Parallel Wavelet Tree Construction

Simple, Fast amd Lightweight

Johannes Fischer *Florian Kurpicz*

Rank and Select

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

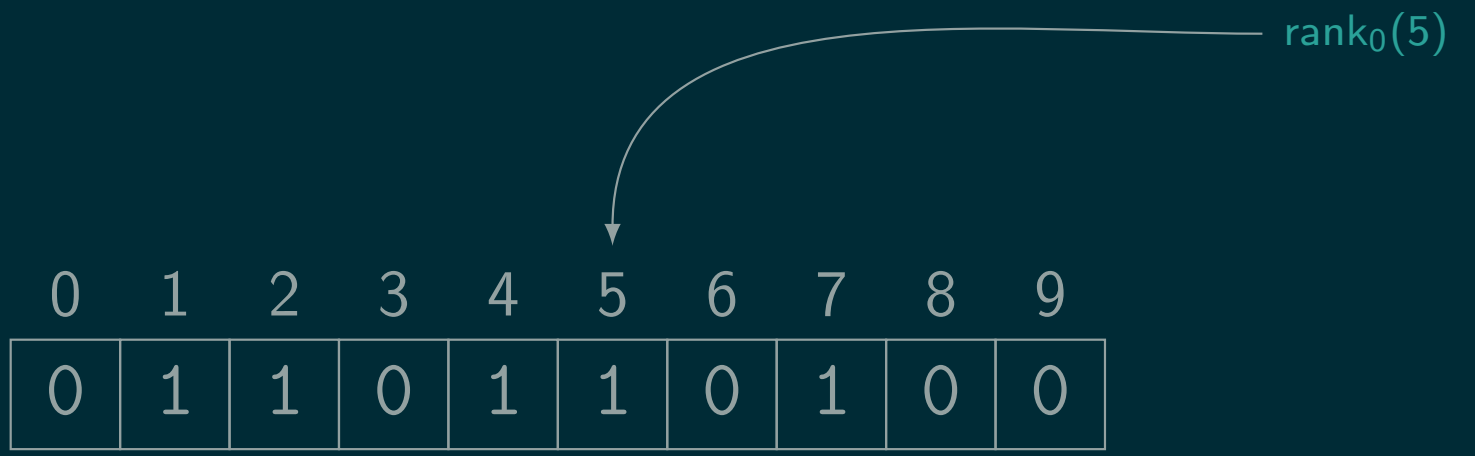
$\text{rank}_0(5)$

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 α before position i

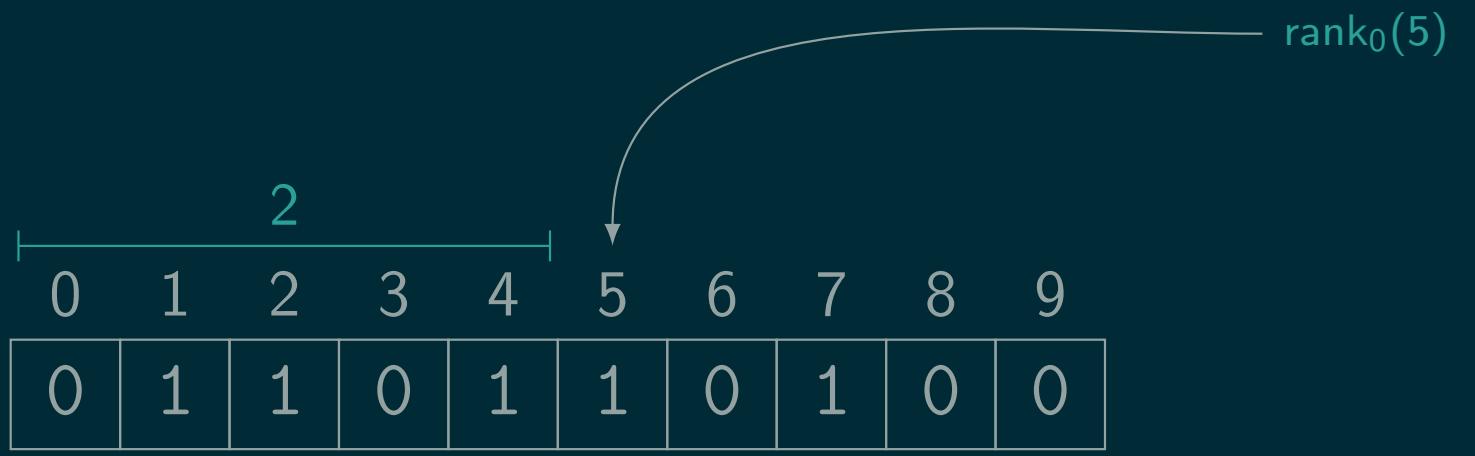
$\text{select}_\alpha(j)$ position of j-th α



Rank and Select

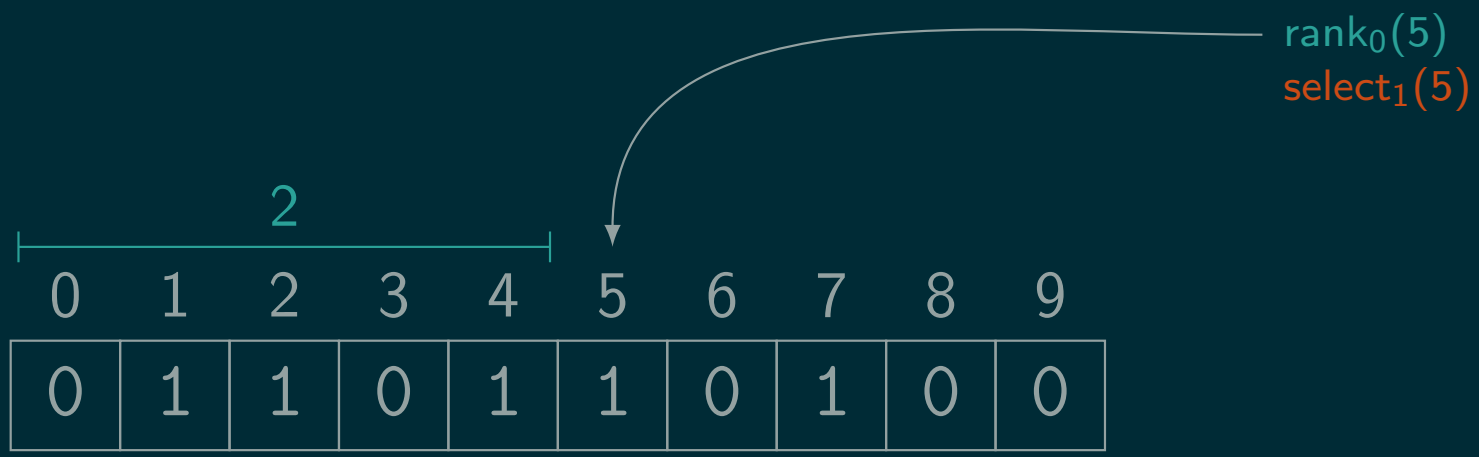
$\mathrm{rank}_\alpha(i)$ # of α before position i

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



Rank and Select

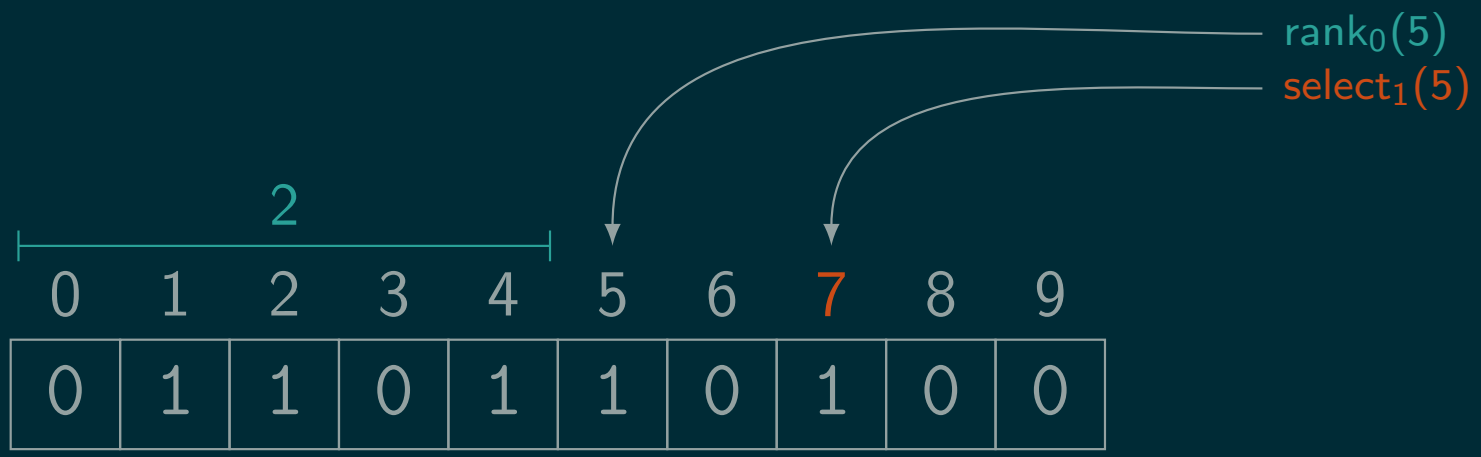
$\text{rank}_\alpha(i)$ # of α before position i
$\text{select}_\alpha(j)$ position of j-th α



Rank and Select

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

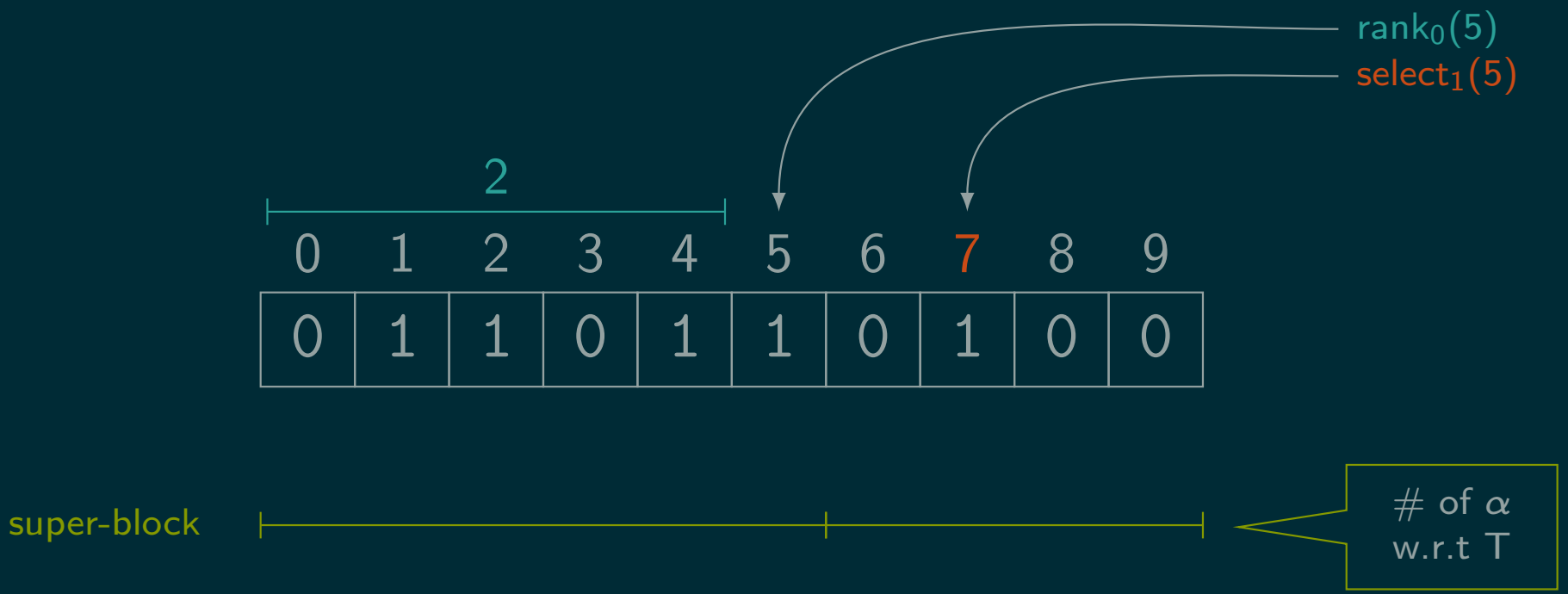
$\text{select}_\alpha(j)$ position of j-th α



