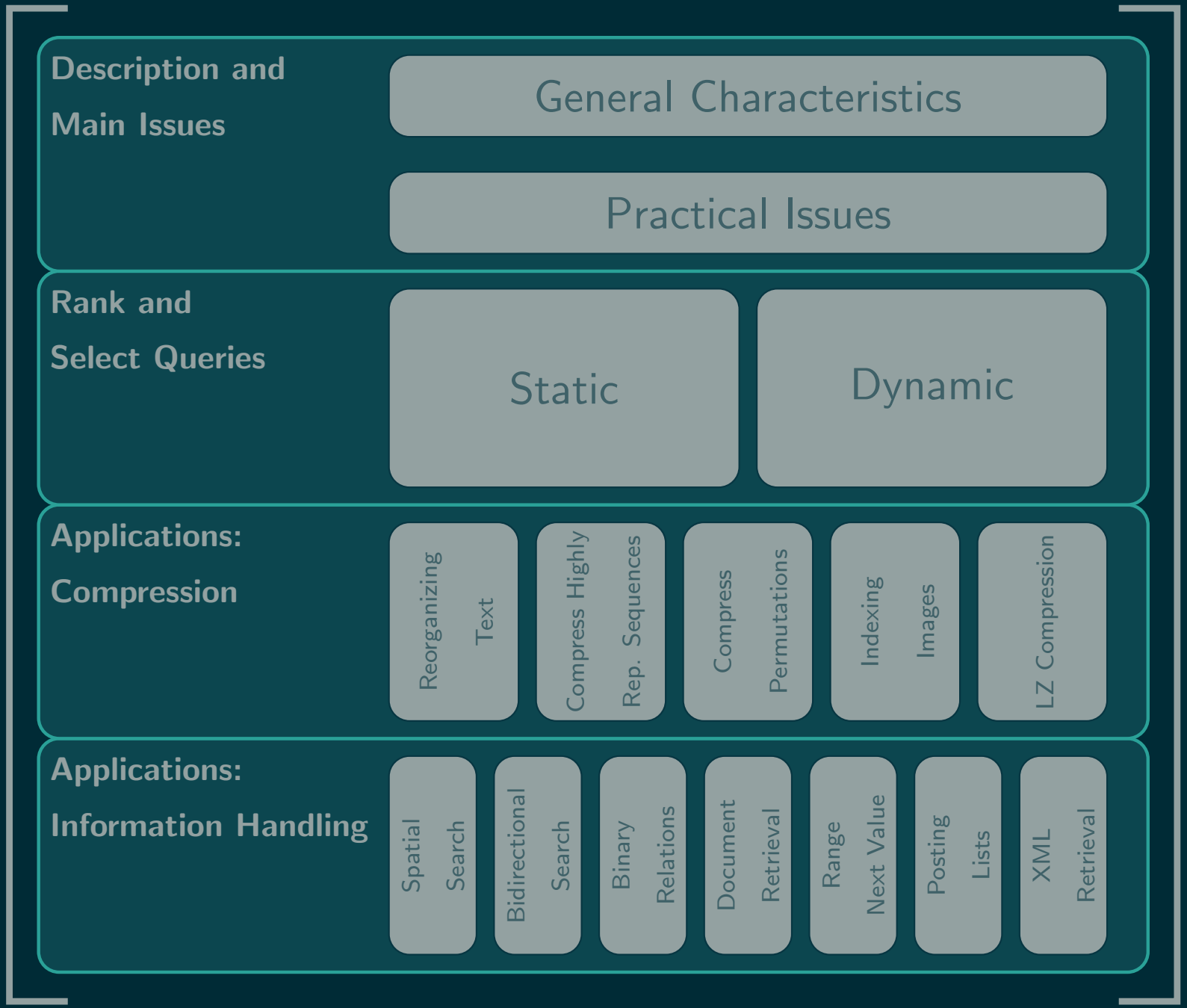
Parallel Wavelet Tree Construction

Simple, Fast and Lightweight

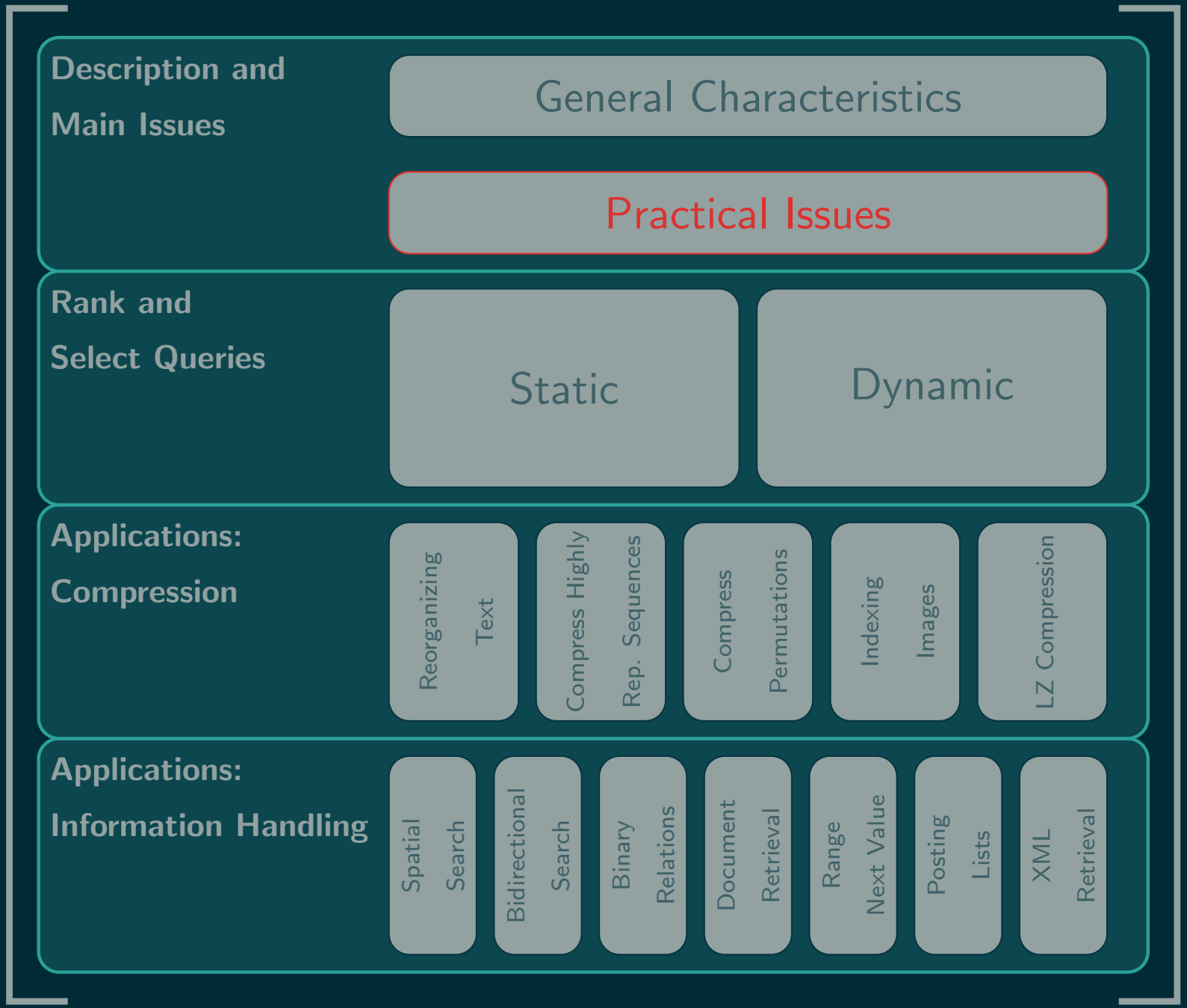
Johannes Fischer *Florian Kurpicz* Marvin Löbel

Wavelet Trees



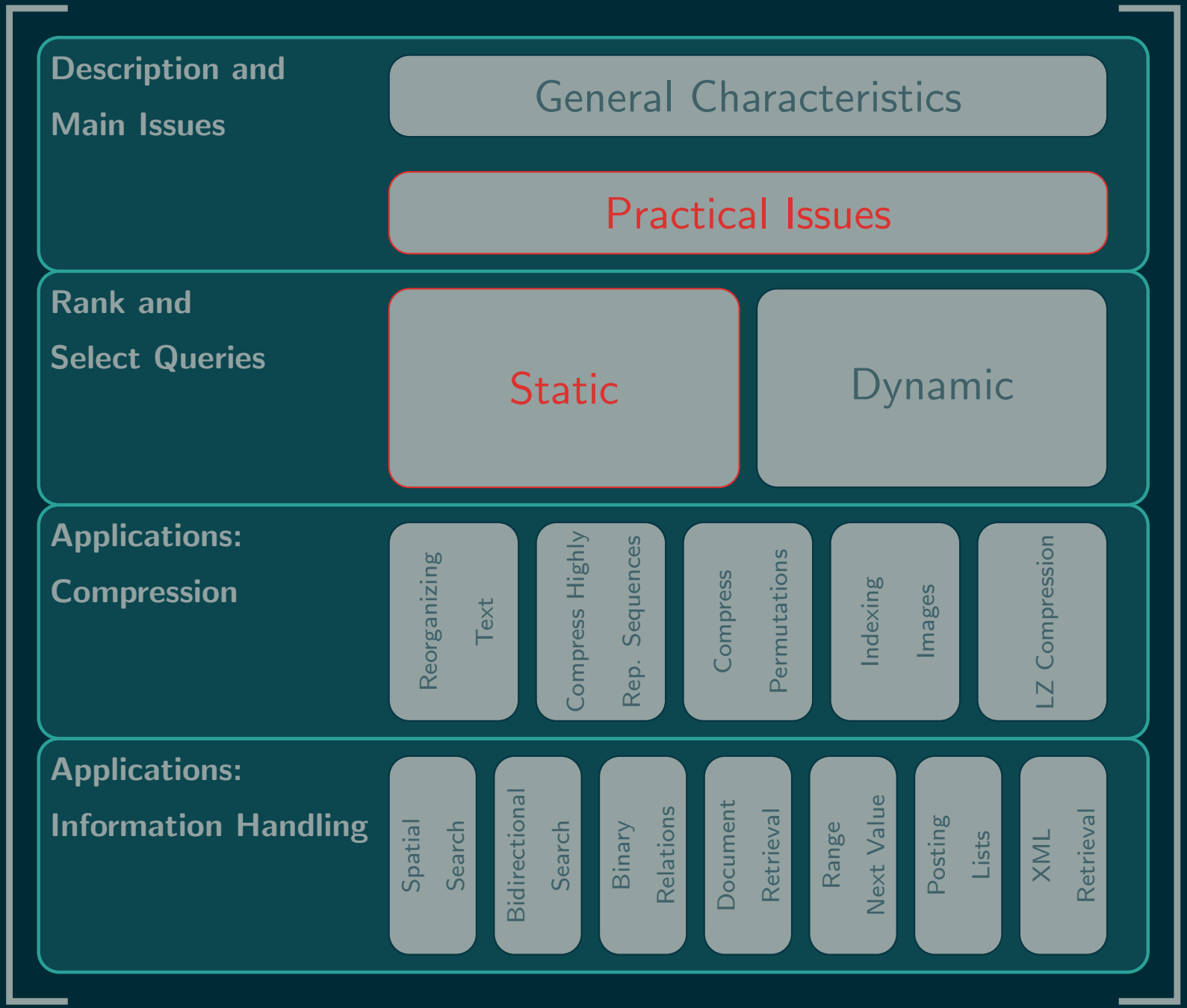
Based on: Christos Makris. *Wavelet trees: A survey*. Comput. Sci. Inf. Syst., 9(2):585–625, 2012.

Wavelet Trees



Based on: Christos Makris. *Wavelet trees: A survey*. Comput. Sci. Inf. Syst., 9(2):585–625, 2012.

Wavelet Trees



Based on: Christos Makris. *Wavelet trees: A survey*. Comput. Sci. Inf. Syst., 9(2):585–625, 2012.

Rank and Select

$\text{rank}_\alpha(i)$ # of αs before position i

$\text{select}_\alpha(j)$ position of j-th α

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

Rank and Select

$\text{rank}_\alpha(i)$ # of αs before position i
$\text{select}_\alpha(j)$ position of j-th α

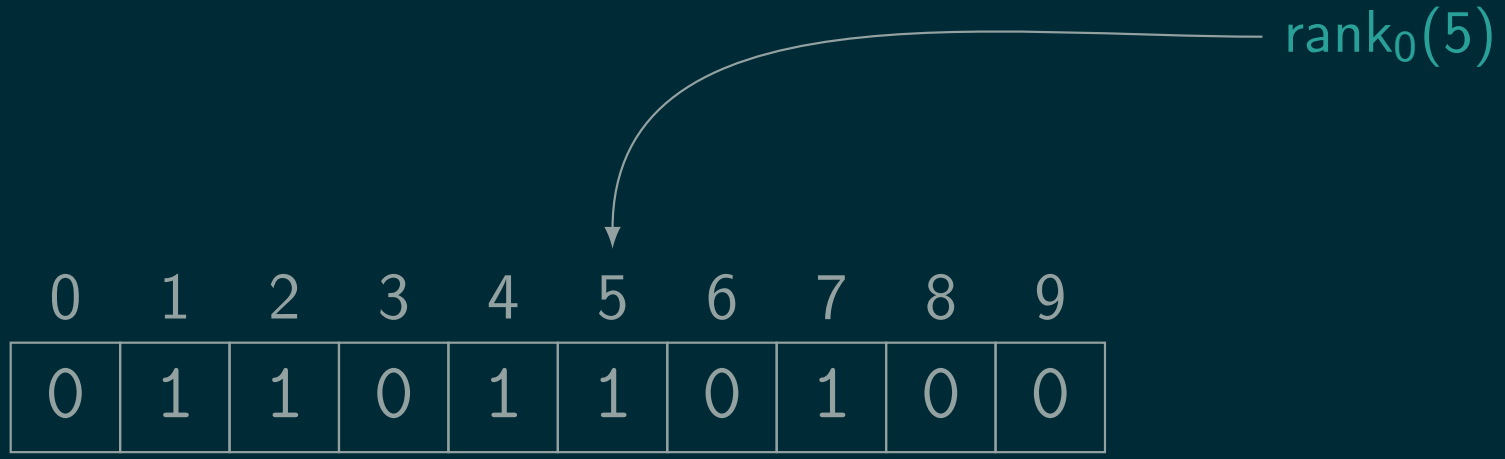
$\text{rank}_0(5)$

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

Rank and Select

$\text{rank}_\alpha(i)$ # of αs before position i

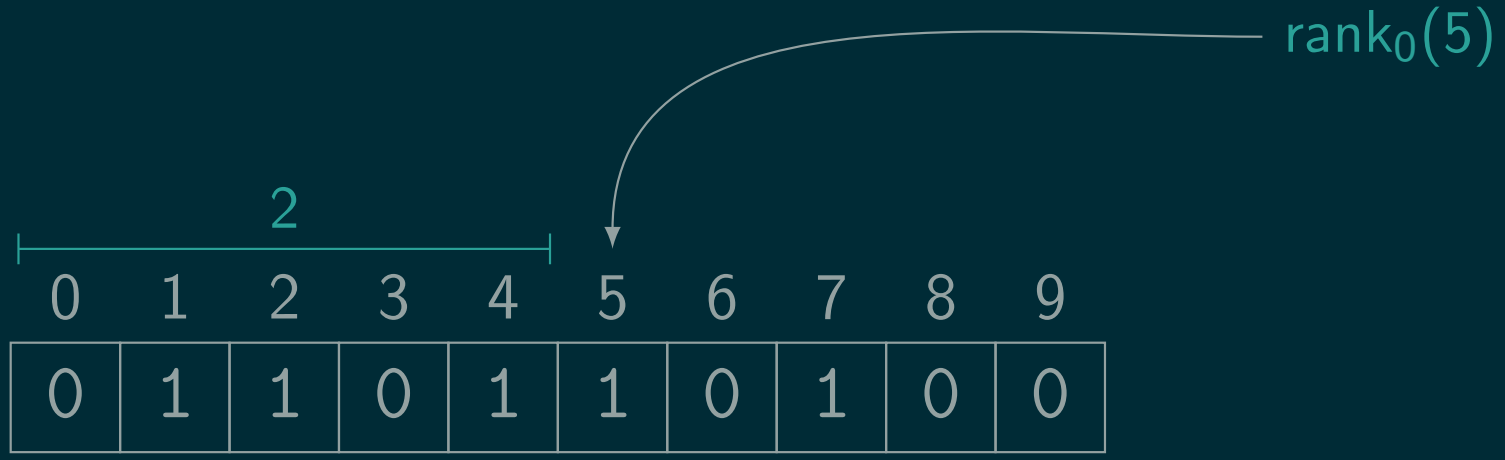
$\text{select}_\alpha(j)$ position of j-th α



Rank and Select

$\text{rank}_\alpha(i)$ # of αs before position i

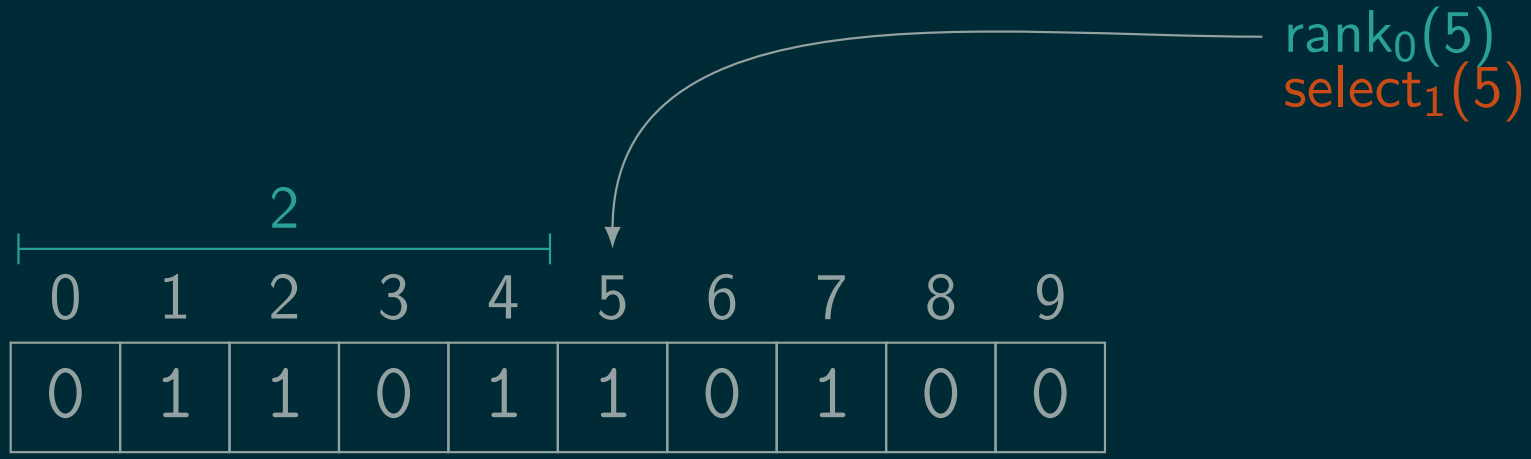
$\text{select}_\alpha(j)$ position of j-th α



Rank and Select

$\text{rank}_\alpha(i)$ # of αs before position i

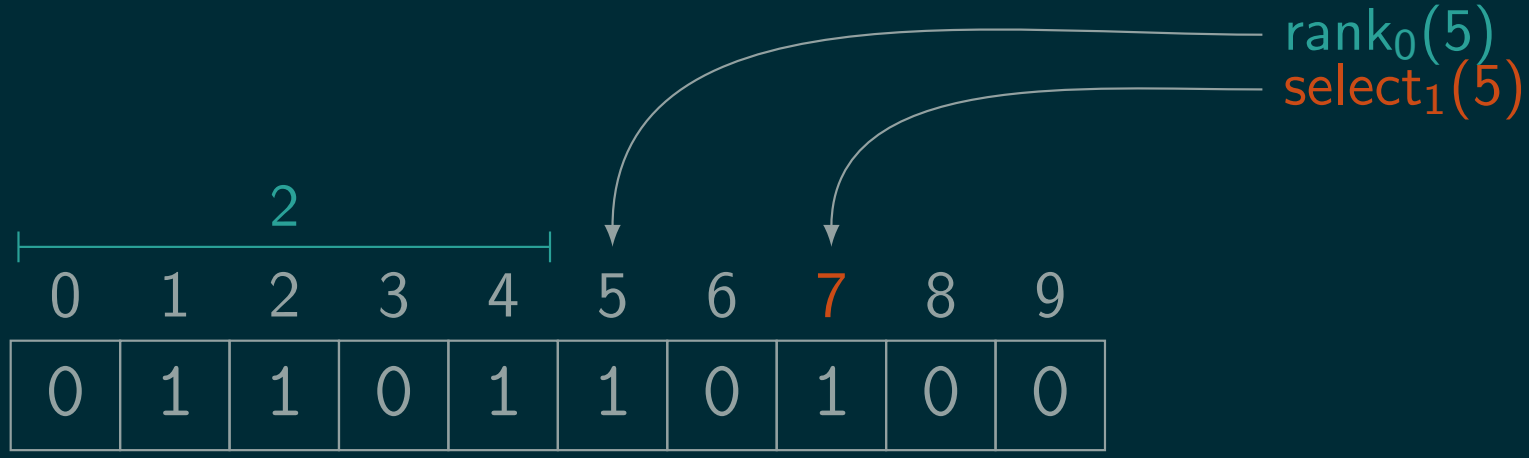
$\text{select}_\alpha(j)$ position of j-th α



Rank and Select

$\text{rank}_\alpha(i)$ # of αs before position i

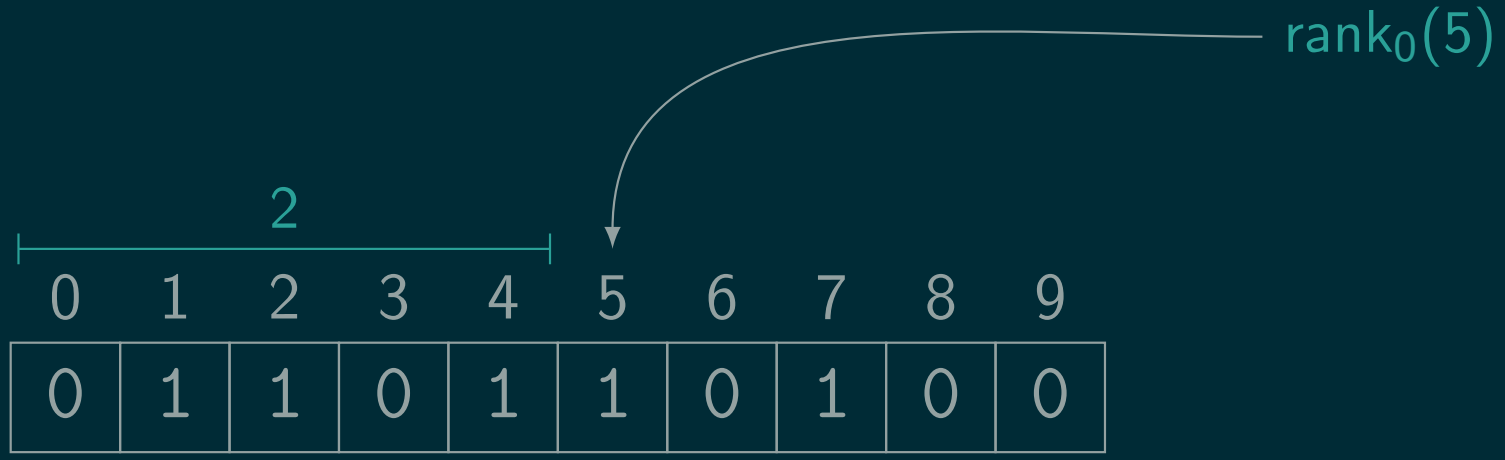
$\text{select}_\alpha(j)$ position of j-th α



Rank and Select

$\text{rank}_\alpha(i)$ # of αs before position i

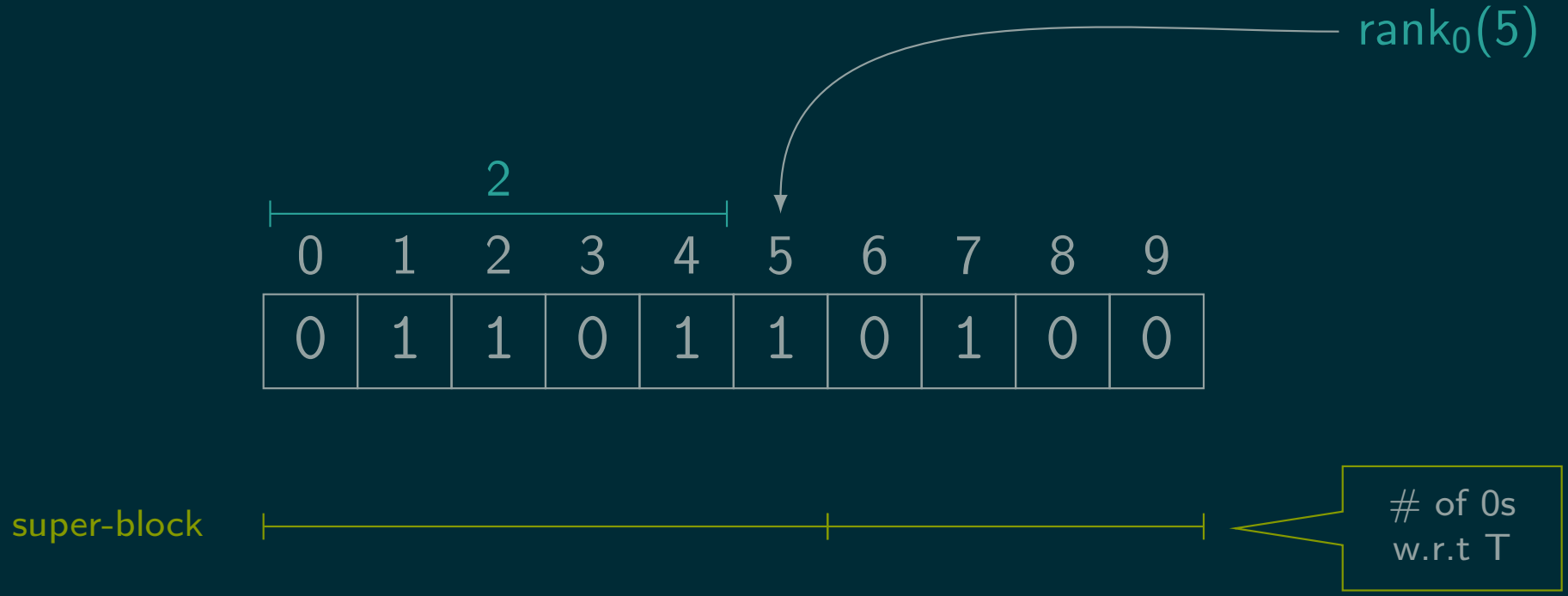
$\text{select}_\alpha(j)$ position of j-th α



Rank and Select

$\text{rank}_\alpha(i)$ # of αs before position i

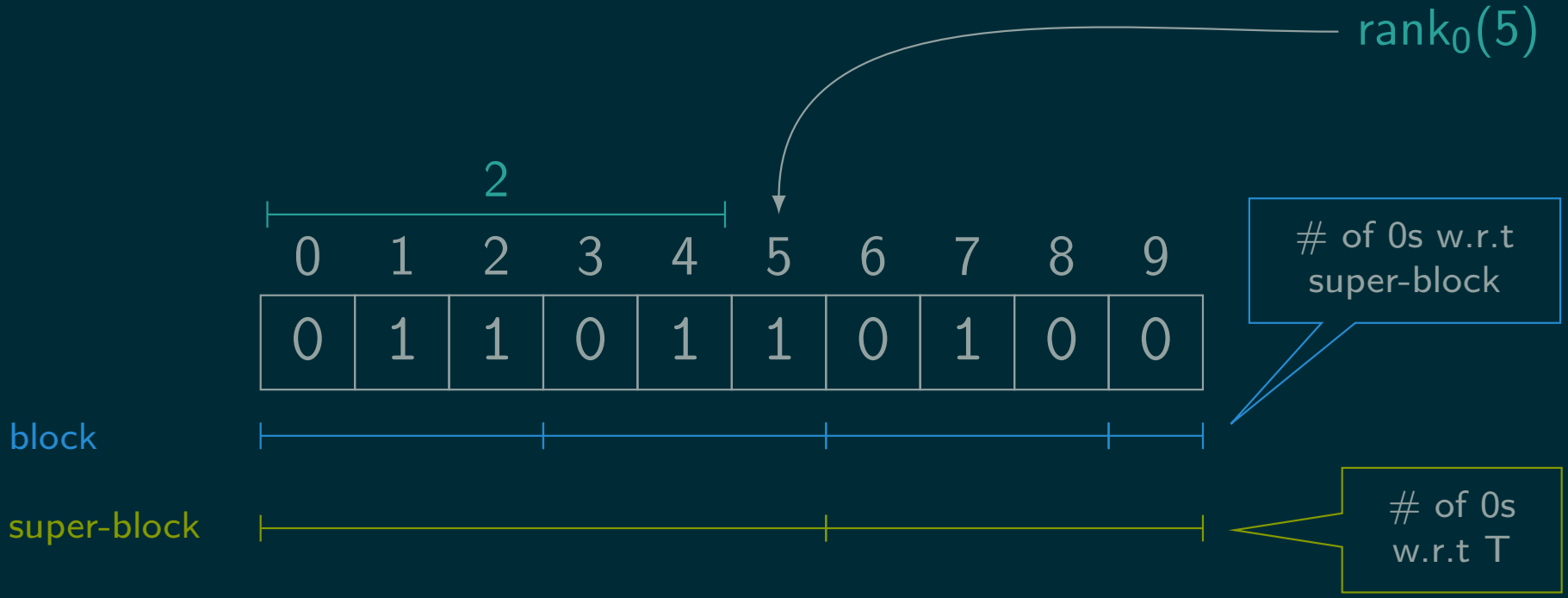
$\text{select}_\alpha(j)$ position of j-th α



Rank and Select

$\text{rank}_\alpha(i)$ # of αs before position i

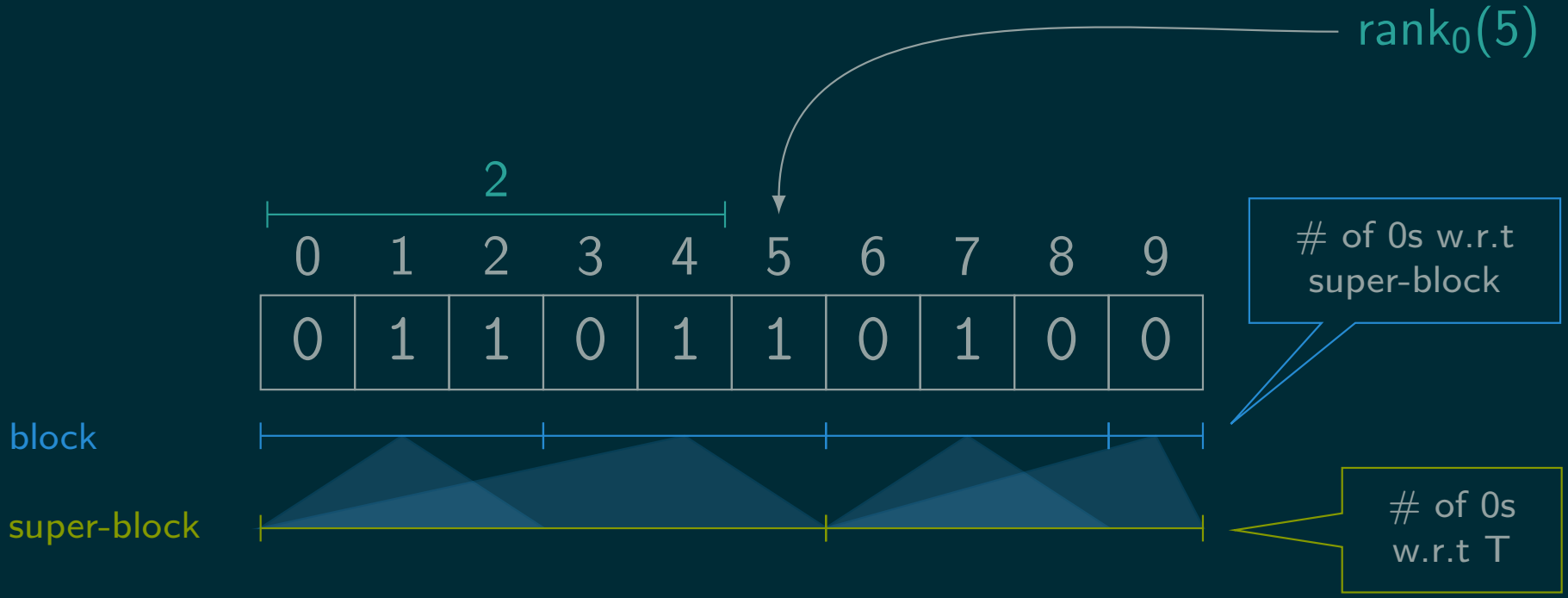
$\text{select}_\alpha(j)$ position of j-th α