Rank and Select

$\mathrm{rank}_\alpha(i)$ # of α before position i

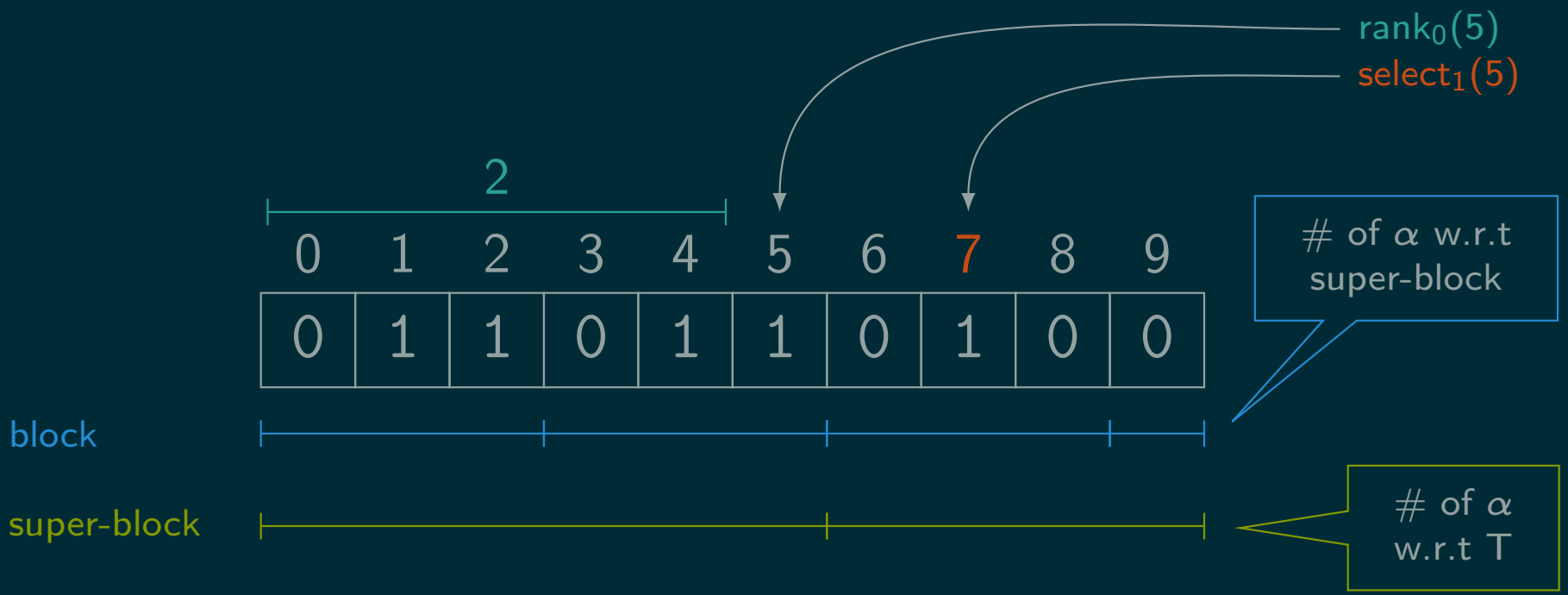
$\mathrm{select}_\alpha(j)$ position of j-th α



Rank and Select

$\mathrm{rank}_\alpha(i)$ # of α before position i

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



Rank and Select

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

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

Rank and Select

$\text{rank}_\alpha(i)$ # of α 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

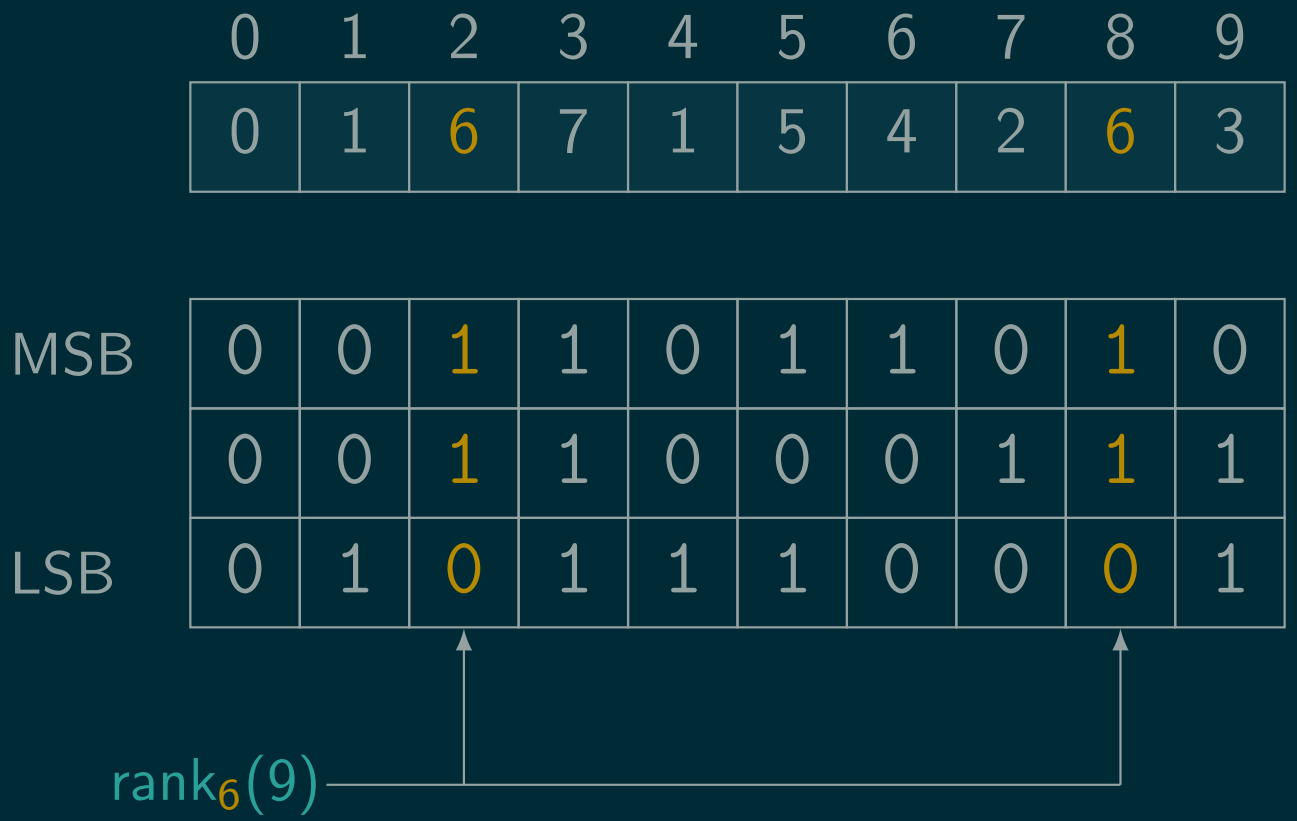
$$\Sigma = \{0, 1, 2, 3, 4, 5, 6, 7\}$$

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

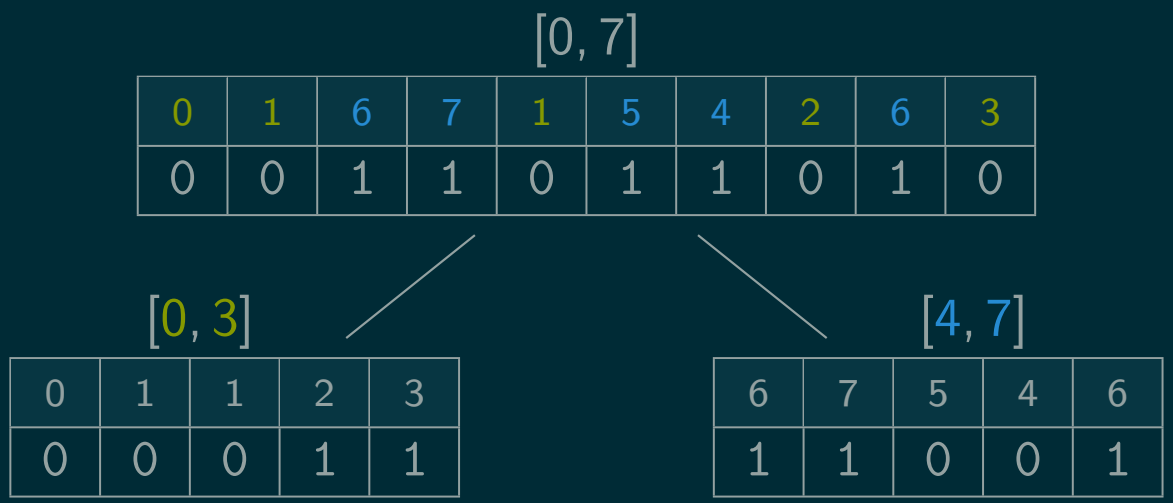
$[0, 7]$

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

Wavelet Tree

Pointer-Based

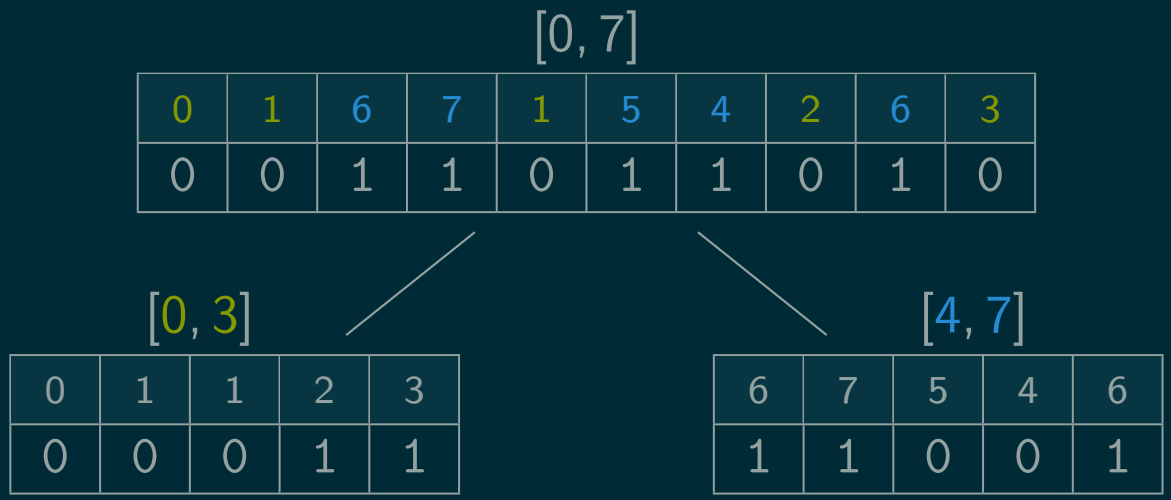
0	1	6	7	1	5	4	2	6	3
0	0	1	1	0	1	1	0	1	0
0	0	1	1	0	0	0	1	1	1
0	1	0	1	1	1	0	0	0	1