Rank and Select

$\text{rank}_\alpha(i)$ # of αs before position i

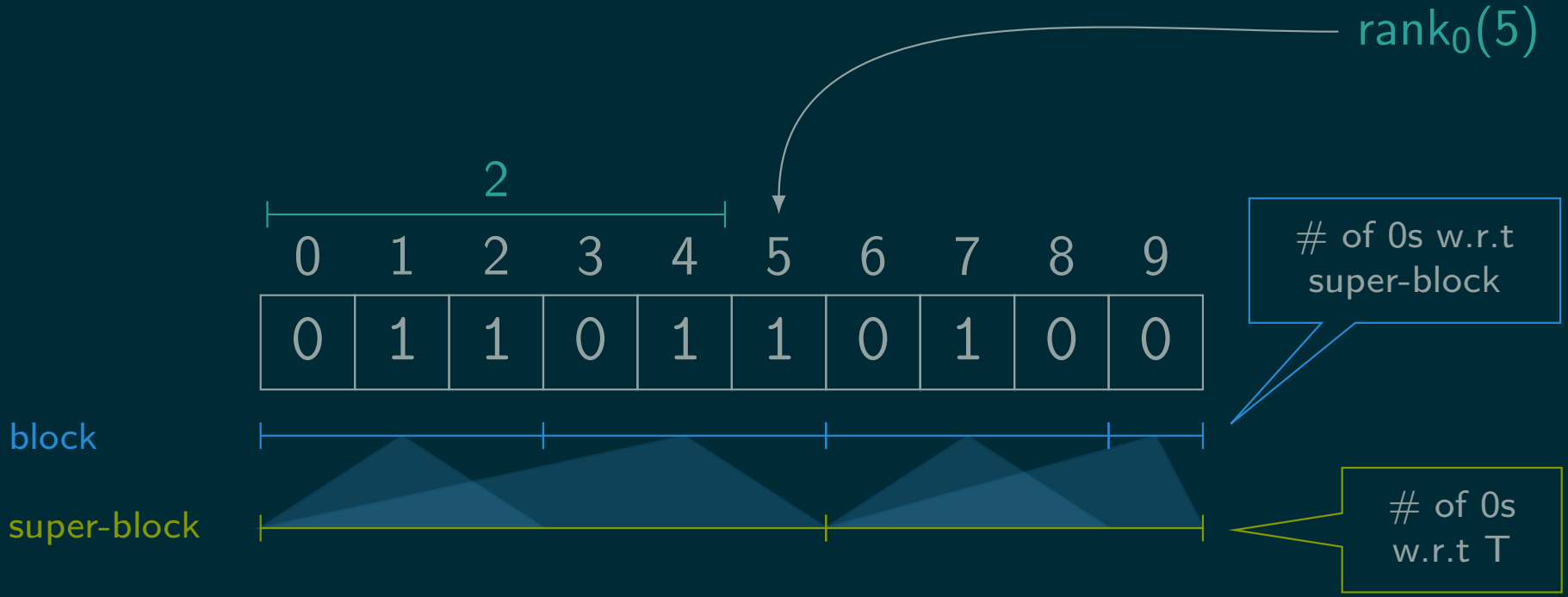
$\text{select}_\alpha(j)$ position of j-th α



Rank and Select

$\text{rank}_\alpha(i)$ # of αs before position i

$\text{select}_\alpha(j)$ position of j-th α



What if $|\Sigma| > 2$

$$\Sigma = \{0, 1, 2, 3, 4, 5, 6, 7\}$$

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

$$\Sigma = \{0, 1, 2, 3, 4, 5, 6, 7\}$$

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

MSB	0	0	1	1	0	1	1	0	1	0
	0	0	1	1	0	0	0	1	1	1
LSB	0	1	0	1	1	1	0	0	0	1

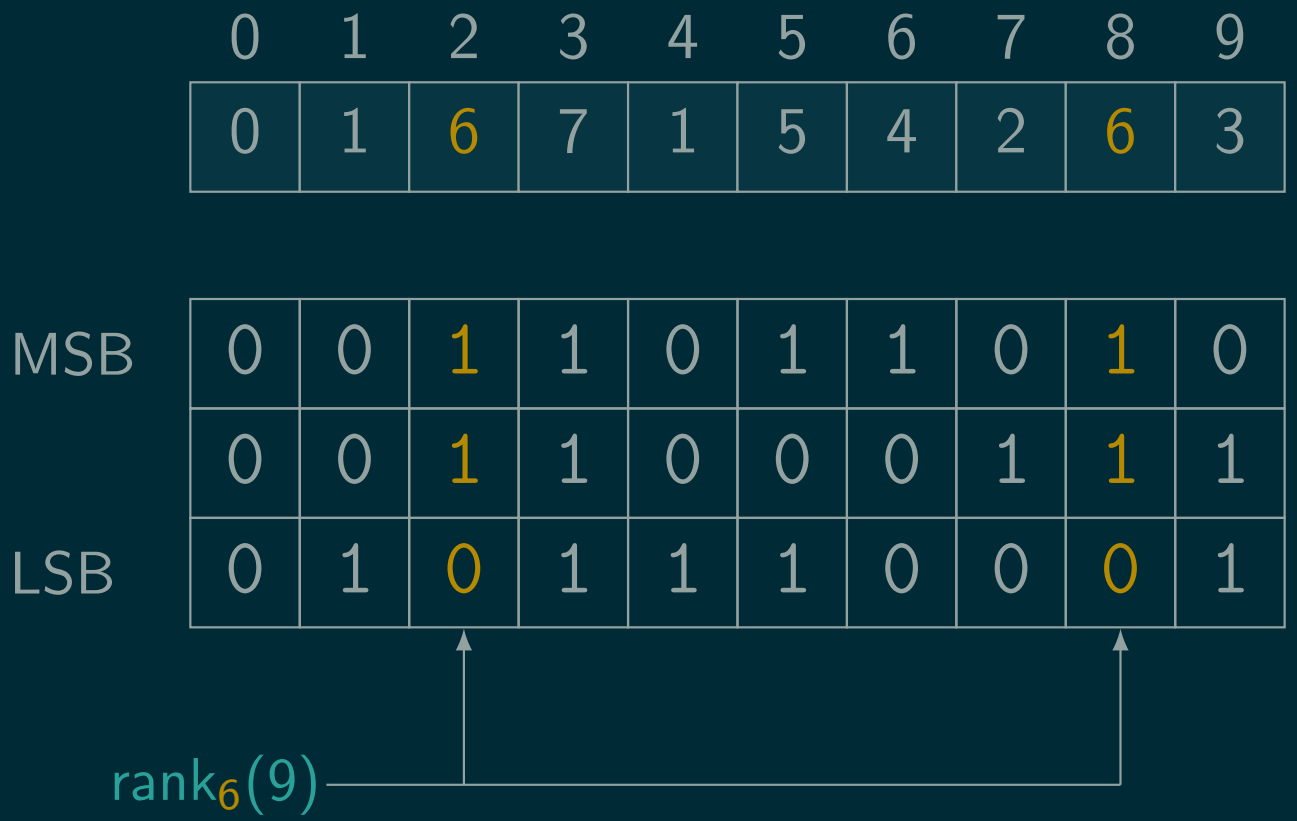
$$\Sigma = \{0, 1, 2, 3, 4, 5, 6, 7\}$$

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

MSB	0	0	1	1	0	1	1	0	1	0
	0	0	1	1	0	0	0	1	1	1
LSB	0	1	0	1	1	1	0	0	0	1

$\text{rank}_6(9)$

$$\Sigma = \{0, 1, 2, 3, 4, 5, 6, 7\}$$



$$\Sigma = \{0, 1, 2, 3, 4, 5, 6, 7\}$$

Wavelet Tree

Pointer-Based

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

$[0, 7]$

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

Wavelet Tree

Pointer-Based

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

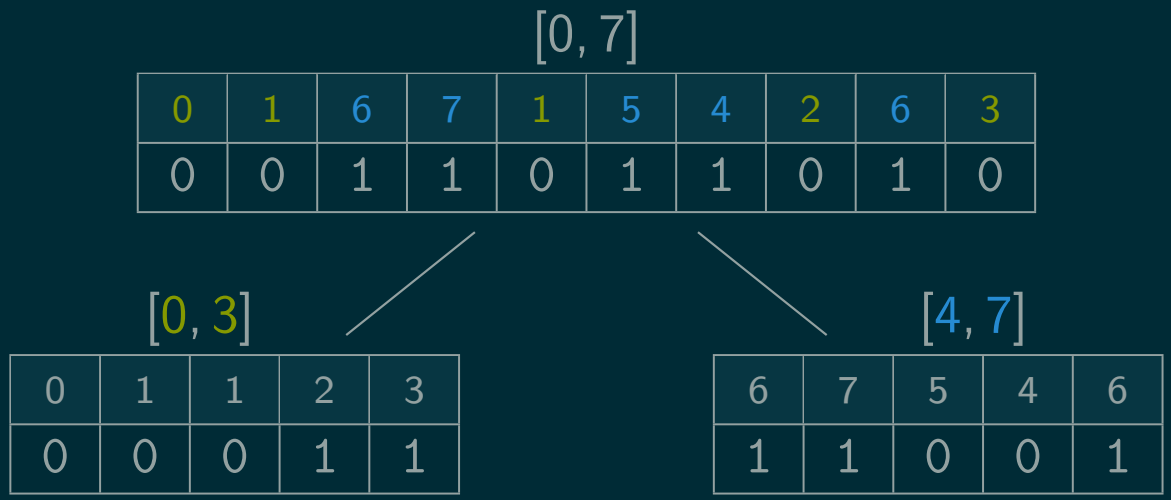
$[0, 7]$

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

Wavelet Tree

Pointer-Based

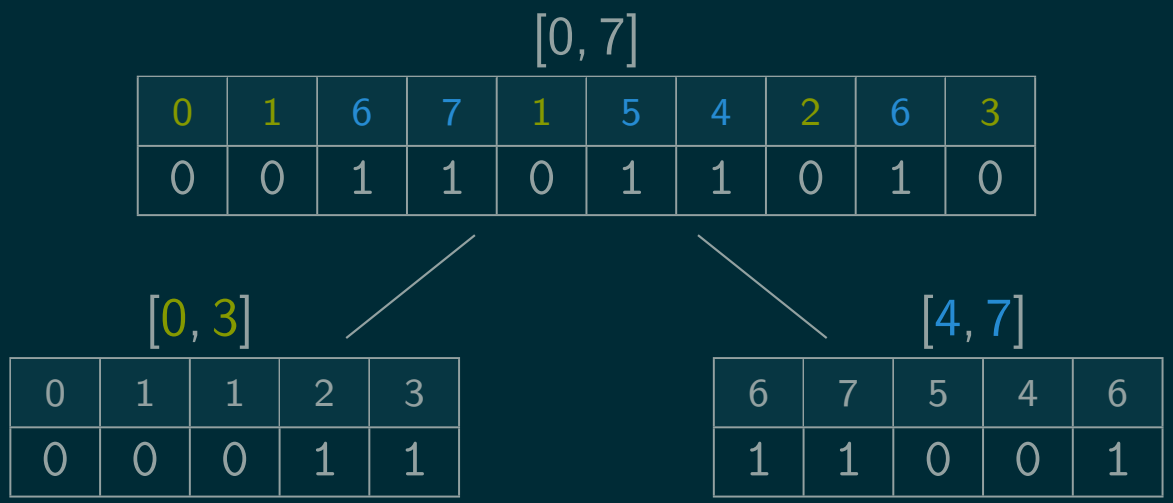
0	1	6	7	1	5	4	2	6	3
0	0	1	1	0	1	1	0	1	0
0	0	1	1	0	0	0	1	1	1
0	1	0	1	1	1	0	0	0	1



Wavelet Tree

Pointer-Based

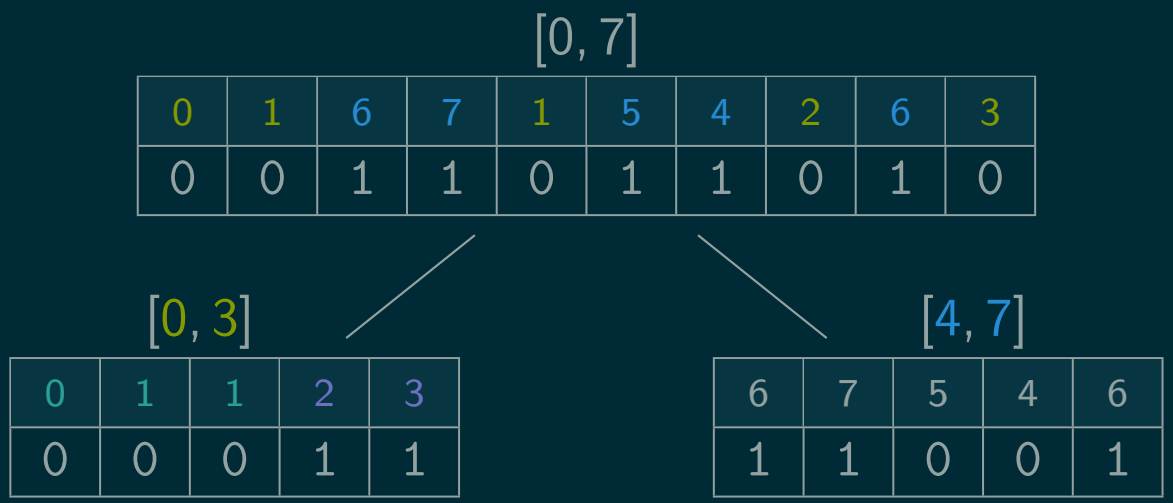
0	1	6	7	1	5	4	2	6	3
0	0	1	1	0	1	1	0	1	0
0	0	1	1	0	0	0	1	1	1
0	1	0	1	1	1	0	0	0	1



Wavelet Tree

Pointer-Based

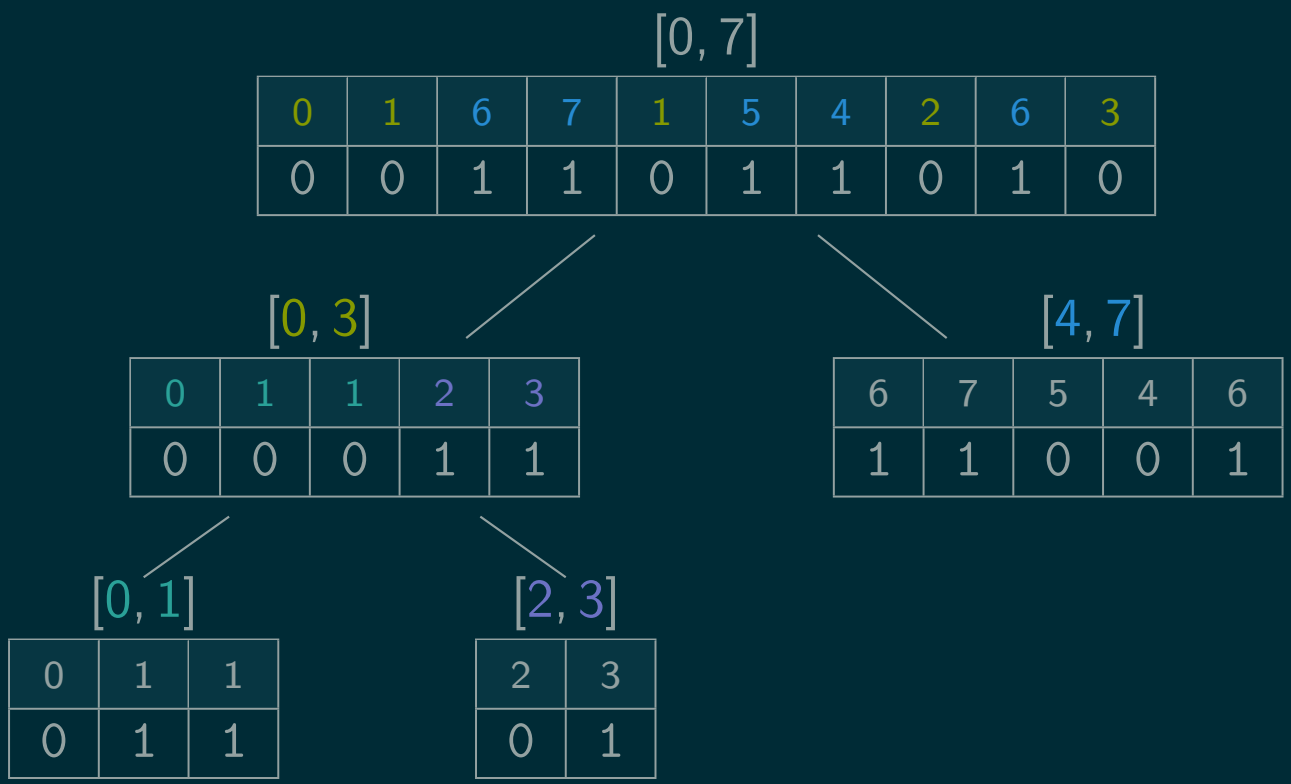
0	1	6	7	1	5	4	2	6	3
0	0	1	1	0	1	1	0	1	0
0	0	1	1	0	0	0	1	1	1
0	1	0	1	1	1	0	0	0	1



Wavelet Tree

Pointer-Based

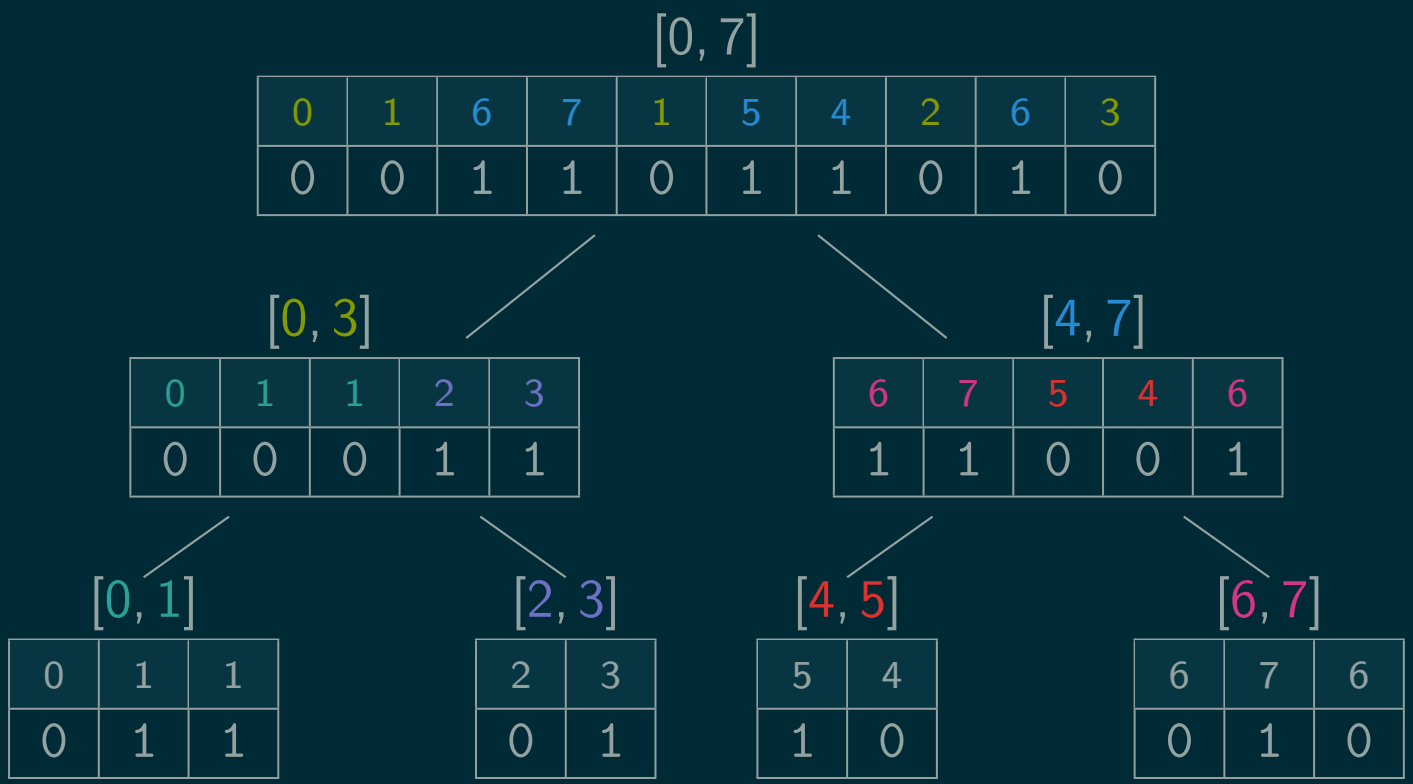
0	1	6	7	1	5	4	2	6	3
0	0	1	1	0	1	1	0	1	0
0	0	1	1	0	0	0	1	1	1
→ 0	1	0	1	1	1	0	0	0	1



Wavelet Tree

Pointer-Based

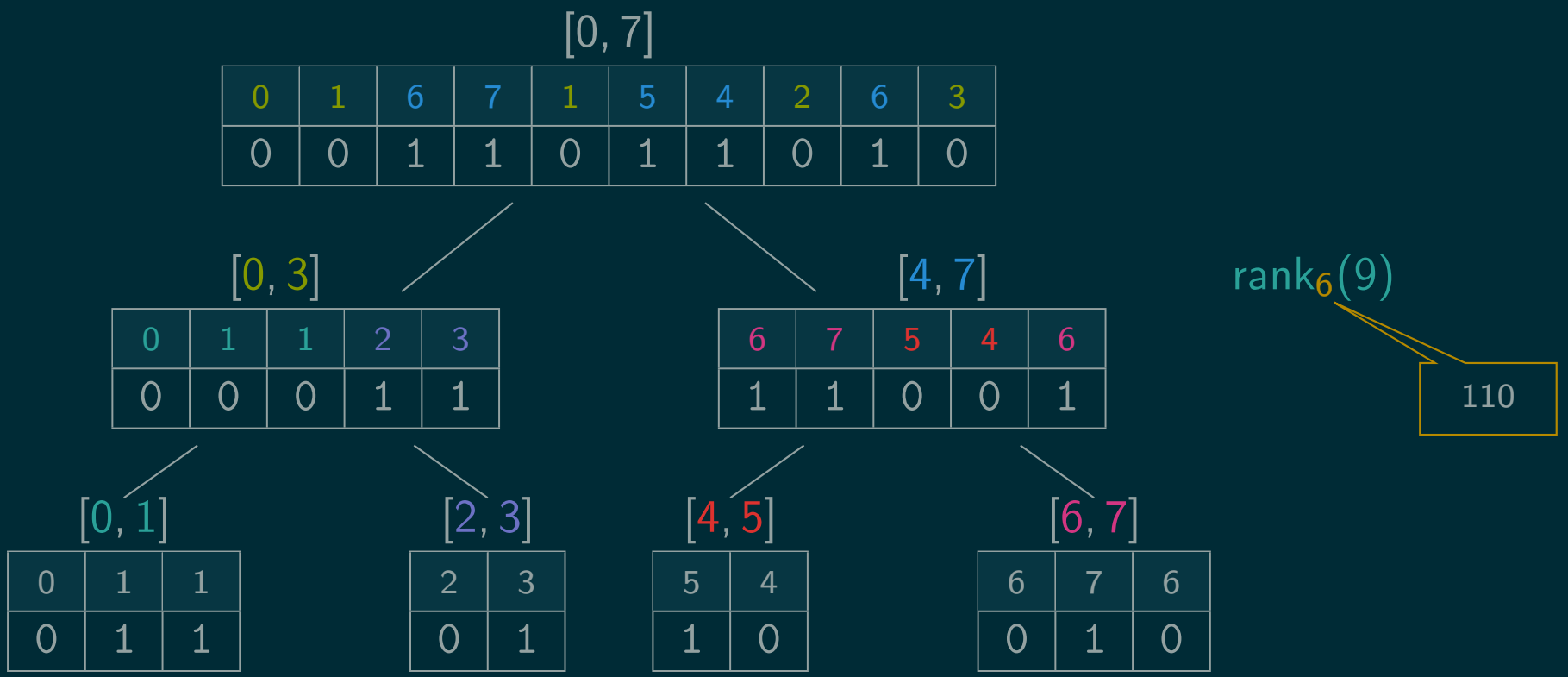
0	1	6	7	1	5	4	2	6	3
0	0	1	1	0	1	1	0	1	0
0	0	1	1	0	0	0	1	1	1
→ 0	1	0	1	1	1	0	0	0	1



Wavelet Tree

Pointer-Based

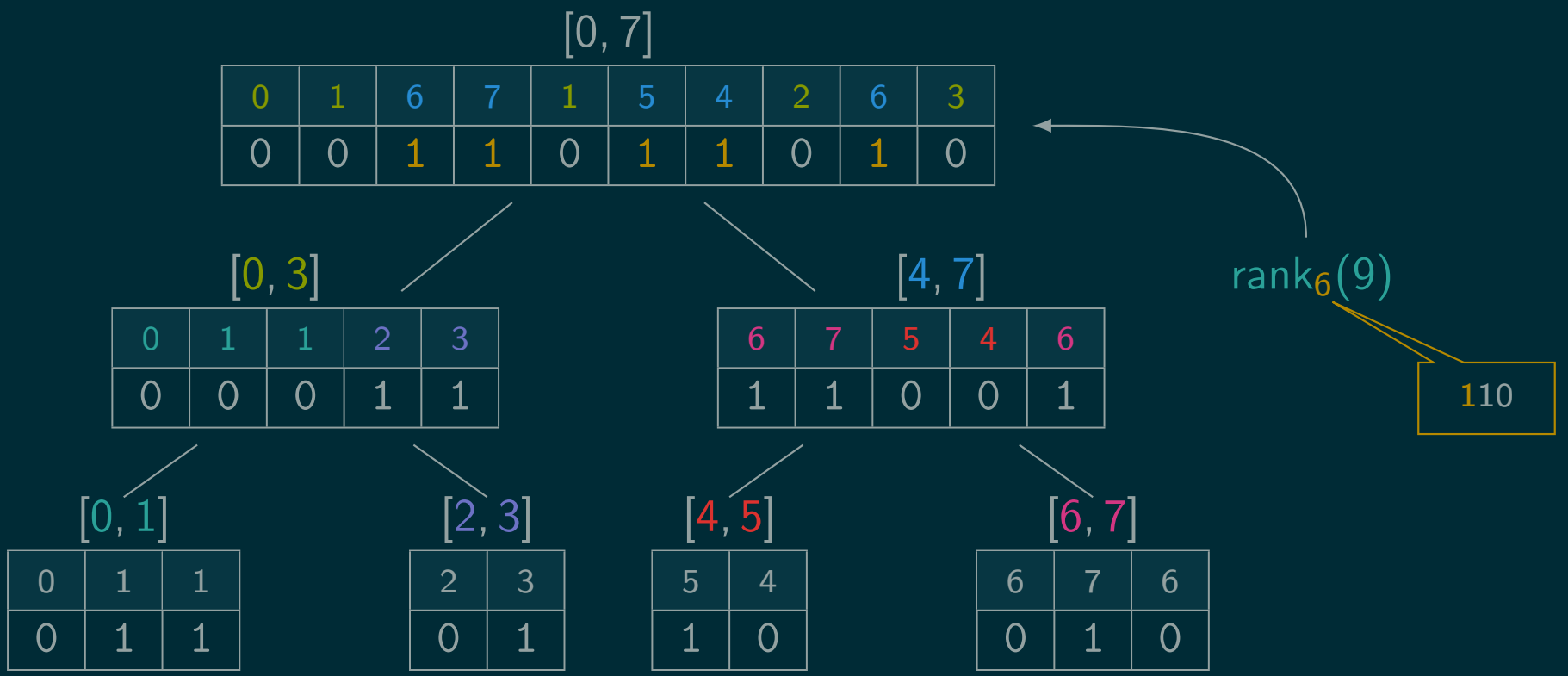
0	1	6	7	1	5	4	2	6	3
0	0	1	1	0	1	1	0	1	0
0	0	1	1	0	0	0	1	1	1
0	1	0	1	1	1	0	0	0	1



Wavelet Tree

Pointer-Based

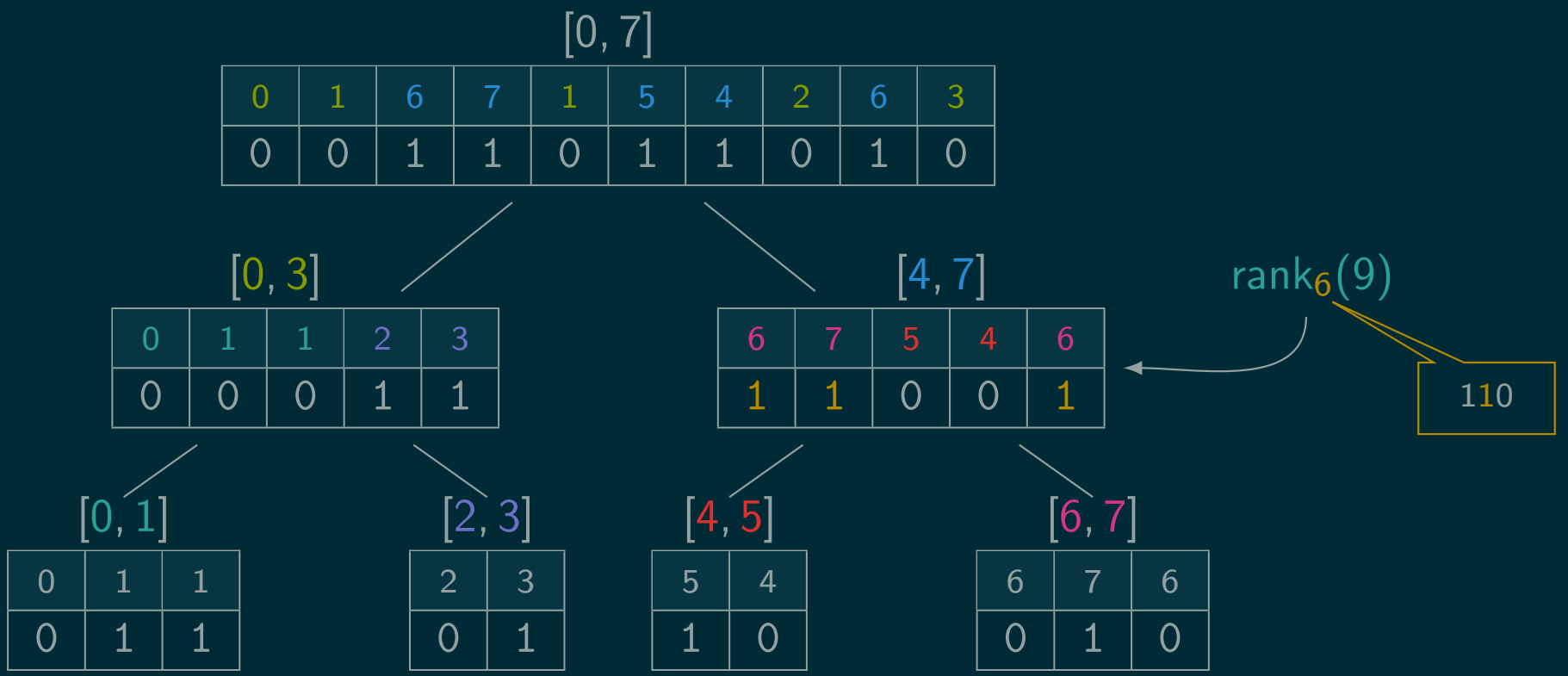
0	1	6	7	1	5	4	2	6	3
0	0	1	1	0	1	1	0	1	0
0	0	1	1	0	0	0	1	1	1
0	1	0	1	1	1	0	0	0	1



Wavelet Tree

Pointer-Based

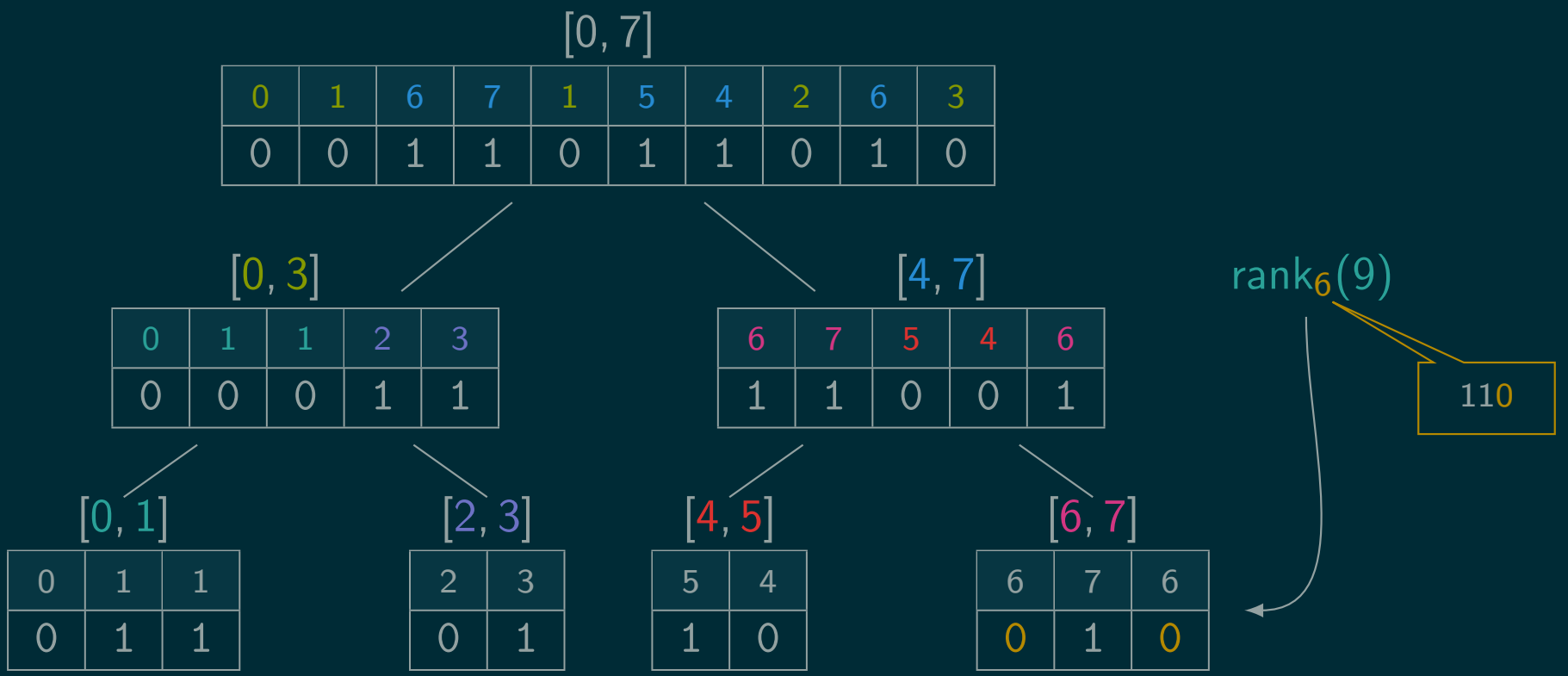
0	1	6	7	1	5	4	2	6	3
0	0	1	1	0	1	1	0	1	0
0	0	1	1	0	0	0	1	1	1
0	1	0	1	1	1	0	0	0	1