Wavelet Tree

Pointer-Based

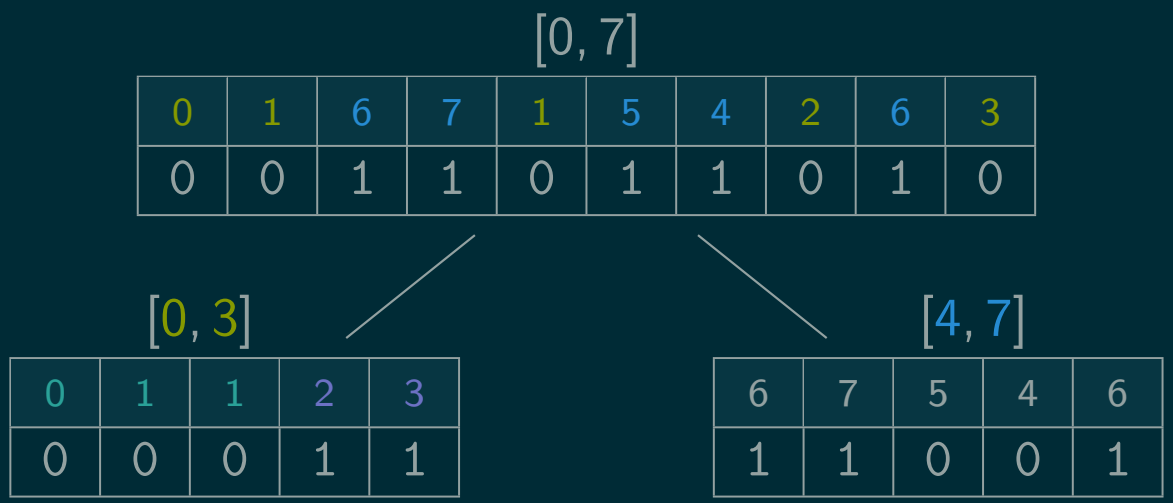
0	1	6	7	1	5	4	2	6	3
0	0	1	1	0	1	1	0	1	0
0	0	1	1	0	0	0	1	1	1
0	1	0	1	1	1	0	0	0	1



Wavelet Tree

Pointer-Based

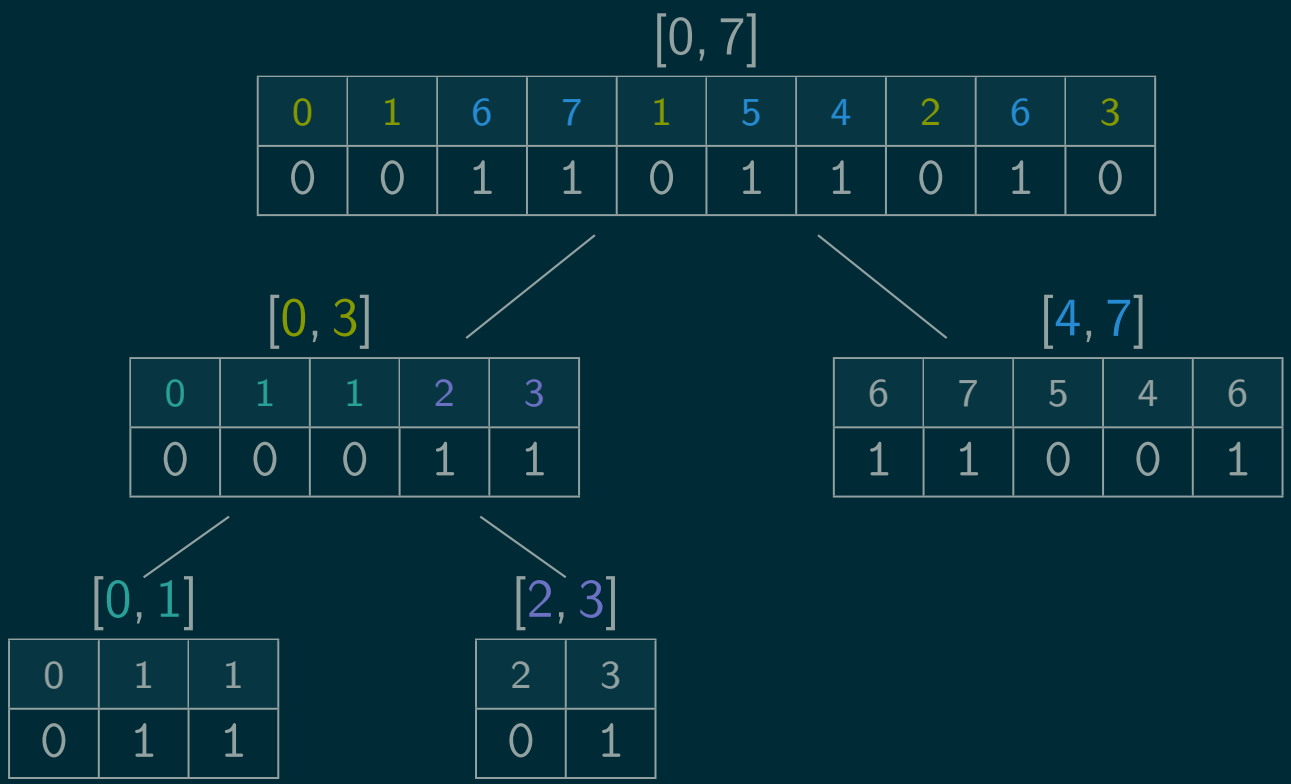
0	1	6	7	1	5	4	2	6	3
0	0	1	1	0	1	1	0	1	0
0	0	1	1	0	0	0	1	1	1
0	1	0	1	1	1	0	0	0	1



Wavelet Tree

Pointer-Based

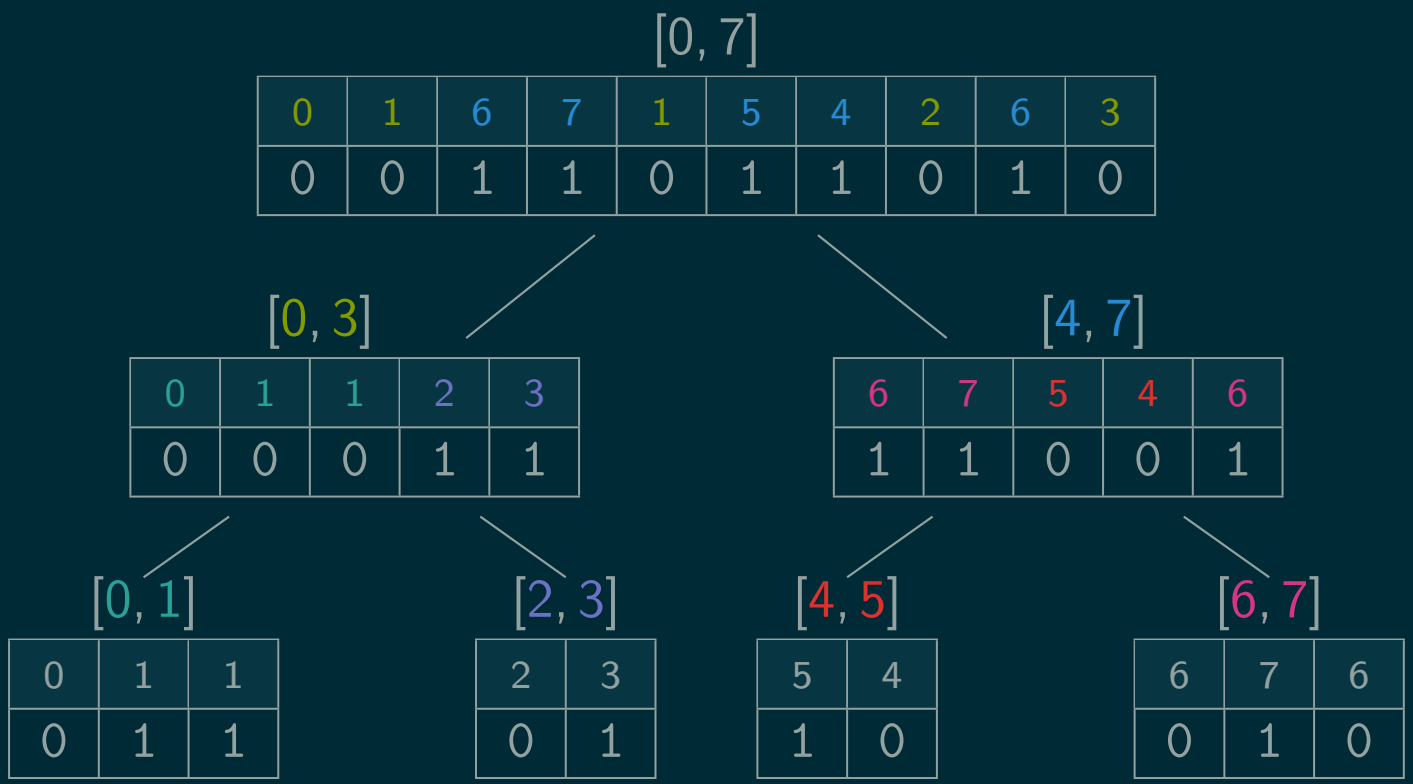
0	1	6	7	1	5	4	2	6	3
0	0	1	1	0	1	1	0	1	0
0	0	1	1	0	0	0	1	1	1
→ 0	1	0	1	1	1	0	0	0	1