Wavelet Tree

Pointer-Based

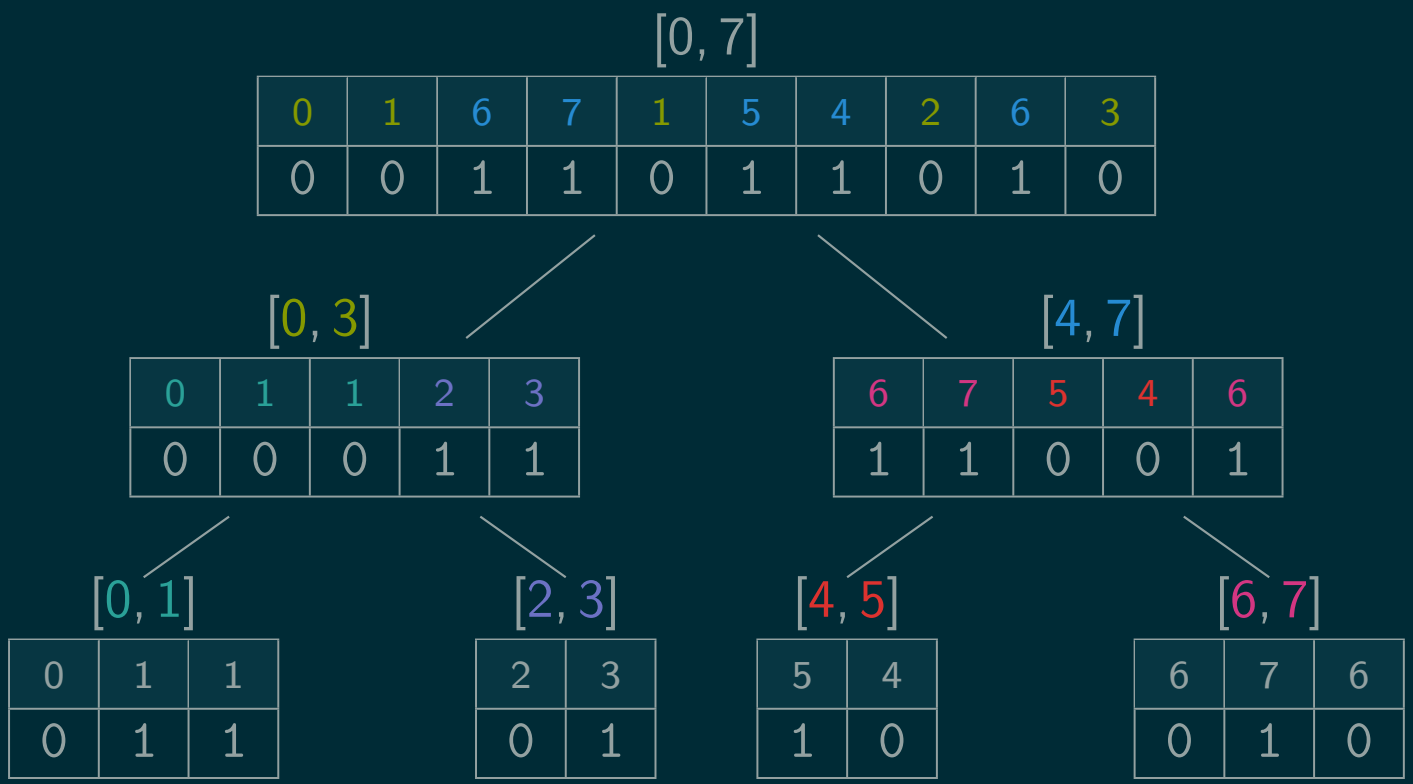
0	1	6	7	1	5	4	2	6	3
0	0	1	1	0	1	1	0	1	0
0	0	1	1	0	0	0	1	1	1
0	1	0	1	1	1	0	0	0	1



Wavelet Tree

Pointer-Based

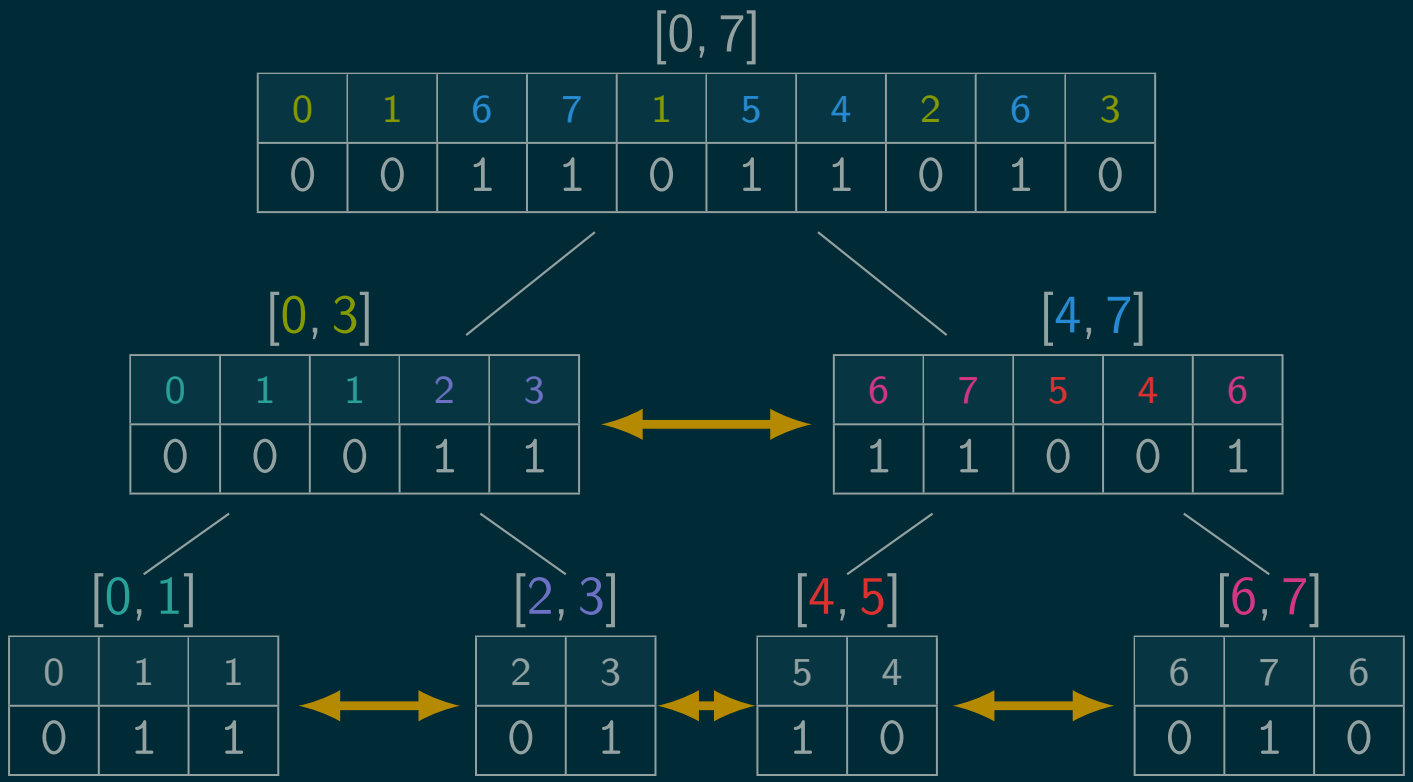
0	1	6	7	1	5	4	2	6	3
0	0	1	1	0	1	1	0	1	0
0	0	1	1	0	0	0	1	1	1
0	1	0	1	1	1	0	0	0	1



Wavelet Tree

Pointer-Based

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



Wavelet Tree

Level-Wise

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

$\ell = 0$

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

$\ell = 1$

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

$\ell = 2$

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

Wavelet Tree

Level-Wise

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

$\ell = 0$

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

MSB of each character

$\ell = 1$

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

$\ell = 2$

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

Wavelet Tree

Level-Wise

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

$\ell = 0$

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

MSB of each character

$\ell = 1$

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

2^{nd} bit of each character with MSB

0 | 1

$\ell = 2$

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

Wavelet Tree

Level-Wise

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

$\ell = 0$

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

MSB of each character

$\ell = 1$

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

2^{nd} bit of each character with MSB

0 | 1

$\ell = 2$

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

3^{rd} bit of each character with MSBs

00 | 01 | 10 | 11

Wavelet Tree

Level-Wise

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

Interval on level ℓ is given by *bit prefix* of length ℓ

$\ell = 0$

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

MSB of each character

$\ell = 1$

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

2nd bit of each character with MSB

0 | 1

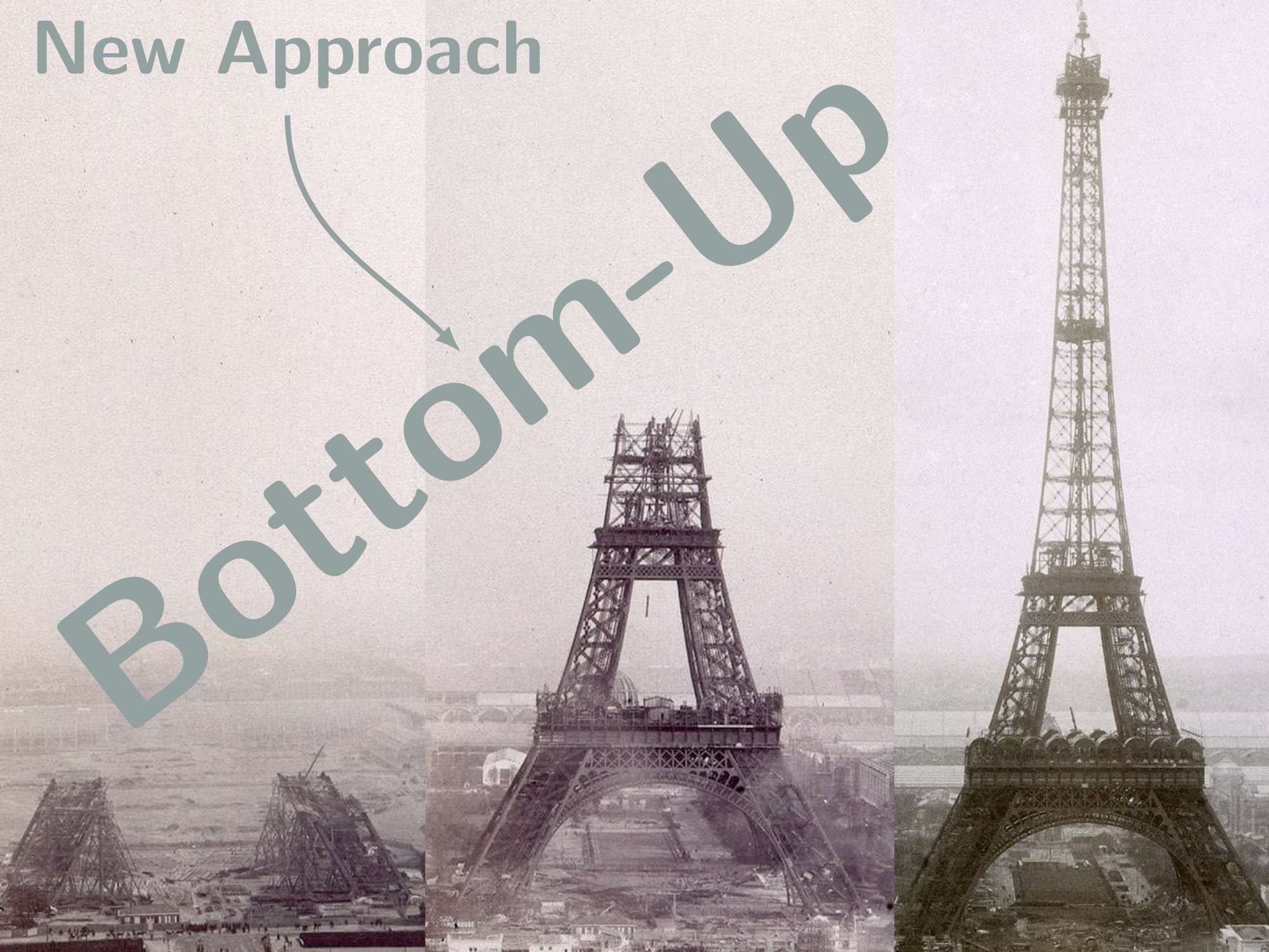
$\ell = 2$

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

3rd bit of each character with MSBs

00 | 01 | 10 | 11

New Approach



Wavelet Tree

Bottom-Up

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

Interval on level ℓ is given by *bit prefix* of length ℓ

$\ell = 0$

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

$\ell = 1$

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

$\ell = 2$

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

Wavelet Tree

Bottom-Up

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

Interval on level ℓ is given by *bit prefix* of length ℓ

$\ell = 0$

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

$00 \rightarrow [0, 1]$	0
$01 \rightarrow [2, 3]$	0
$10 \rightarrow [4, 5]$	0
$11 \rightarrow [6, 7]$	0

$\ell = 1$

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

$\ell = 2$

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

Wavelet Tree

Bottom-Up

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

Interval on level ℓ is given by *bit prefix* of length ℓ

$\ell = 0$

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

$00 \rightarrow [0, 1]$	1
$01 \rightarrow [2, 3]$	0
$10 \rightarrow [4, 5]$	0
$11 \rightarrow [6, 7]$	0

$\ell = 1$

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

$\ell = 2$

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

Wavelet Tree

Bottom-Up

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

Interval on level ℓ is given by *bit prefix* of length ℓ

$\ell = 0$

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

$00 \rightarrow [0, 1]$	2
$01 \rightarrow [2, 3]$	0
$10 \rightarrow [4, 5]$	0
$11 \rightarrow [6, 7]$	0

$\ell = 1$

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

$\ell = 2$

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

Wavelet Tree

Bottom-Up

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

Interval on level ℓ is given by *bit prefix* of length ℓ

$\ell = 0$

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

$00 \rightarrow [0, 1]$	2
$01 \rightarrow [2, 3]$	0
$10 \rightarrow [4, 5]$	0
$11 \rightarrow [6, 7]$	1

$\ell = 1$

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

$\ell = 2$

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

Wavelet Tree

Bottom-Up

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

Interval on level ℓ is given by *bit prefix* of length ℓ

$\ell = 0$

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

$\ell = 1$

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

$\ell = 2$

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

$00 \rightarrow [0, 1]$	2
$01 \rightarrow [2, 3]$	0
$10 \rightarrow [4, 5]$	0
$11 \rightarrow [6, 7]$	2

Wavelet Tree

Bottom-Up

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

Interval on level ℓ is given by *bit prefix* of length ℓ

$\ell = 0$

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

$\ell = 1$

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

$\ell = 2$

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

$00 \rightarrow [0, 1]$	3
$01 \rightarrow [2, 3]$	2
$10 \rightarrow [4, 5]$	2
$11 \rightarrow [6, 7]$	3

Wavelet Tree

Bottom-Up

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

Interval on level ℓ is given by *bit prefix* of length ℓ

$\ell = 0$

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

$\ell = 1$

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

$\ell = 2$

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

$00 \rightarrow [0, 1]$	3
$01 \rightarrow [2, 3]$	2
$10 \rightarrow [4, 5]$	2
$11 \rightarrow [6, 7]$	3

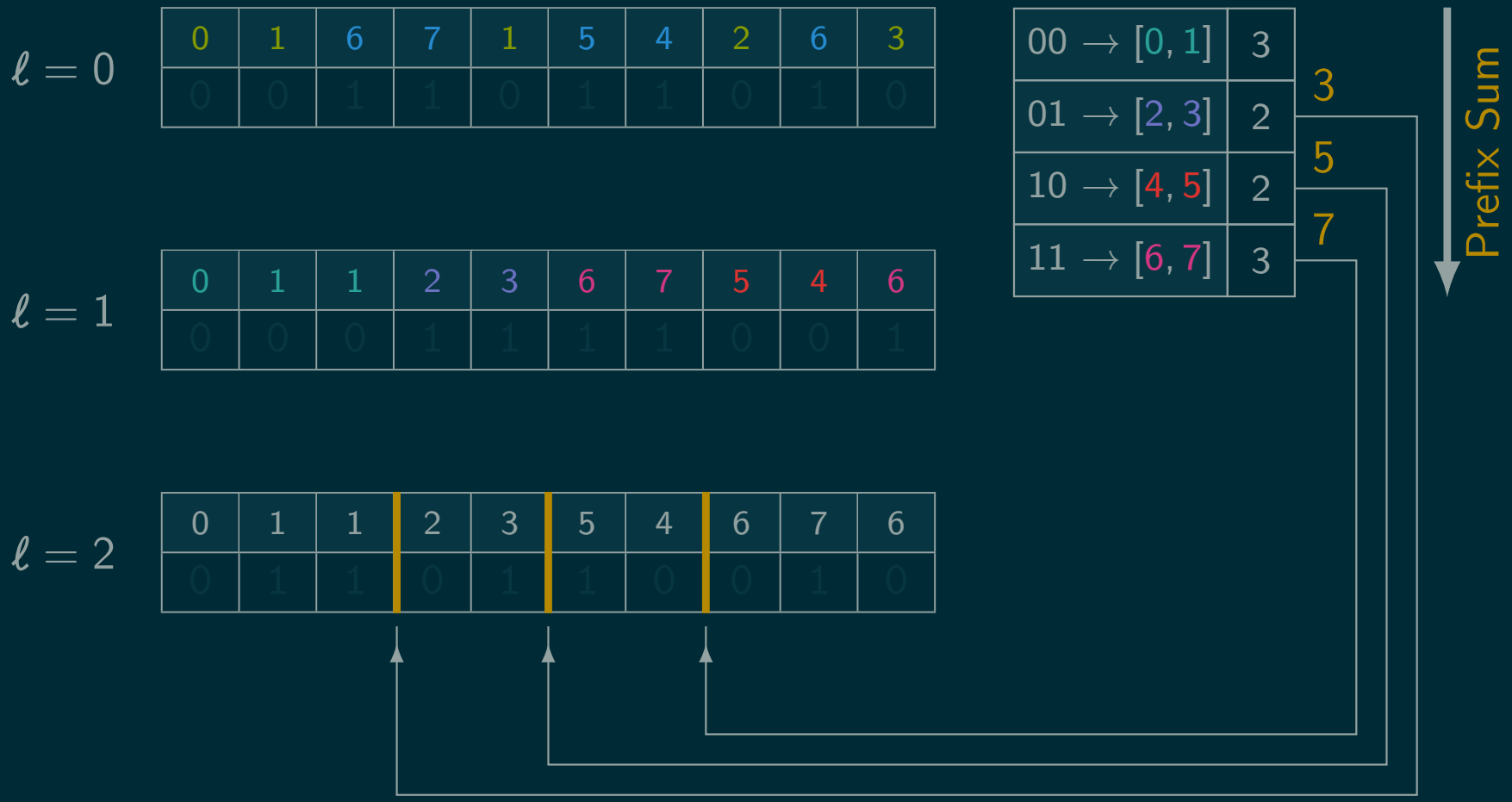
Prefix Sum

Wavelet Tree

Bottom-Up

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

Interval on level ℓ is given by *bit prefix* of length ℓ



Wavelet Tree

Bottom-Up

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

Interval on level ℓ is given by *bit prefix* of length ℓ

$\ell = 0$

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

$\ell = 1$

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

$\ell = 2$

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

$00 \rightarrow [0, 1]$	3
$01 \rightarrow [2, 3]$	2
$10 \rightarrow [4, 5]$	2
$11 \rightarrow [6, 7]$	3

3
5
7

Prefix Sum

Wavelet Tree

Bottom-Up

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

Interval on level ℓ is given by *bit prefix* of length ℓ

$\ell = 0$

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

$\ell = 1$

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

$\ell = 2$

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

$00 \rightarrow [0, 1]$	3
$01 \rightarrow [2, 3]$	2
$10 \rightarrow [4, 5]$	2
$11 \rightarrow [6, 7]$	3

3
5
7

Prefix Sum

Wavelet Tree

Bottom-Up

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

Interval on level ℓ is given by *bit prefix* of length ℓ

$\ell = 0$

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

$\ell = 1$

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

$\ell = 2$

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

$00 \rightarrow [0, 1]$	3
$01 \rightarrow [2, 3]$	2
$10 \rightarrow [4, 5]$	2
$11 \rightarrow [6, 7]$	3

3
5
7

Prefix Sum

Wavelet Tree

Bottom-Up

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

Interval on level ℓ is given by *bit prefix* of length ℓ

$\ell = 0$

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

$\ell = 1$

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

$\ell = 2$

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

$00 \rightarrow [0, 1]$	3
$01 \rightarrow [2, 3]$	2
$10 \rightarrow [4, 5]$	2
$11 \rightarrow [6, 7]$	3

3
5
7

Prefix Sum

Wavelet Tree

Bottom-Up

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

Interval on level ℓ is given by *bit prefix* of length ℓ

$\ell = 0$

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

$\ell = 1$

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

$\ell = 2$

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

$00 \rightarrow [0, 1]$	3
$01 \rightarrow [2, 3]$	2
$10 \rightarrow [4, 5]$	2
$11 \rightarrow [6, 7]$	3

3
5
7

Prefix Sum

Wavelet Tree

Bottom-Up

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

Interval on level ℓ is given by *bit prefix* of length ℓ

$\ell = 0$

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

$\ell = 1$

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

$\ell = 2$

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

$00 \rightarrow [0, 1]$	3
$01 \rightarrow [2, 3]$	2
$10 \rightarrow [4, 5]$	2
$11 \rightarrow [6, 7]$	3

Compute the borders for the **previous** level without rescanning T.

Wavelet Tree

Bottom-Up

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

Interval on level ℓ is given by *bit prefix* of length ℓ

$\ell = 0$

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

$\ell = 1$

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

$\ell = 2$

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

$00 \rightarrow [0, 1]$	3
$01 \rightarrow [2, 3]$	2
$10 \rightarrow [4, 5]$	2
$11 \rightarrow [6, 7]$	3

5

5

Compute the borders for the **previous** level without rescanning T.

Wavelet Tree

Bottom-Up

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

Interval on level ℓ is given by *bit prefix* of length ℓ

$\ell = 0$

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

$\ell = 1$

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

$\ell = 2$

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

$00 \rightarrow [0, 1]$	3
$01 \rightarrow [2, 3]$	2
$10 \rightarrow [4, 5]$	2
$11 \rightarrow [6, 7]$	3

5

5

Compute the borders for the **previous** level without rescanning T.

Wavelet Tree

Bottom-Up

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

Interval on level ℓ is given by *bit prefix* of length ℓ

$\ell = 0$

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

$\ell = 1$

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

$\ell = 2$

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

$00 \rightarrow [0, 1]$	3
$01 \rightarrow [2, 3]$	2
$10 \rightarrow [4, 5]$	2
$11 \rightarrow [6, 7]$	3

5

5

Compute the borders for the **previous** level without rescanning T.

Wavelet Tree

Bottom-Up

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

Interval on level ℓ is given by *bit prefix* of length ℓ

$\ell = 0$

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

$\ell = 1$

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

$\ell = 2$

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

$00 \rightarrow [0, 1]$	3
$01 \rightarrow [2, 3]$	2
$10 \rightarrow [4, 5]$	2
$11 \rightarrow [6, 7]$	3

5

5

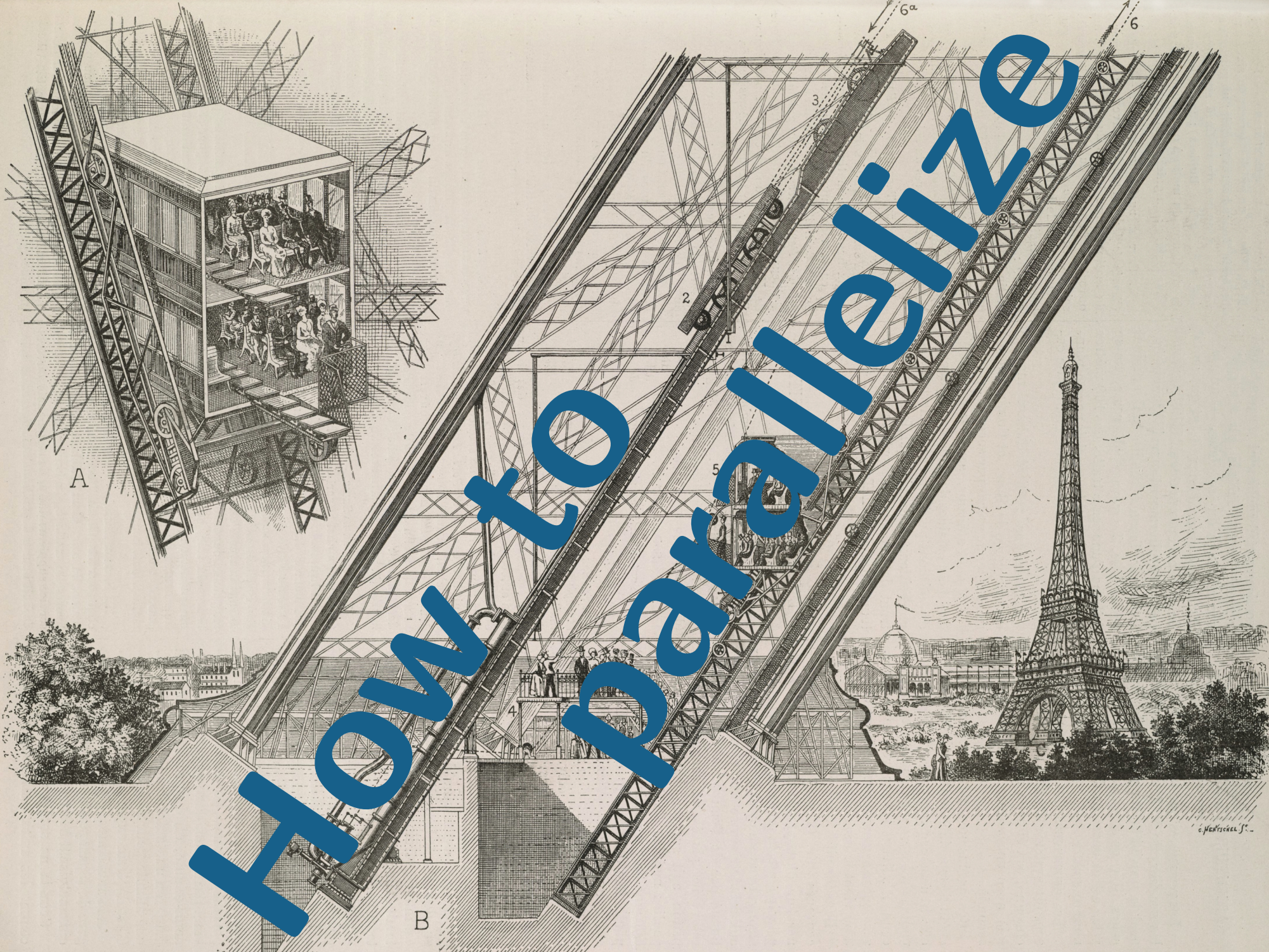
Compute the borders for the **previous** level without rescanning T.

Recap
Bottom-Up

1. Compute histogram for level $\ell = \lceil \lg |\Sigma| \rceil$ (plus bit vector of first level)
2. Compute borders for level ℓ
3. Fill bit vector for level ℓ accordingly
4. Compute histogram for level $\ell - 1$ using the current histogram
5. If $\ell \geq 1$ set $\ell = \ell - 1$ and repeat at step 2

Recap
Bottom-Up

1. Compute histogram for level $\ell = \lceil \lg |\Sigma| \rceil$ (plus bit vector of first level)
2. Compute borders for level ℓ
3. Fill bit vector for level ℓ accordingly
4. Compute histogram for level $\ell - 1$ using the current histogram
5. If $\ell \geq 1$ set $\ell = \ell - 1$ and repeat at step 2



How to parallelize

Prefix Counting

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

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

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

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

Prefix Counting

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

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

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

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

Prefix Counting

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

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

$0 \rightarrow [0, 3]$	3
$1 \rightarrow [4, 7]$	2

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

$00 \rightarrow [0, 1]$	3
$01 \rightarrow [2, 3]$	2
$10 \rightarrow [4, 5]$	2
$11 \rightarrow [6, 7]$	3

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

Prefix Counting

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

Problem: Utilizes at most $\lceil \lg \sigma \rceil$ cores

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

$0 \rightarrow [0, 3]$	3
$1 \rightarrow [4, 7]$	2

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

$00 \rightarrow [0, 1]$	3
$01 \rightarrow [2, 3]$	2
$10 \rightarrow [4, 5]$	2
$11 \rightarrow [6, 7]$	3

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

Prefix Sorting

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

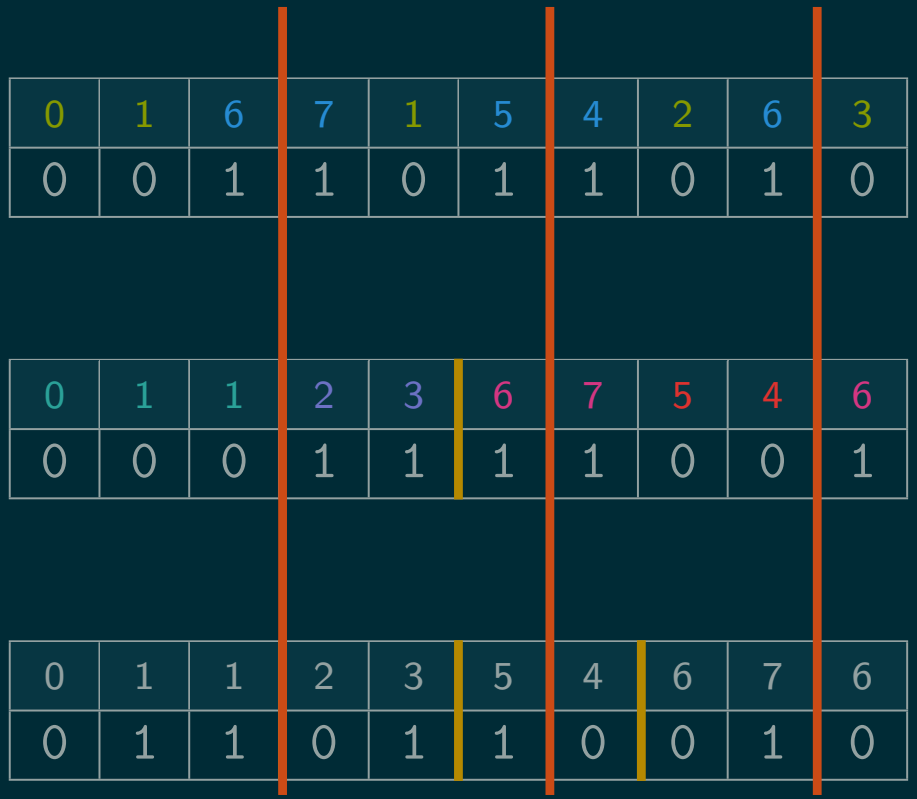
0	1	6	7	1	5	4	2	6	3
0	0	1	1	0	1	1	0	1	0

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

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

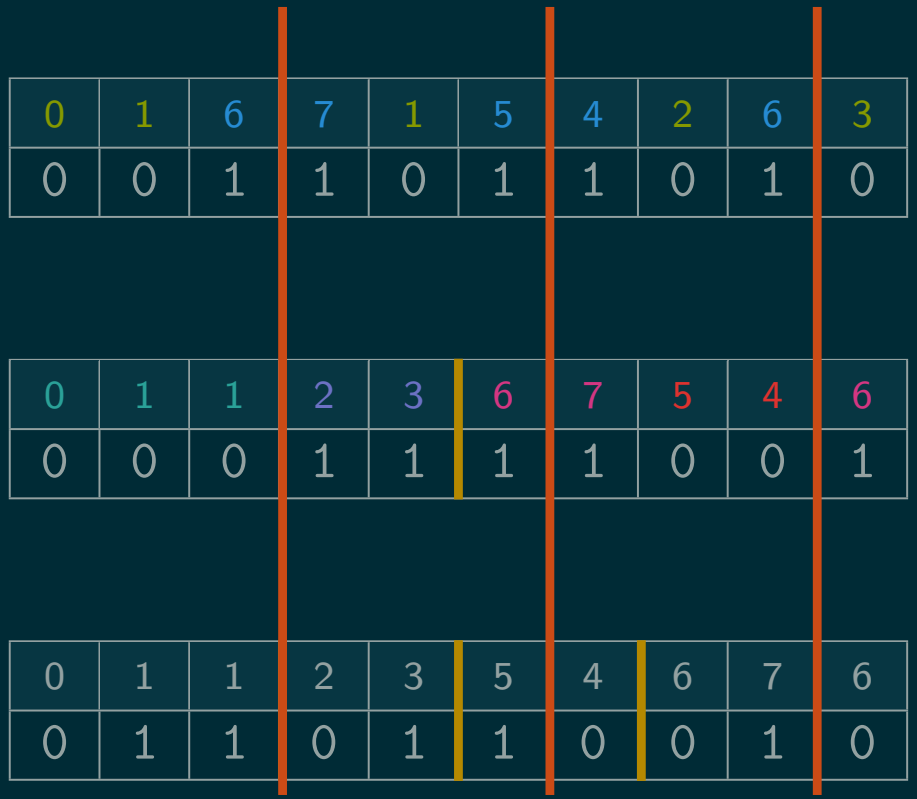
Prefix Sorting

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



Prefix Sorting

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



Starting with the last level:

- Compute borders
- Sort text (keeping original)
- Fill bit vector

Domain Decomposition

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

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

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

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

Domain Decomposition

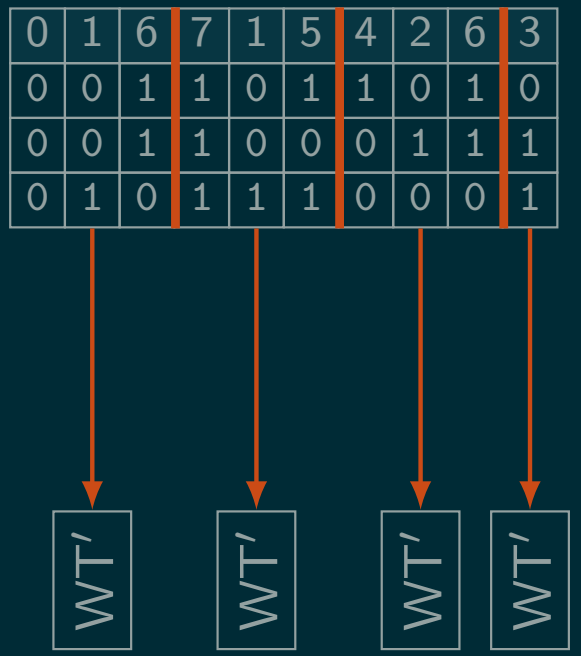
0	1	6	7	1	5	4	2	6	3
0	0	1	1	0	1	1	0	1	0
0	0	1	1	0	0	0	1	1	1
0	1	0	1	1	1	0	0	0	1