Wavelet Tree

Pointer-Based

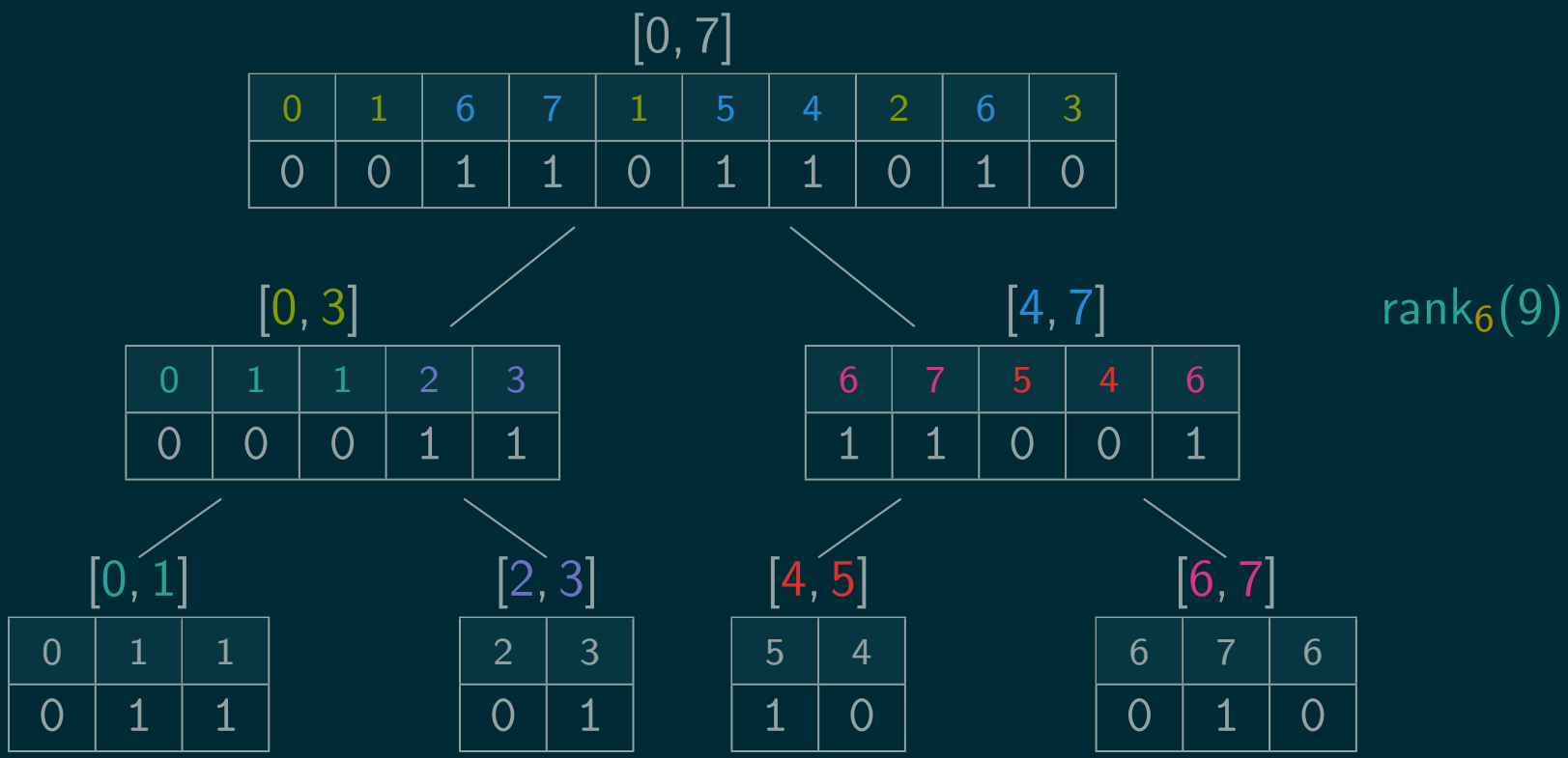
0	1	6	7	1	5	4	2	6	3
0	0	1	1	0	1	1	0	1	0
0	0	1	1	0	0	0	1	1	1
→ 0	1	0	1	1	1	0	0	0	1



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

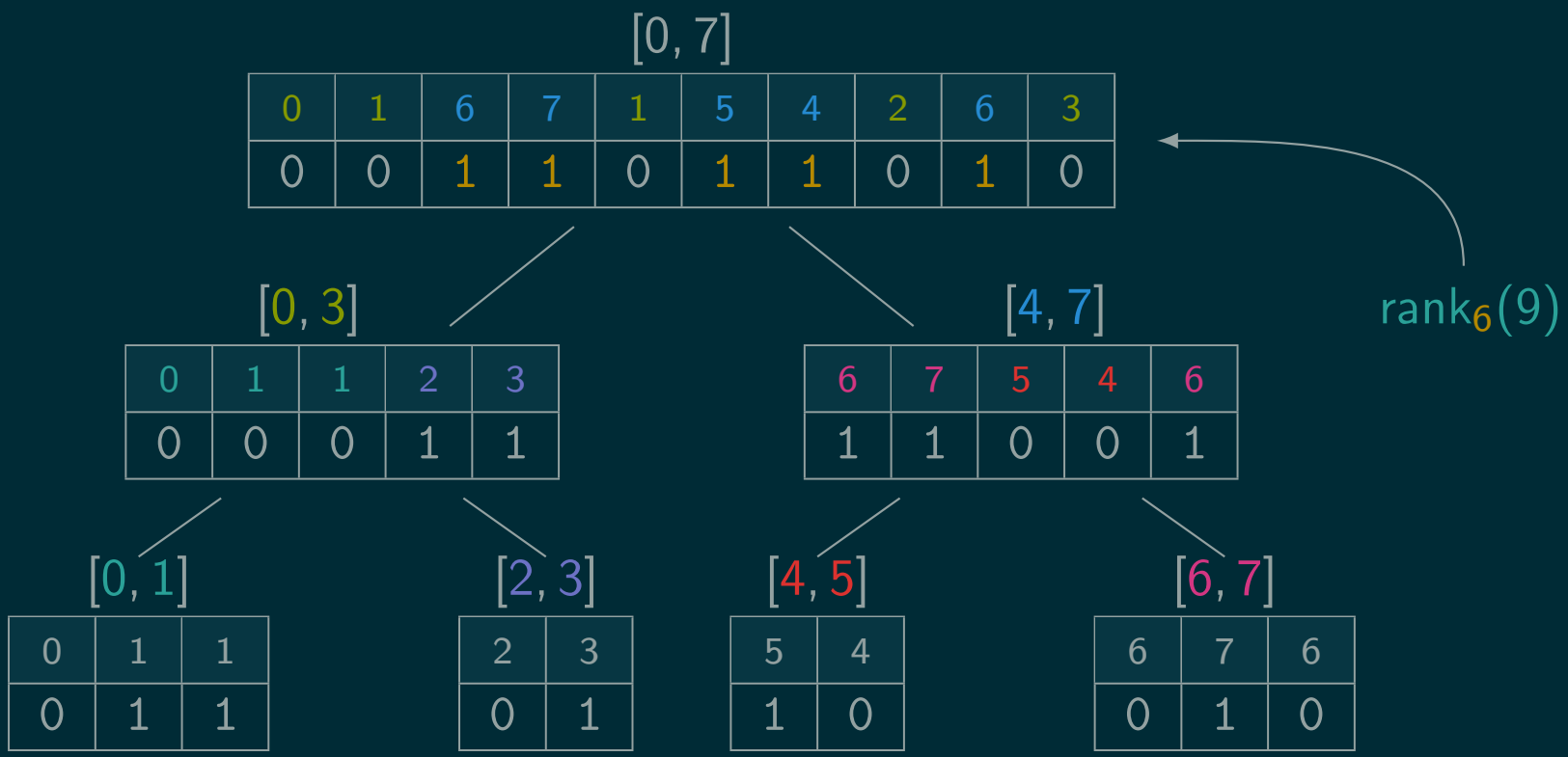


$\text{rank}_6(9)$

Wavelet Tree

Pointer-Based

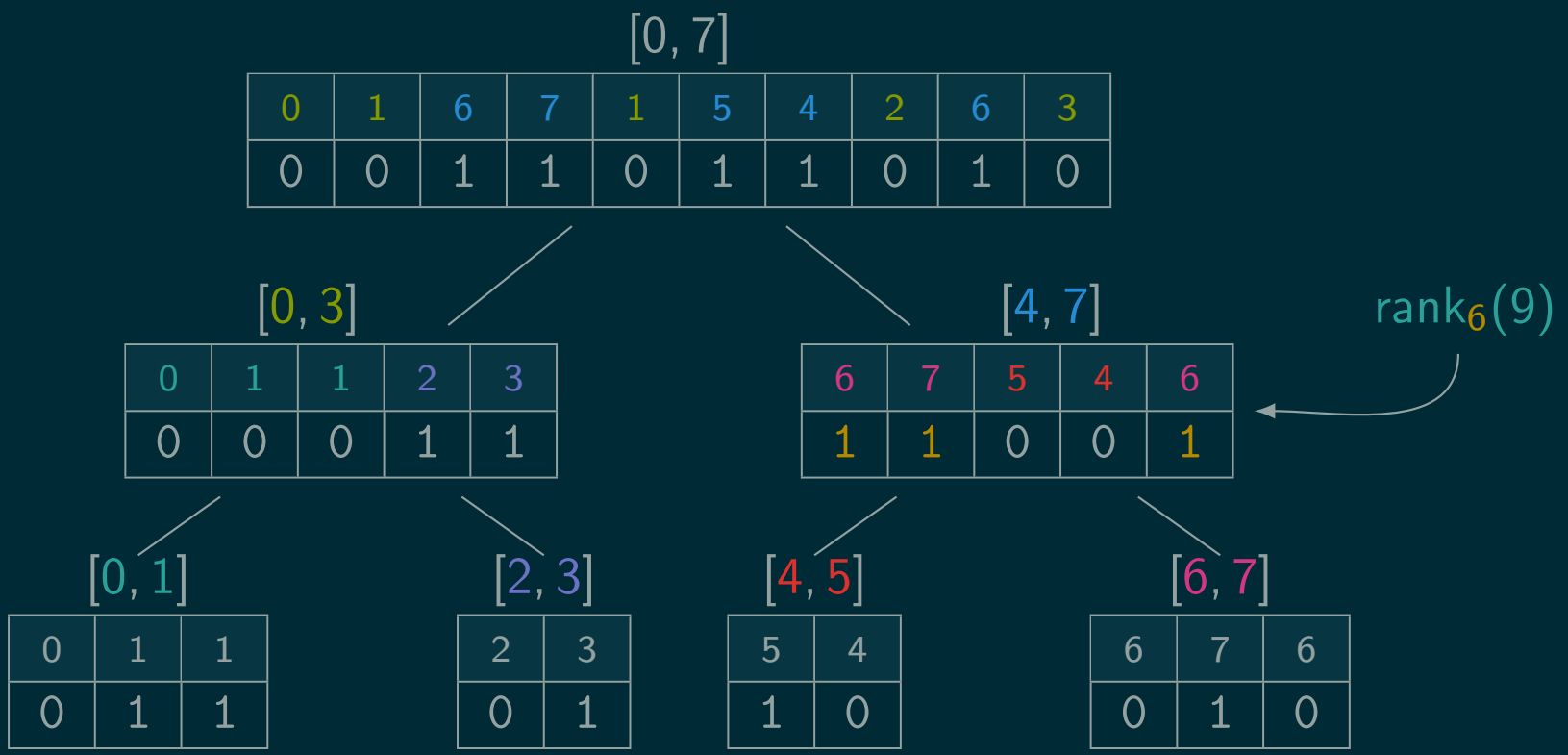
0	1	6	7	1	5	4	2	6	3
0	0	1	1	0	1	1	0	1	0
0	0	1	1	0	0	0	1	1	1
0	1	0	1	1	1	0	0	0	1