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

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

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

Domain Decomposition

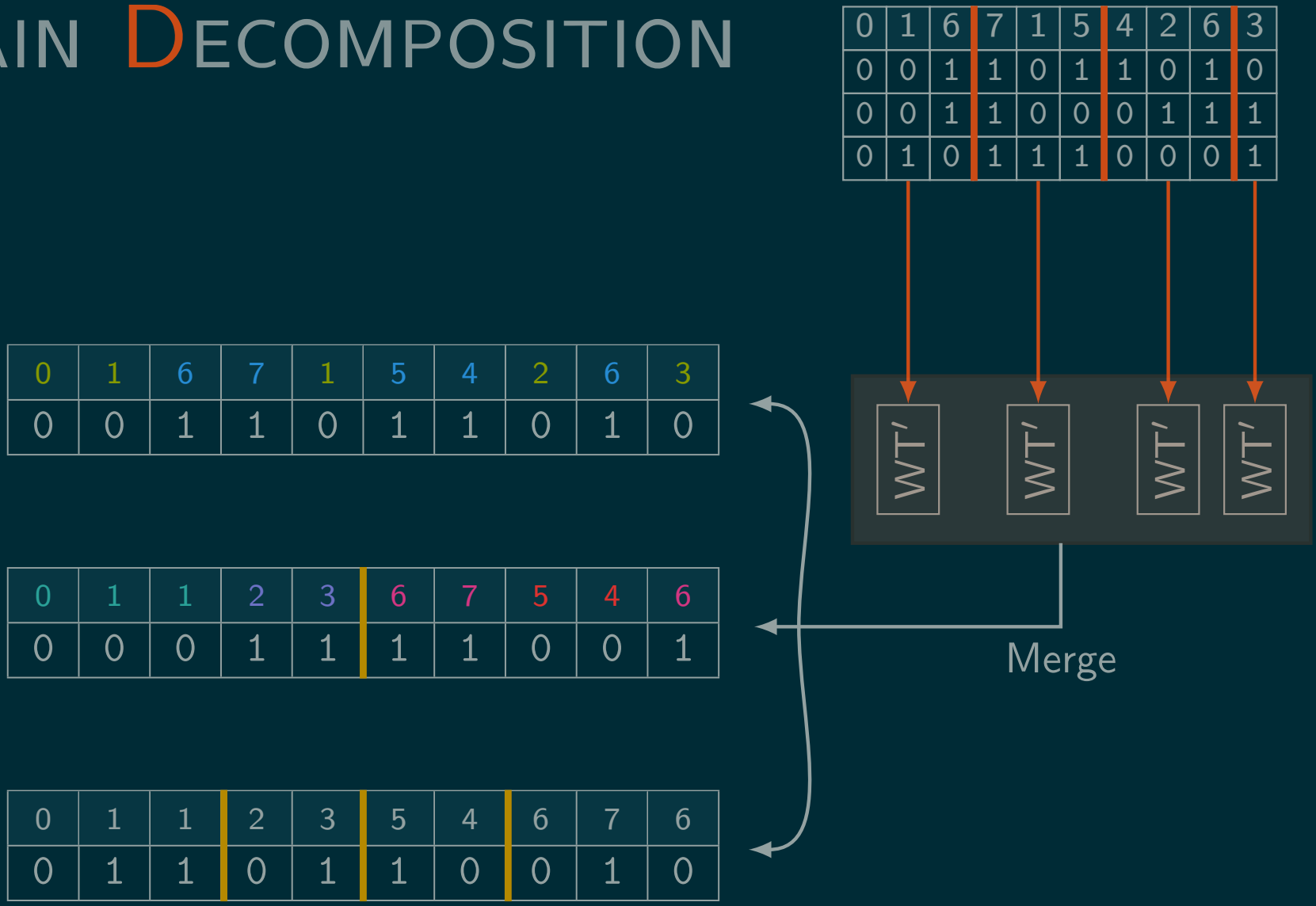


0	1	6	7	1	5	4	2	6	3
0	0	1	1	0	1	1	0	1	0

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

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

Domain Decomposition



Wavelet Matrix

Wavelet Matrix

Similar but Different

Wavelet Matrix

Similar but Different

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

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

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

Wavelet Matrix

Similar but Different

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

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

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

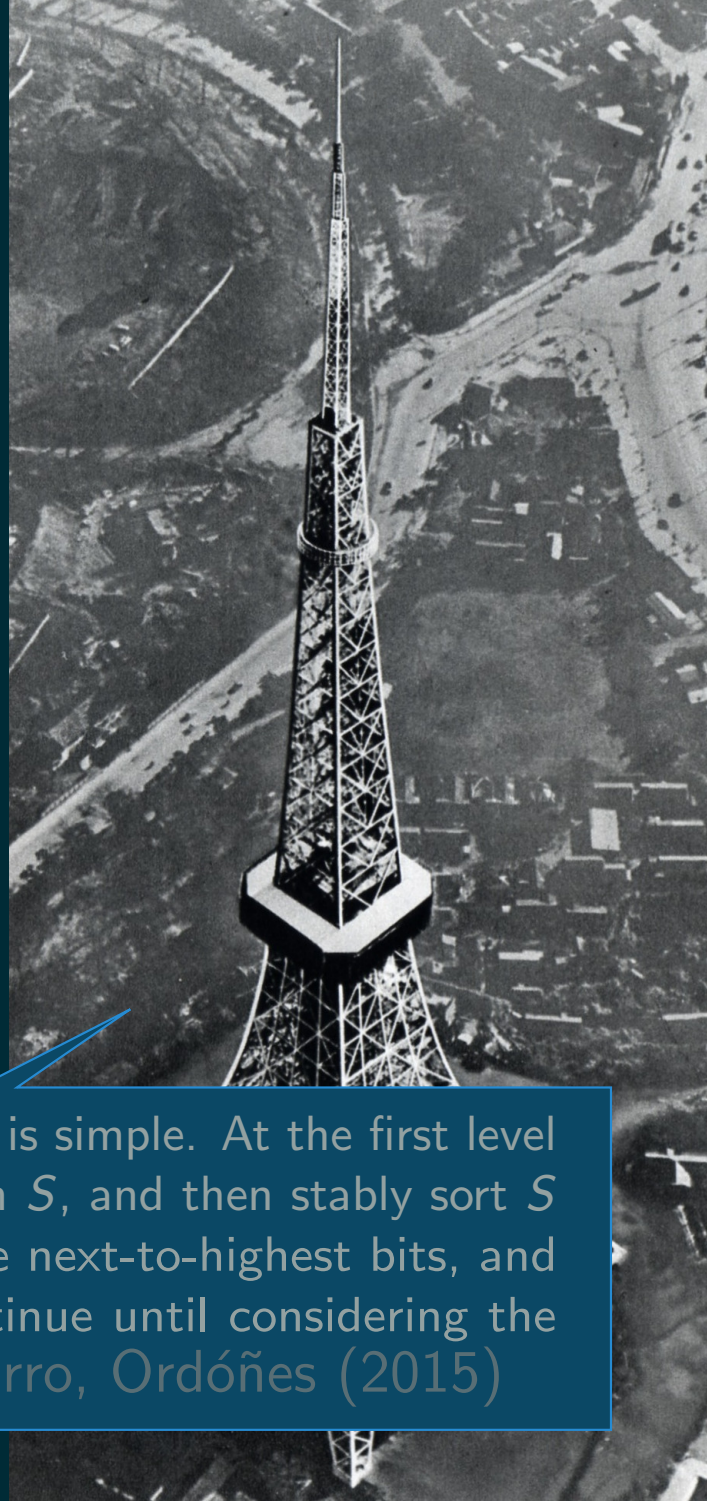
Wavelet Matrix

Similar but Different

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

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

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



Construction: The construction of the wavelet matrix is simple. At the first level we keep in bitmap $\tilde{B}_0$ the highest bit of the symbols in S, and then stably sort S by those highest bits. New we keep in bitmap $\tilde{B}_1$ the next-to-highest bits, and stably sort S by those next-to-highest bits. We continue until considering the lowest bit. This takes $\mathcal{O}(n \lg \sigma)$ time. Claude, Navarro, Ordóñes (2015)

Wavelet Matrix

Considered Bit Prefixes

- Bit-reversal permutation

Wavelet Matrix

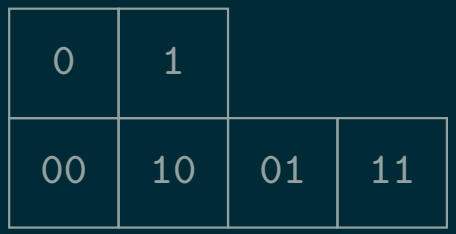
Considered Bit Prefixes

- Bit-reversal permutation

Wavelet Matrix

Considered Bit Prefixes

- Bit-reversal permutation



Wavelet Matrix

Considered Bit Prefixes

- Bit-reversal permutation

0	1						
00	10	01	11				
000	100	010	110	001	101	011	111

Wavelet Matrix

Considered Bit Prefixes

- Bit-reversal permutation

0	1														
00	10	01	11												
000	100	010	110	001	101	011	111								
0000	1000	0100	1100	0010	1010	0110	1110	0001	1001	0101	1101	0011	1011	0111	1111

Wavelet Matrix

Considered Bit Prefixes

- Bit-reversal permutation
- Compute borders in this order

0	1														
00	10	01	11												
000	100	010	110	001	101	011	111								
0000	1000	0100	1100	0010	1010	0110	1110	0001	1001	0101	1101	0011	1011	0111	1111

Wavelet Matrix

Considered Bit Prefixes

- Bit-reversal permutation
- Compute borders in this order

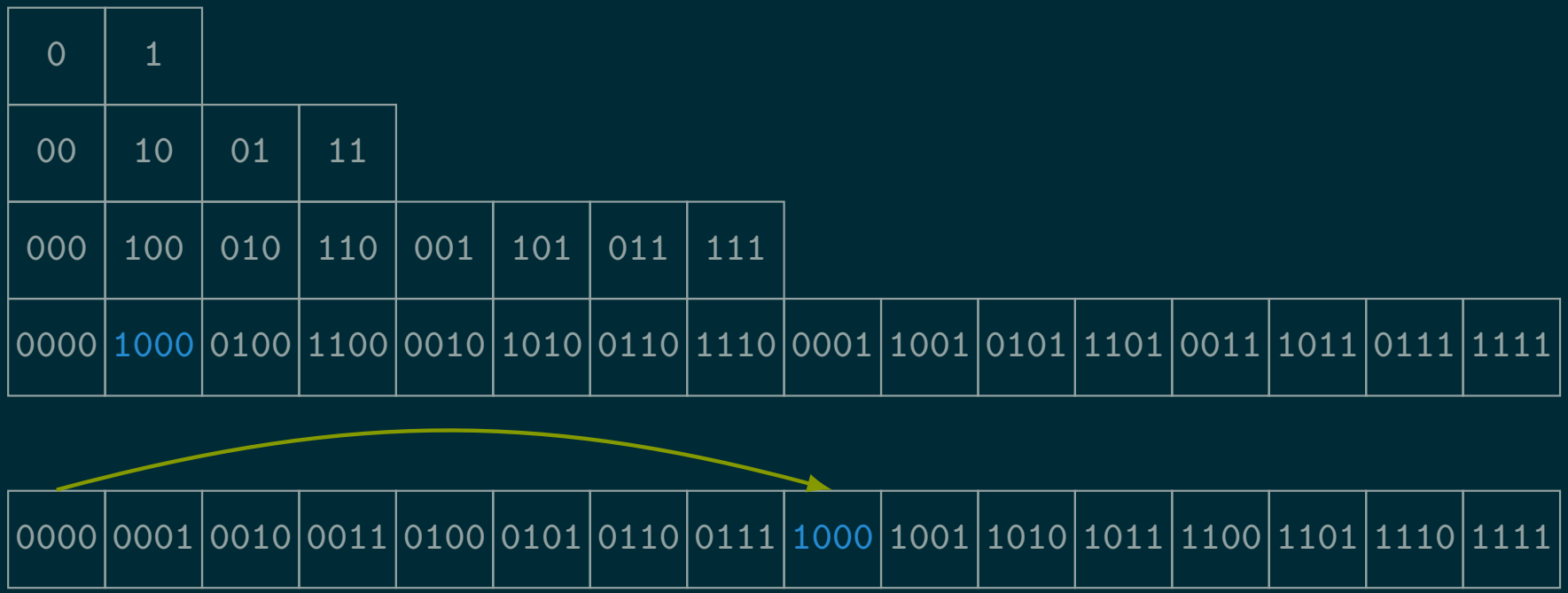
0	1														
00	10	01	11												
000	100	010	110	001	101	011	111								
0000	1000	0100	1100	0010	1010	0110	1110	0001	1001	0101	1101	0011	1011	0111	1111

0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111

Wavelet Matrix

Considered Bit Prefixes

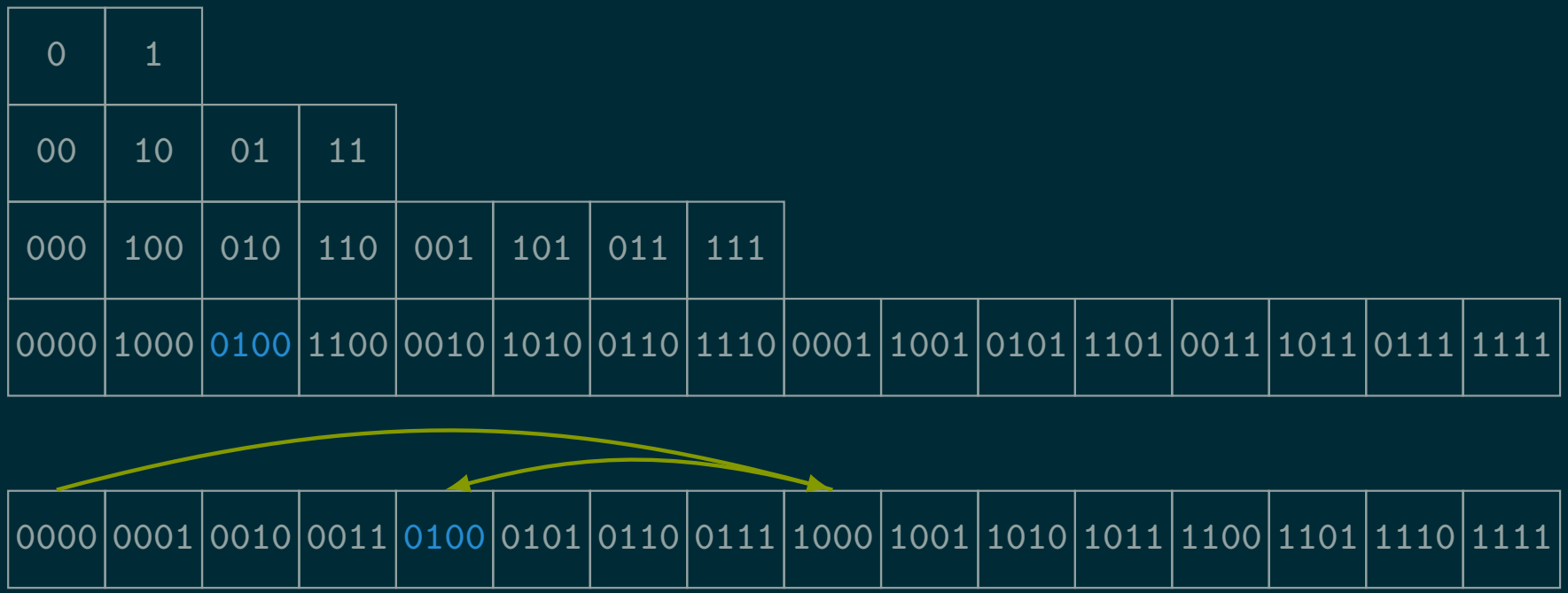
- Bit-reversal permutation
- Compute borders in this order



Wavelet Matrix

Considered Bit Prefixes

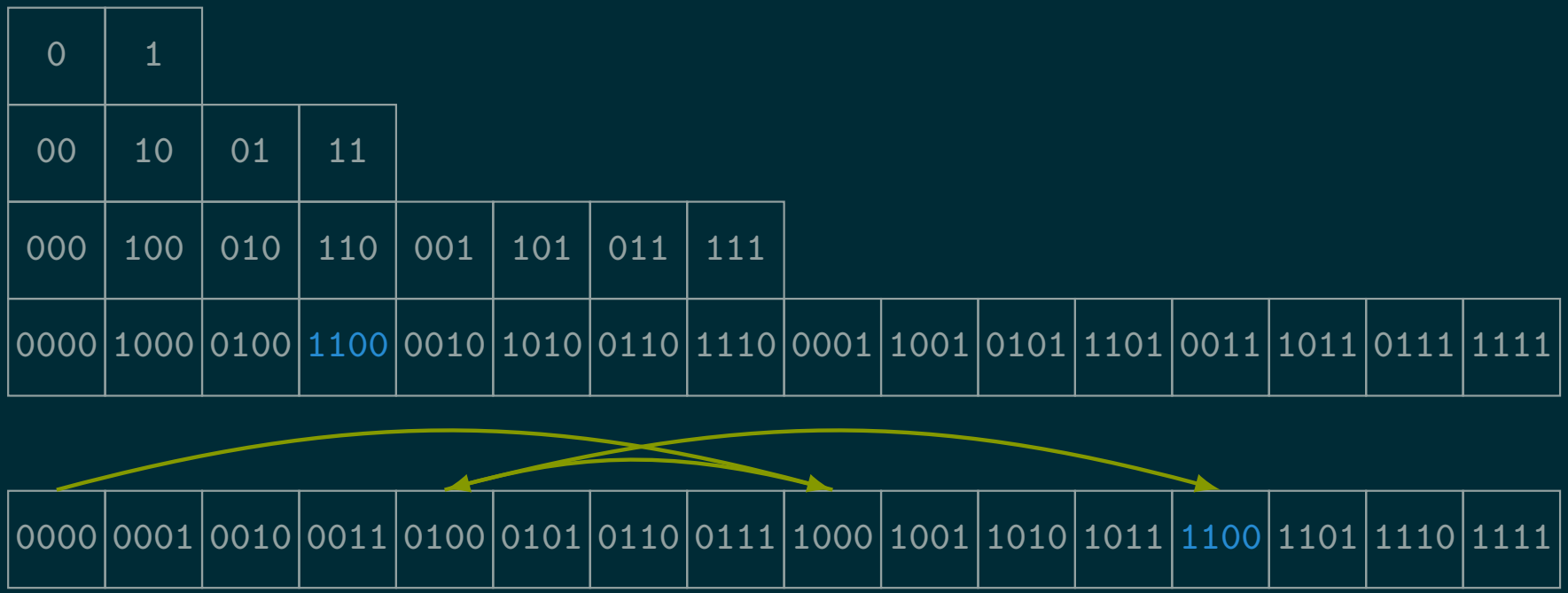
- Bit-reversal permutation
- Compute borders in this order