Wavelet Tree

Pointer-Based

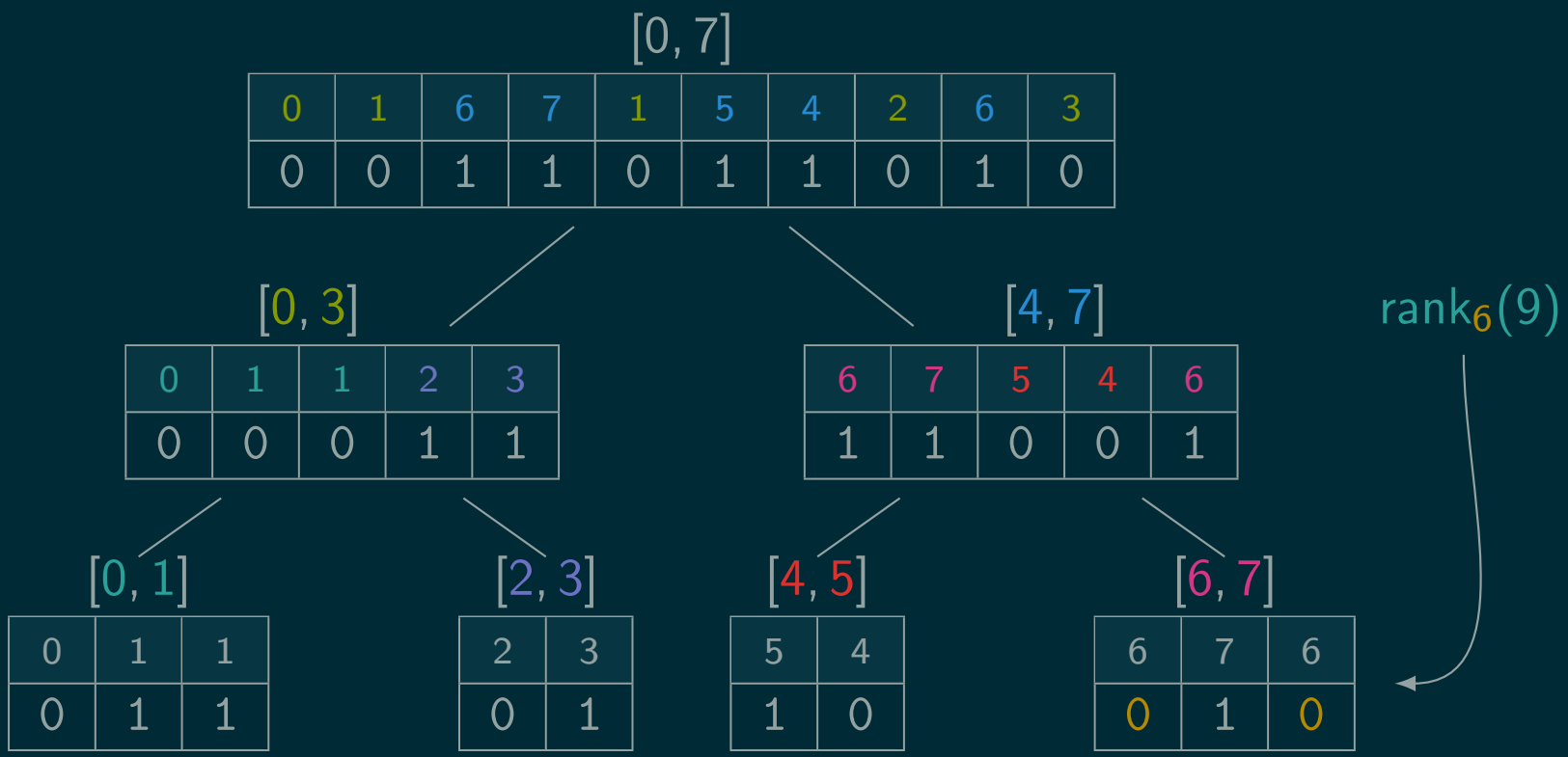
0	1	6	7	1	5	4	2	6	3
0	0	1	1	0	1	1	0	1	0
0	0	1	1	0	0	0	1	1	1
0	1	0	1	1	1	0	0	0	1



Wavelet Tree

Pointer-Based

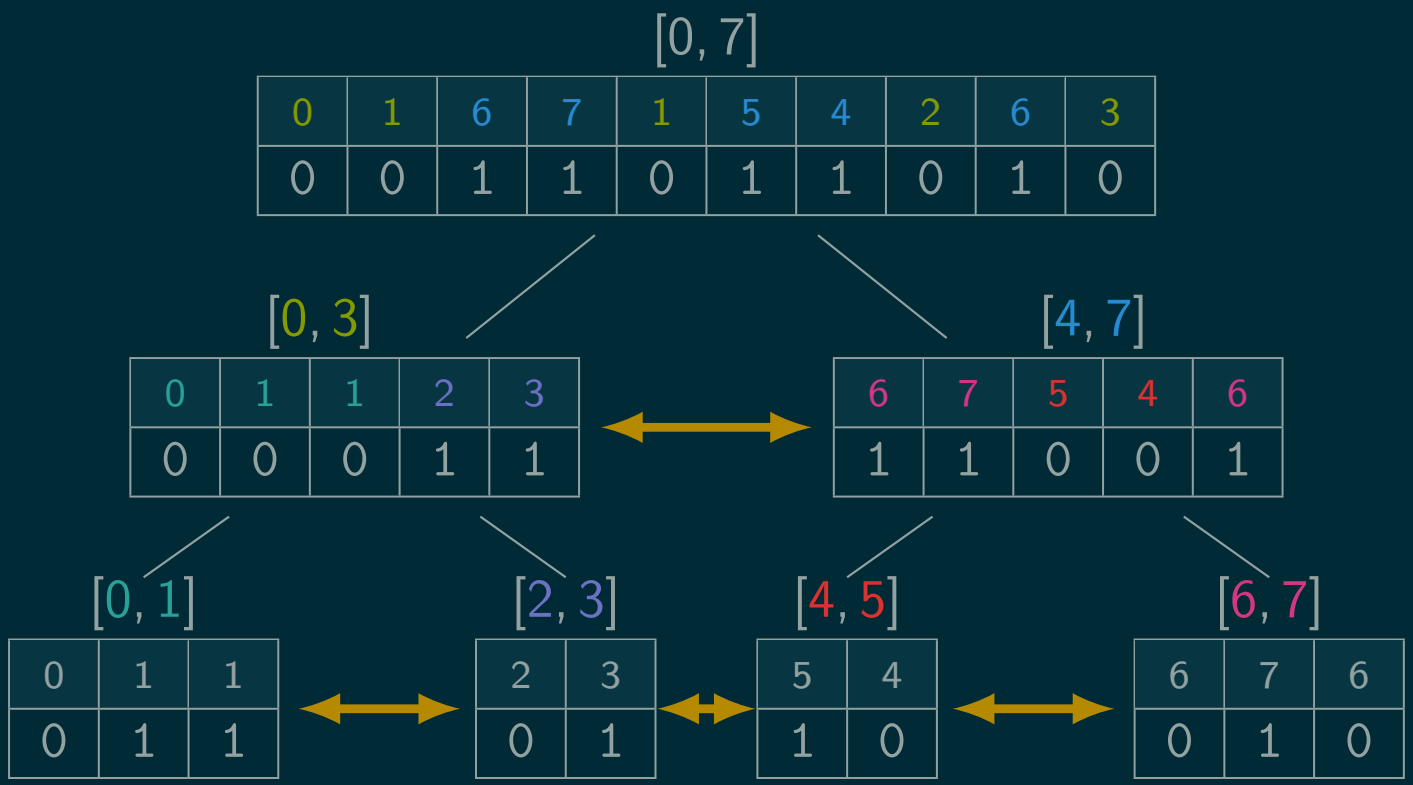
0	1	6	7	1	5	4	2	6	3
0	0	1	1	0	1	1	0	1	0
0	0	1	1	0	0	0	1	1	1
0	1	0	1	1	1	0	0	0	1



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

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

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

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

MSB of each character

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

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

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

MSB of each character

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

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

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

MSB of each character

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

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

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

MSB of each character

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

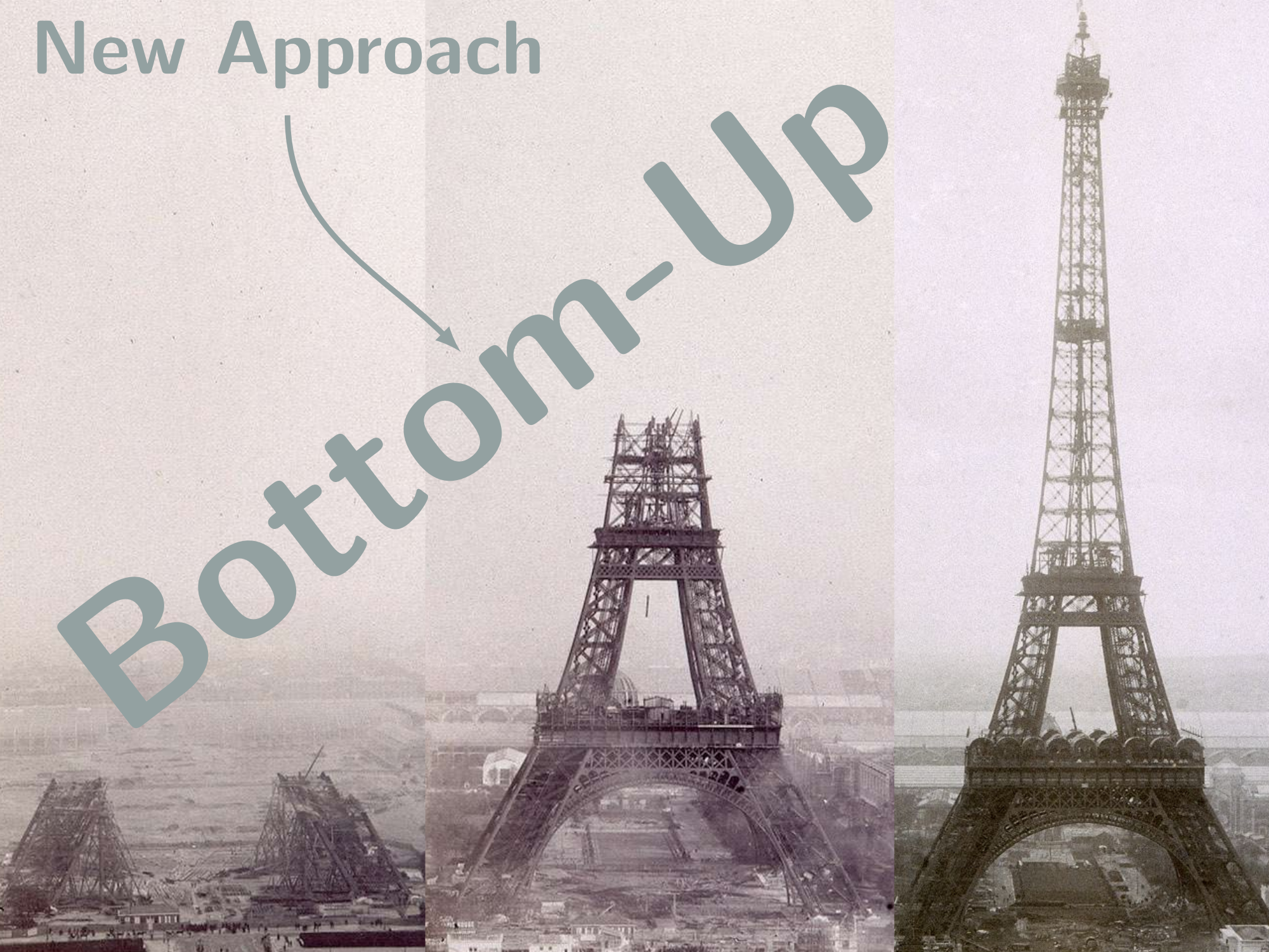
0 | 1

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

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

$\ell = 1$

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
$11 \rightarrow [6, 7]$	1

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

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

$\ell = 1$

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

$00 \rightarrow [0, 1]$	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

$\ell = 0$

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

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

$\ell = 1$

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

$00 \rightarrow [0, 1]$	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

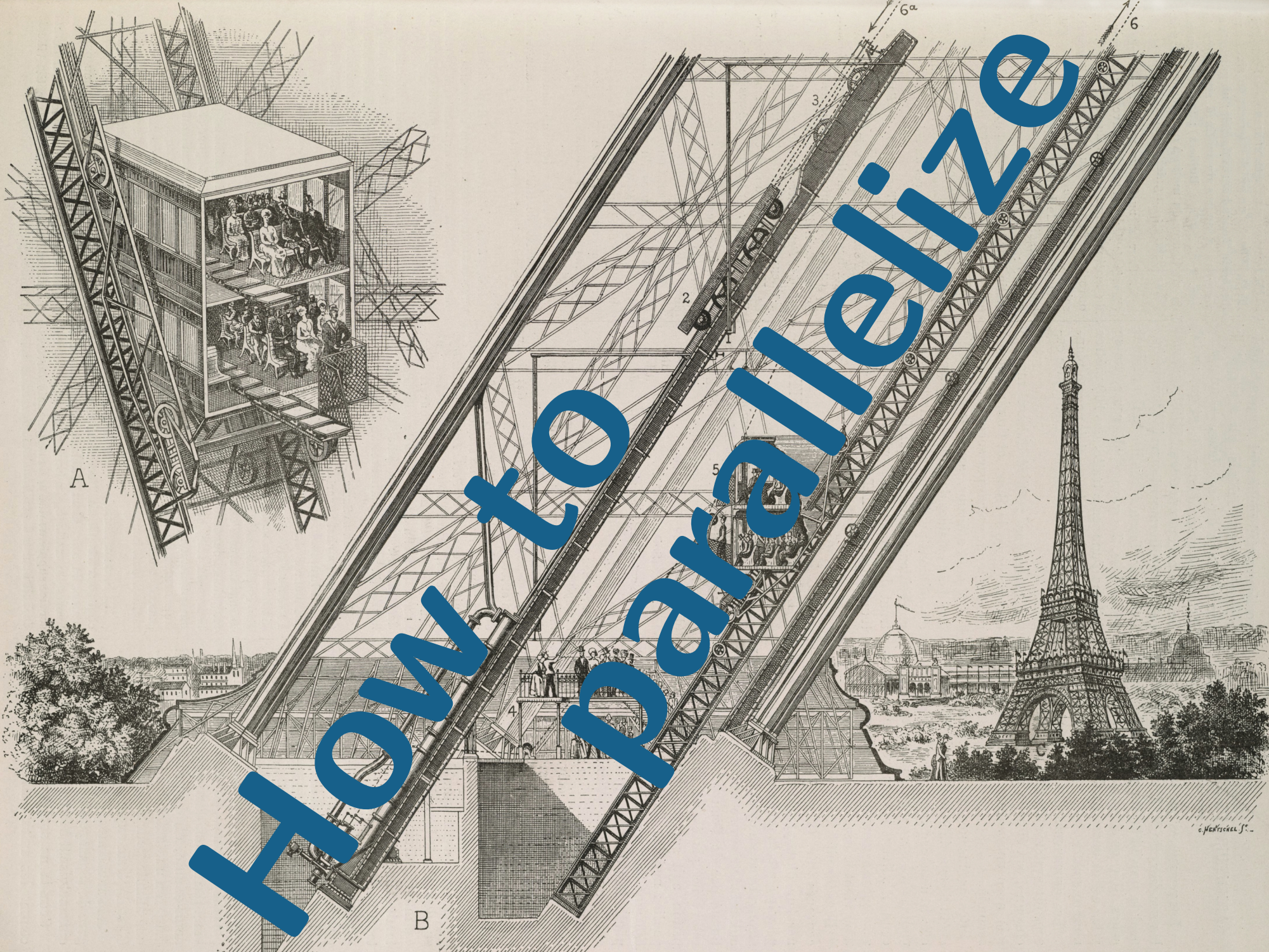
Compute the borders for the **previous** level without rescanning T.

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

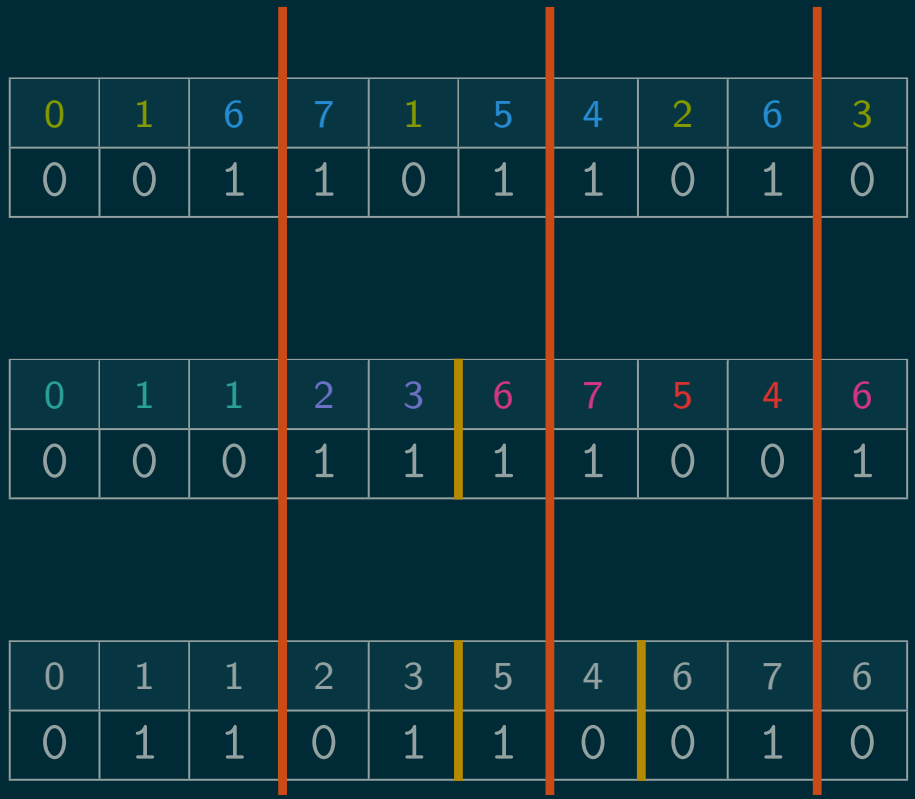
0	1	6	7	1	5	4	2	6	3
0	0	1	1	0	1	1	0	1	0

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

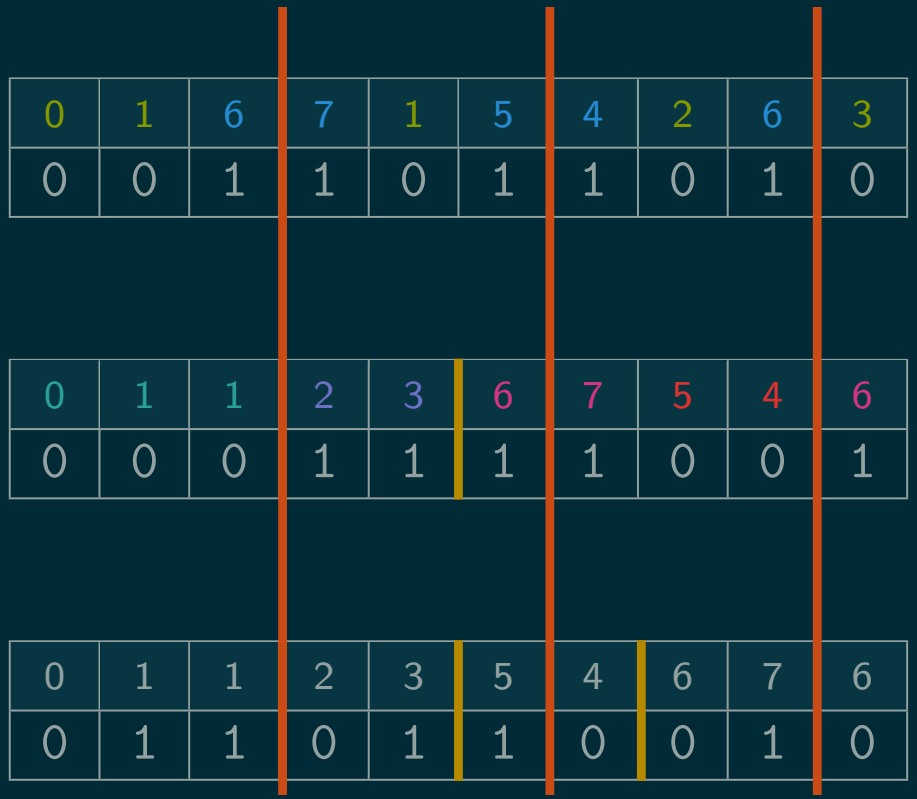
Prefix Sorting

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

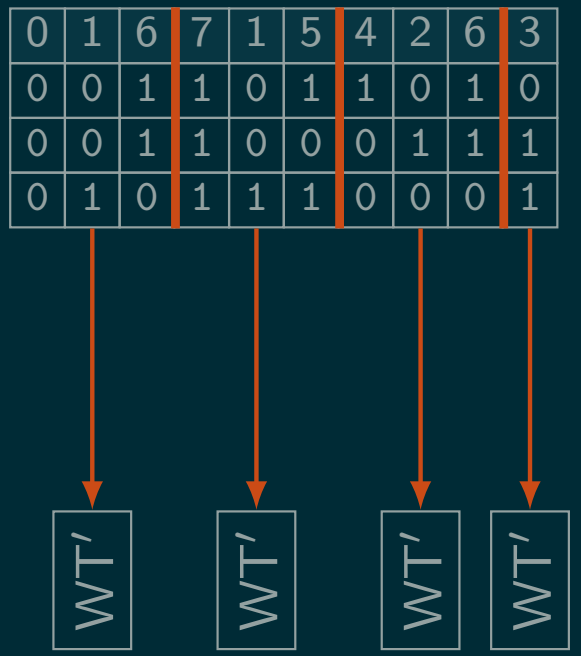
0	1	6	7	1	5	4	2	6	3
0	0	1	1	0	1	1	0	1	0
0	0	1	1	0	0	0	1	1	1
0	1	0	1	1	1	0	0	0	1