Wavelet Matrix

Considered Bit Prefixes

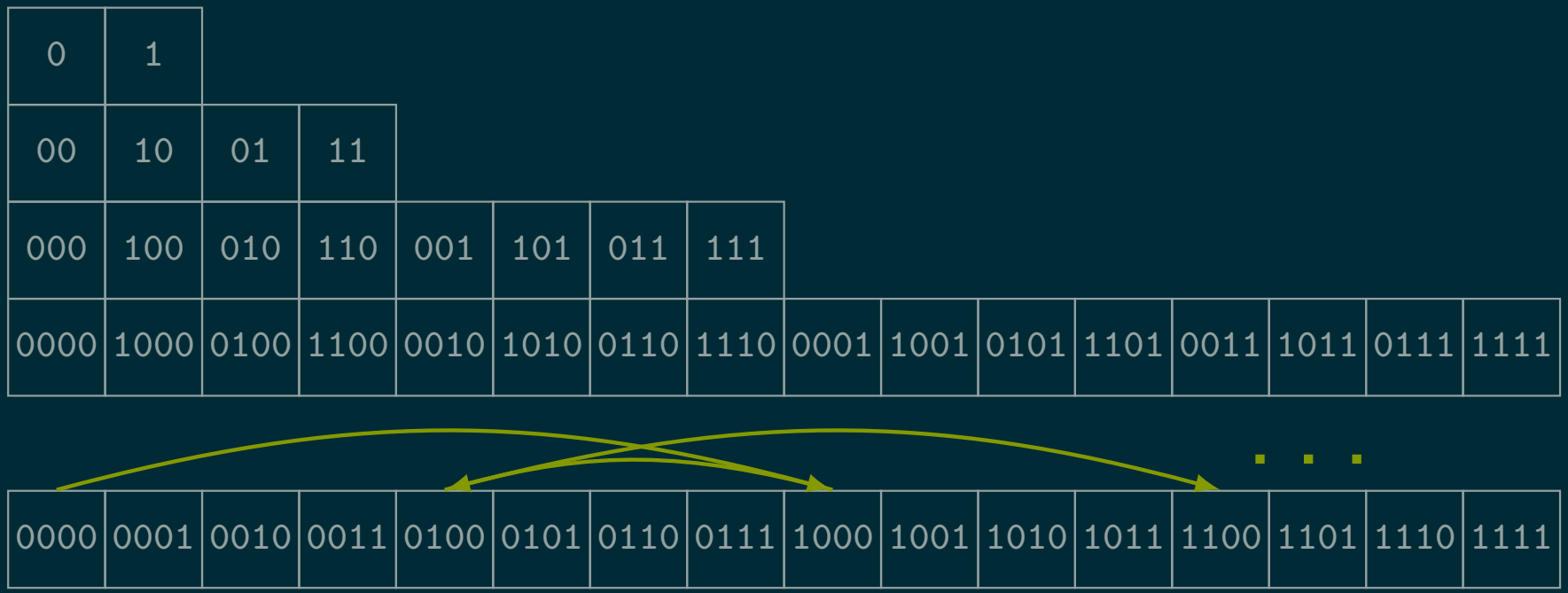
- Bit-reversal permutation
- Compute borders in this order



Wavelet Matrix

Considered Bit Prefixes

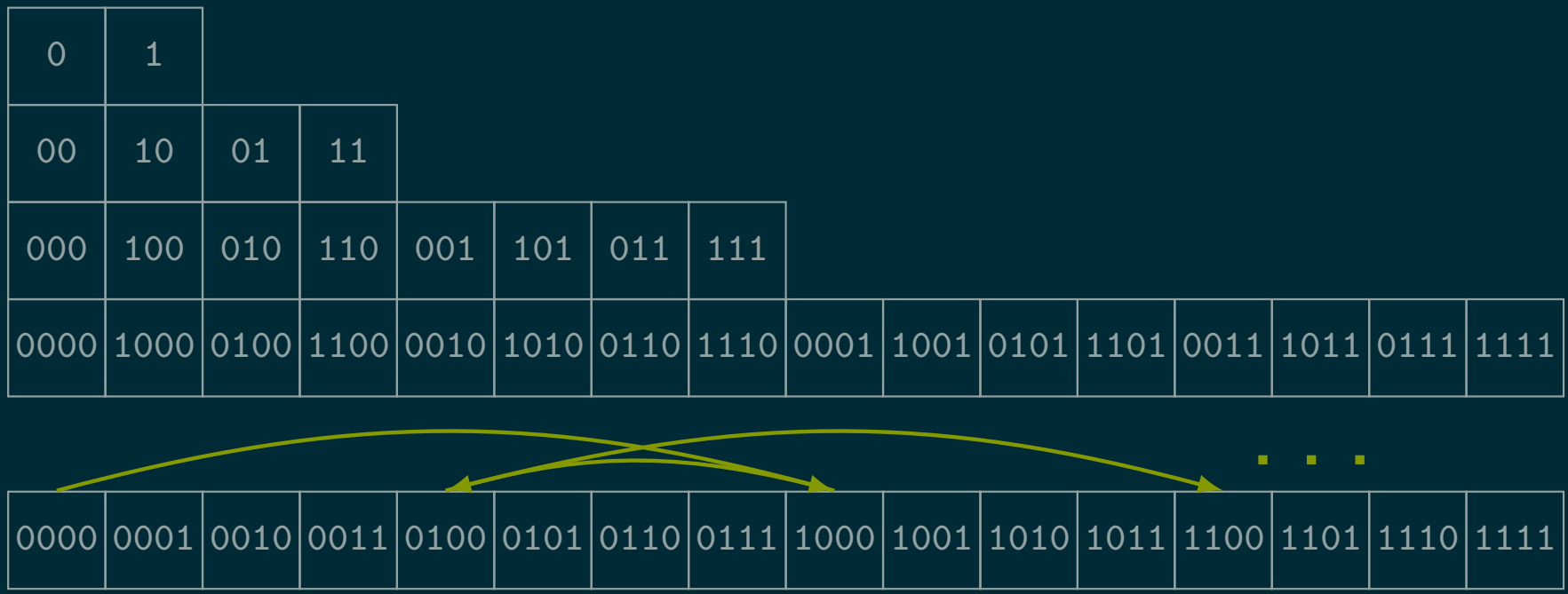
- Bit-reversal permutation
- Compute borders in this order



Wavelet Matrix

Considered Bit Prefixes

- Bit-reversal permutation
- Compute borders in this order



- All algorithms presented can also compute the Wavelet Matrix

Wavelet Tree vs. Wavelet Matrix

Wavelet Tree vs. Wavelet Matrix

From the Tree to the Matrix

Wavelet Tree vs. Wavelet Matrix

From the Tree to the Matrix

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

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

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

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

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

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

Wavelet Tree vs. Wavelet Matrix

From the Tree to the Matrix

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

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

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

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

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

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

Wavelet Tree vs. Wavelet Matrix

From the Tree to the Matrix

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

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

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

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

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

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

1	0	1	1	0	1	0	1	0	1	0	1	0	1	1	0	1
1	1	2	3	2	4	3	5	4	6	5	7	6	8	9	7	10

Wavelet Tree vs. Wavelet Matrix

From the Tree to the Matrix

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

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

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

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

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

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

1	0	1	1	0	1	0	1	0	1	0	1	0	1	1	0	1
1	1	2	3	2	4	3	5	4	6	5	7	6	8	9	7	10

Utility can be constructed in $\mathcal{O}(n + \sigma)$ time.

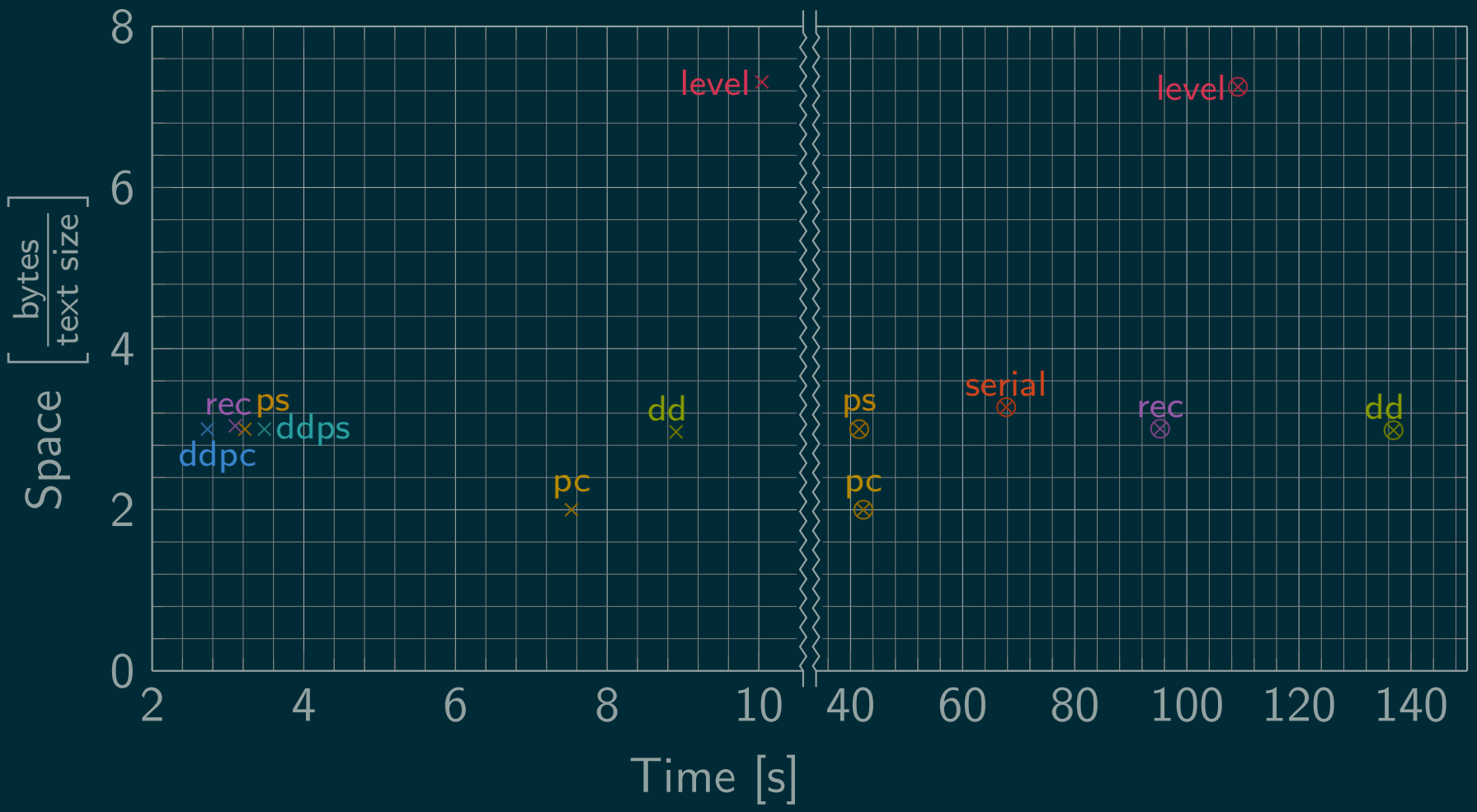
Experiments

- WT-construction on 1 and 32 cores
- Comparing speed and memory usage
- Comparing with:

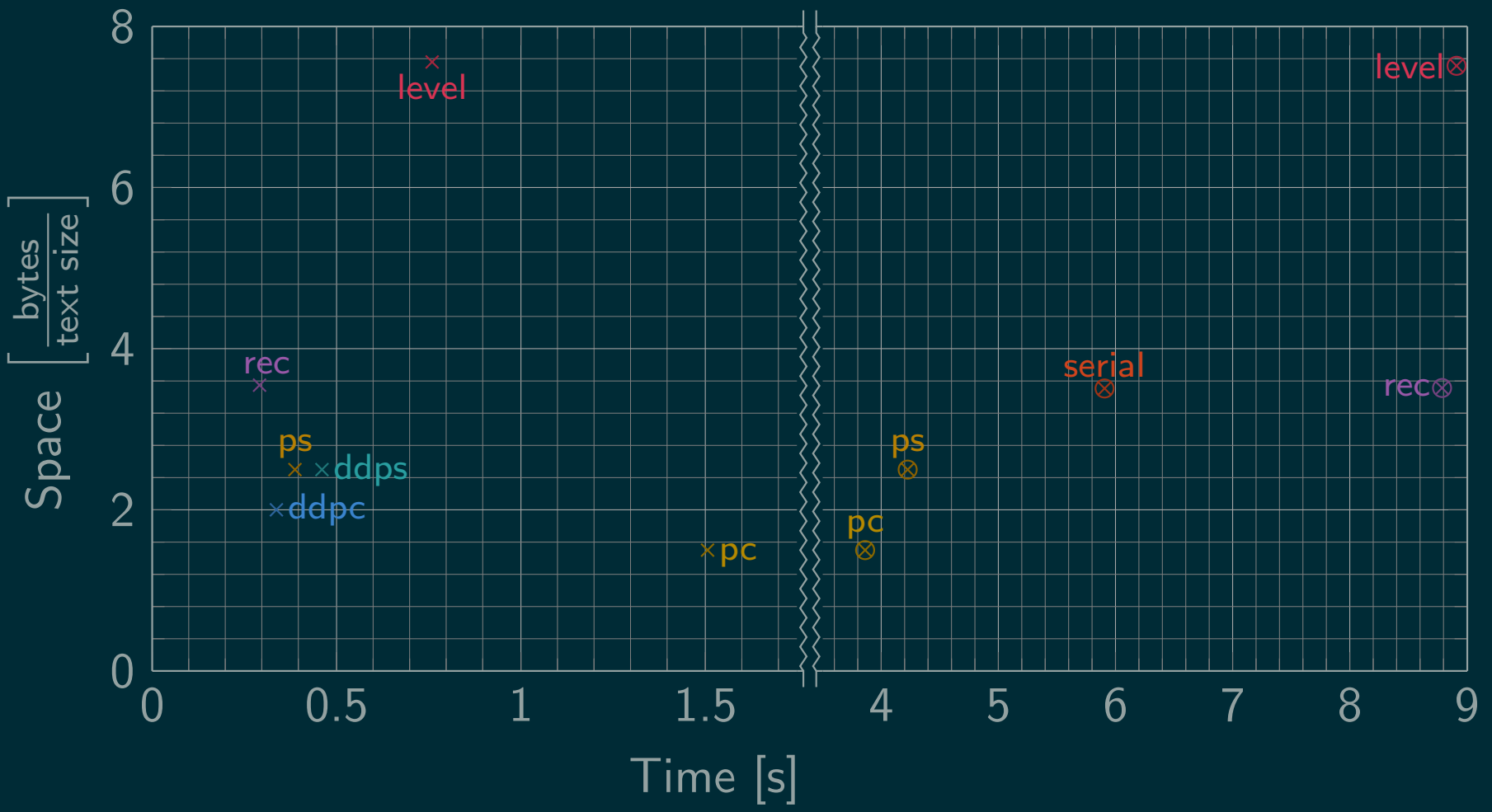
rec	Fastest parallel WT-construction [Labeit et al., 2016]
level	Parallel WT-construction [Shun, 2015]
serial	Fastest sequential WT-construction [Shun, 2015]
dd	Memory efficient WT-construction [Fuentes-Sepúlveda et al., 2017]

- Results for WM-construction are similar

ENG: 2.21 GB & $\sigma = 239$



DNA: 400 MB & $\sigma = 16$



Wrapping Up

Conclusion

- New parallel algorithms for the Wavelet Tree
- First practical parallel algorithms for the Wavelet Matrix
- Simple, fast and lightweight

Wrapping Up

Conclusion

- New parallel algorithms for the Wavelet Tree
- First practical parallel algorithms for the Wavelet Matrix
- Simple, fast and lightweight

Current & Future Work

- *Huffman-shaped* WT/WMs
- Construction in distributed memory

Wrapping Up

Conclusion

- New parallel algorithms for the Wavelet Tree
- First practical parallel algorithms for the Wavelet Matrix
- Simple, fast and lightweight

Current & Future Work

- *Huffman-shaped* WT/WMs
- Construction in distributed memory

Thank You!