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

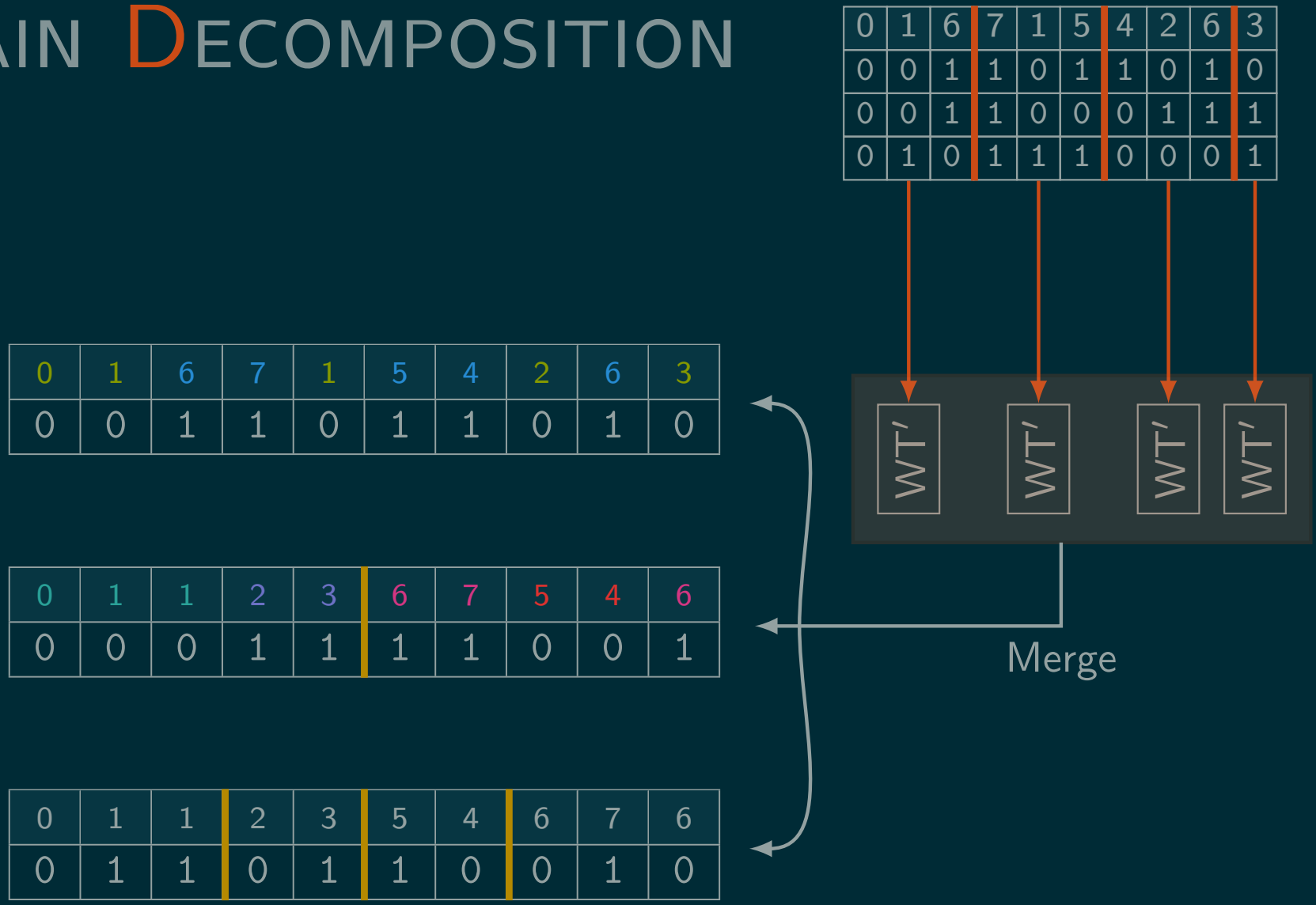


0	1	6	7	1	5	4	2	6	3
0	0	1	1	0	1	1	0	1	0

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



Recap
Parallelization

- Prefix Counting is simple but not scaling

Recap
Parallelization

- Prefix Counting is simple but not scaling
- Prefix Sorting requires more space but scales

Recap
Parallelization

- Prefix Counting is simple but not scaling
- Prefix Sorting requires more space but scales
- Domain Decomposition scales and is flexible (but needs more space)

Wavelet Matrix

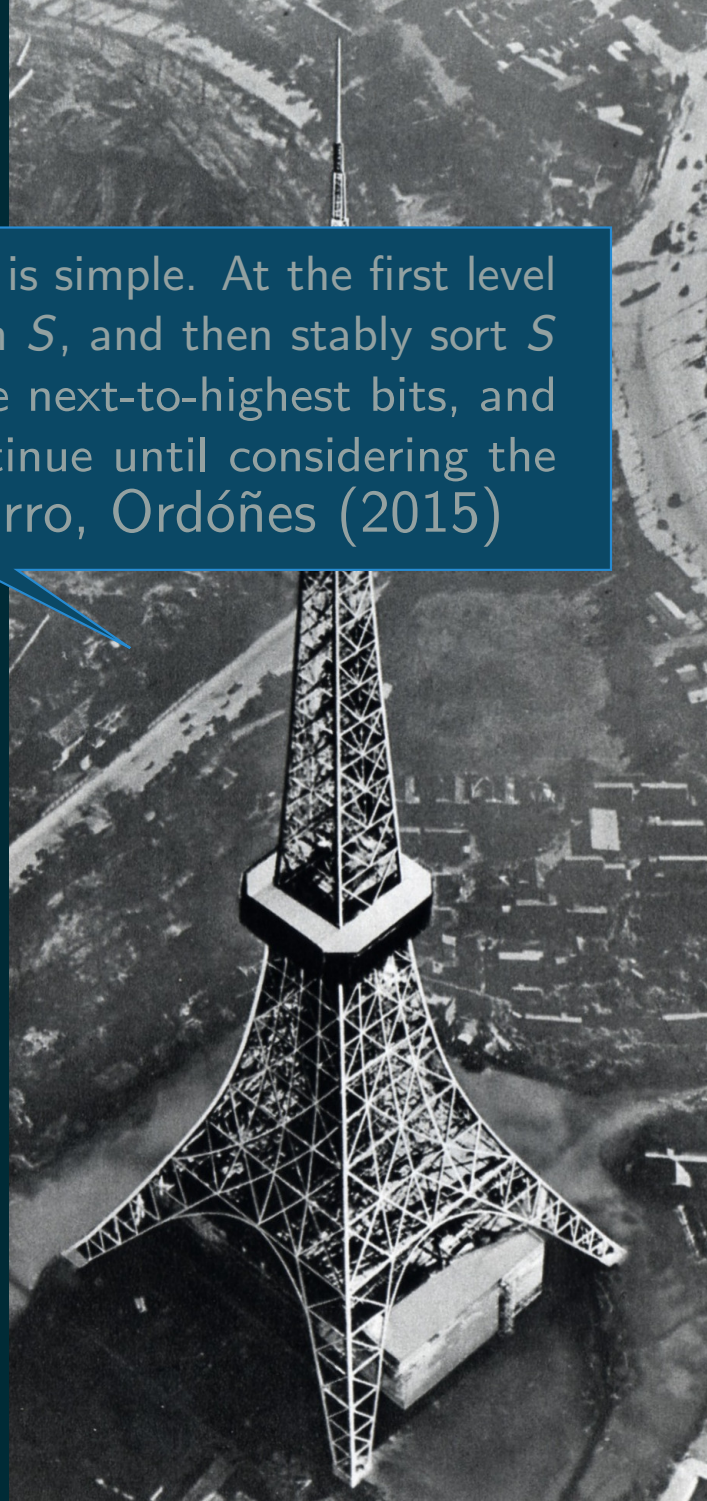
Wavelet Matrix

Similar but Different

Wavelet Matrix

Similar but Different

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

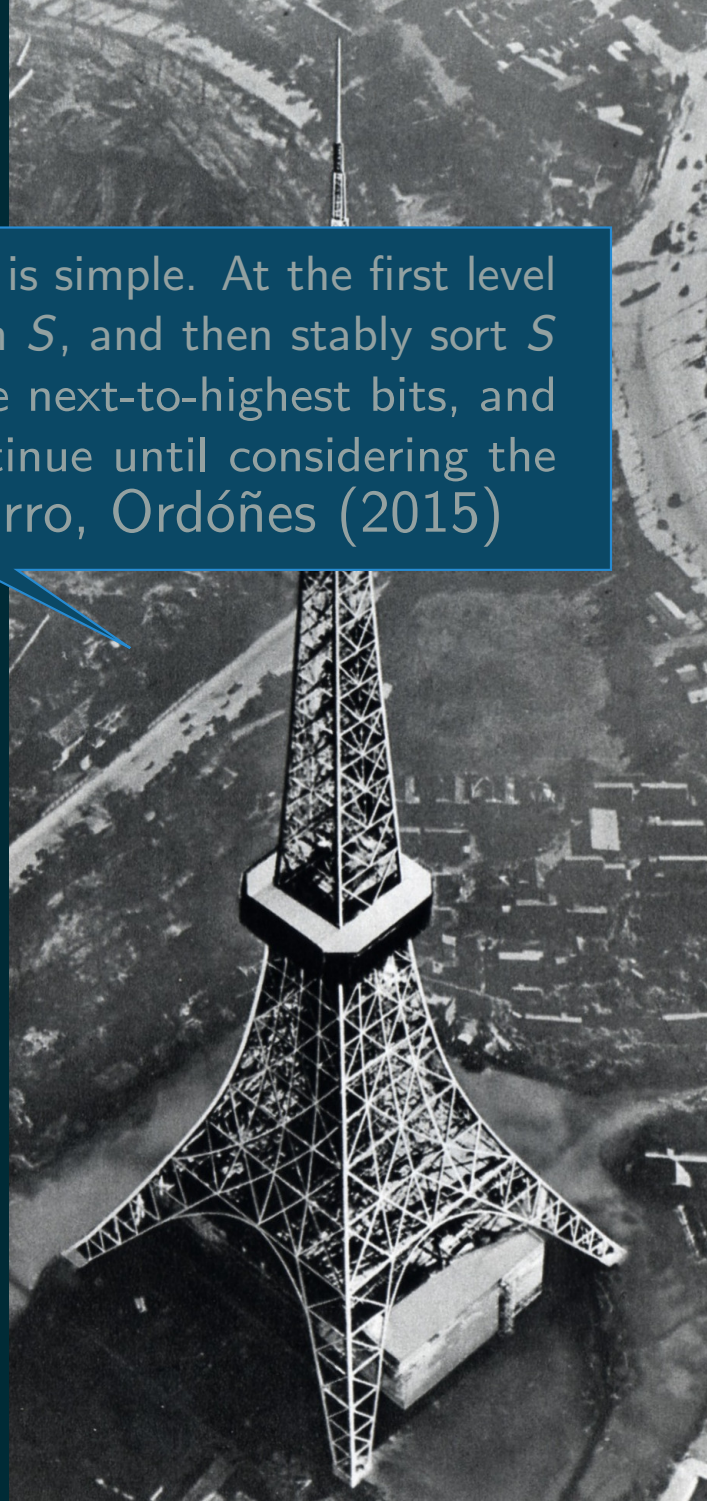
Similar but Different

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)

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

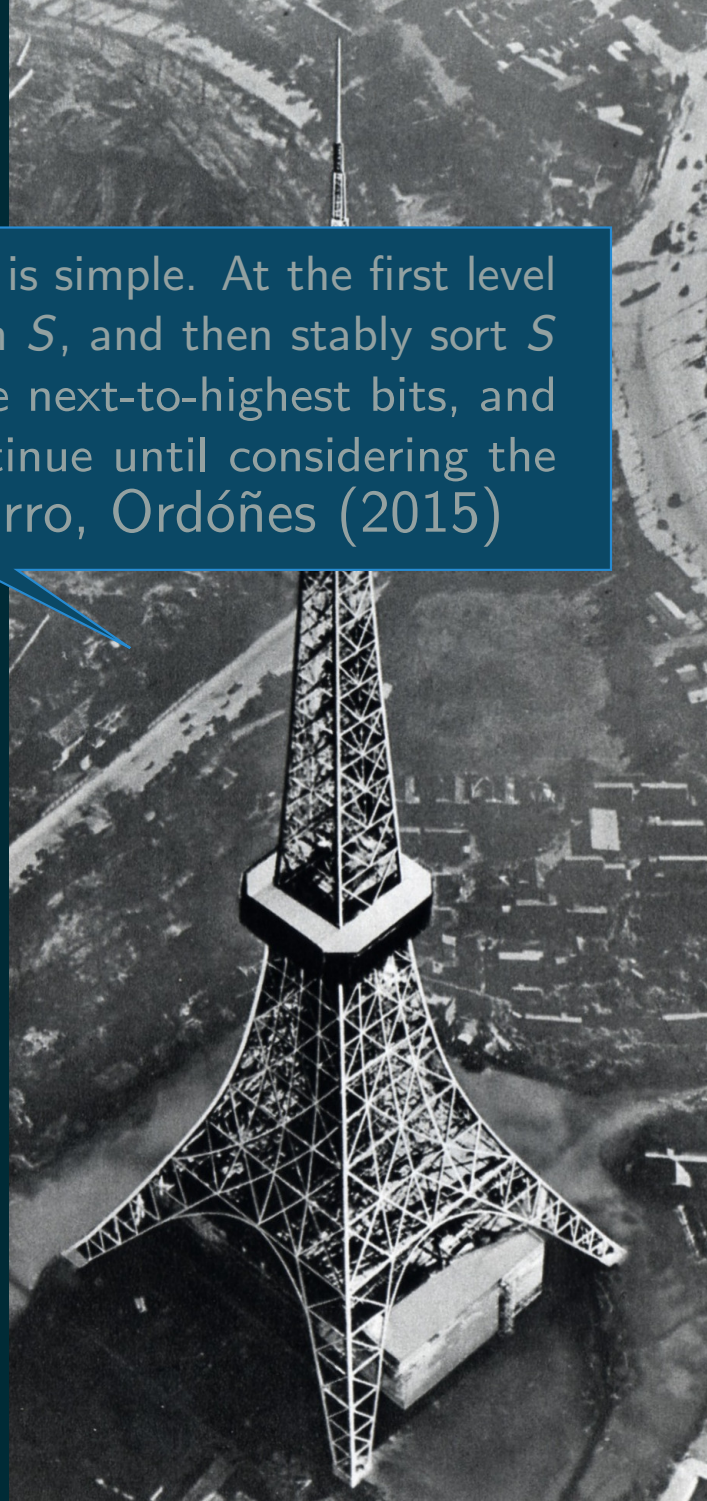
Similar but Different

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)

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

Considered Bit Prefixes

- Bit-reversal permutation

Wavelet Matrix

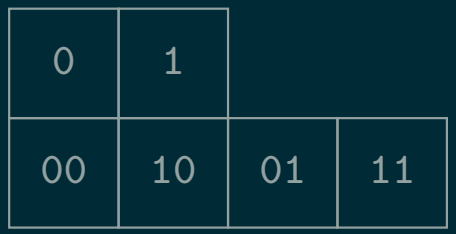
Considered Bit Prefixes

- 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

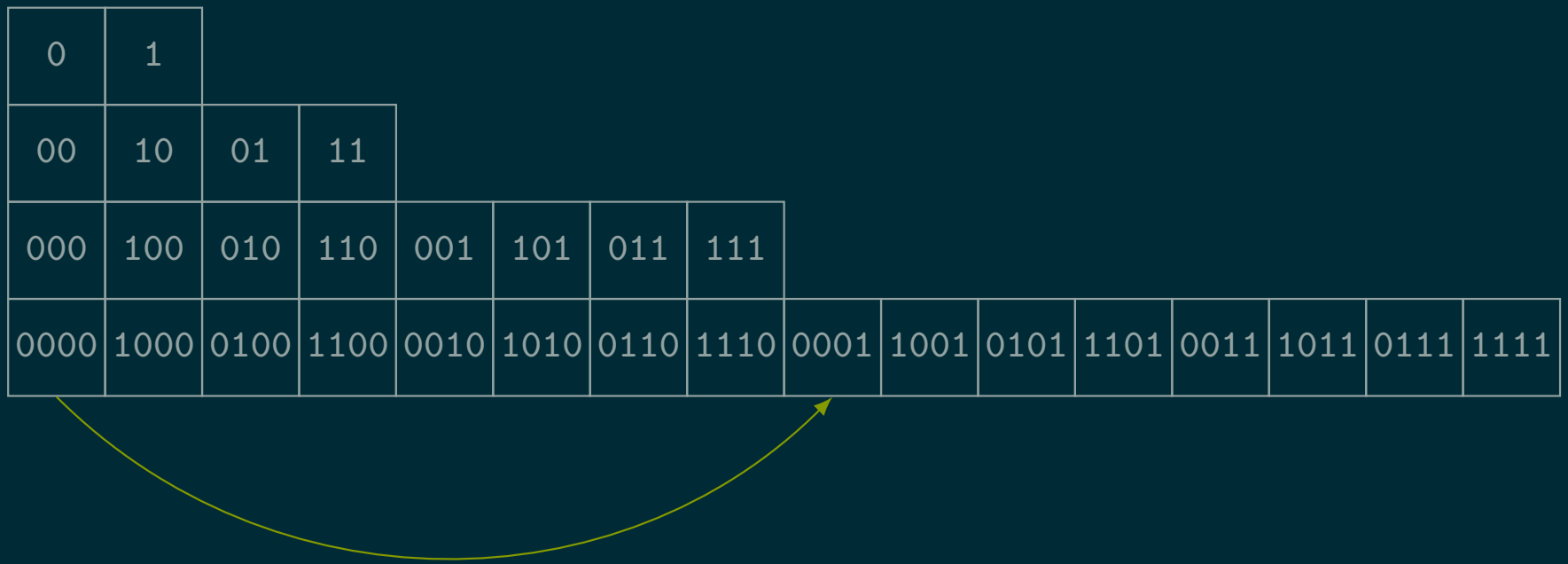
- Bit-reversal permutation
- Compute borders in this order

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

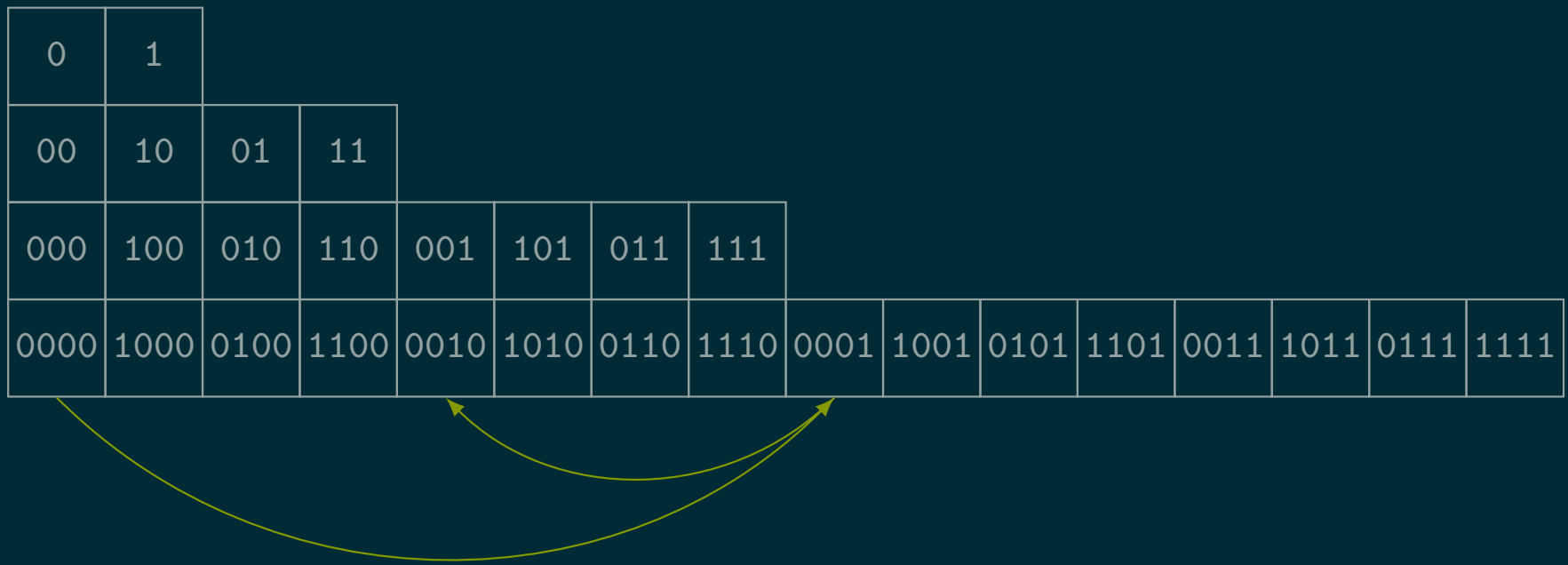
- Bit-reversal permutation
- Compute borders in this order



Wavelet Matrix

Considered Bit Prefixes

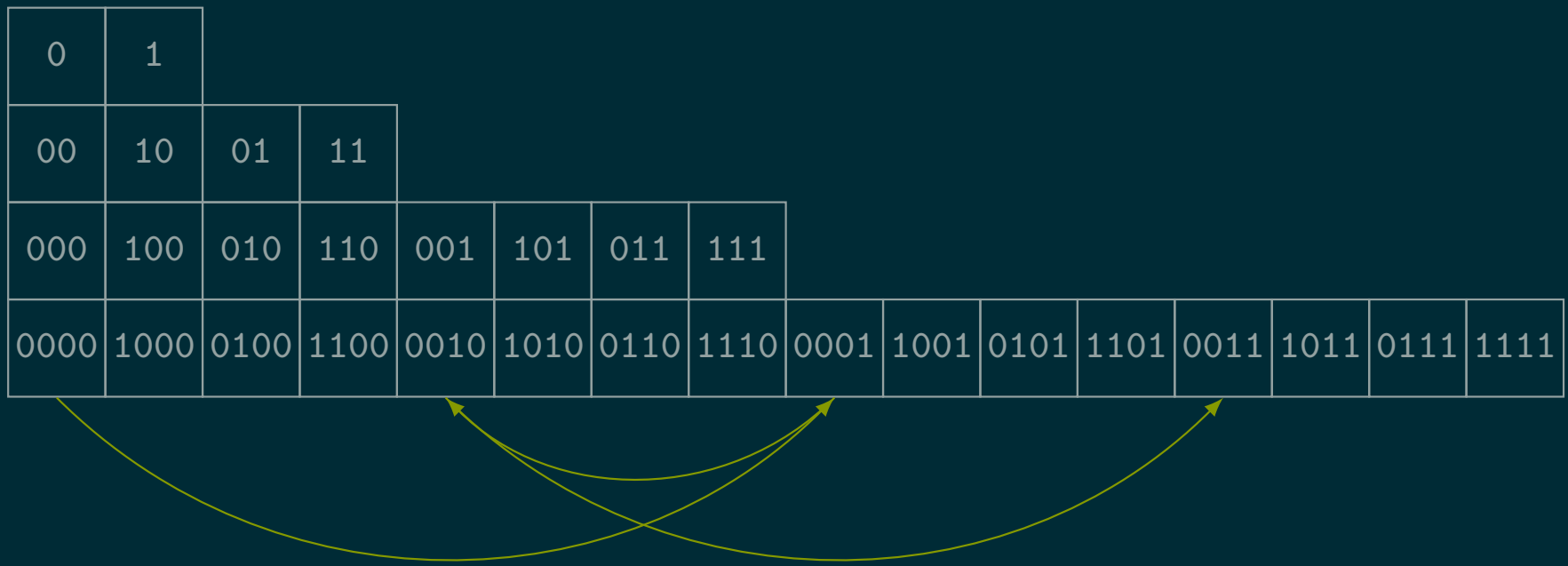
- Bit-reversal permutation
- Compute borders in this order



Wavelet Matrix

Considered Bit Prefixes

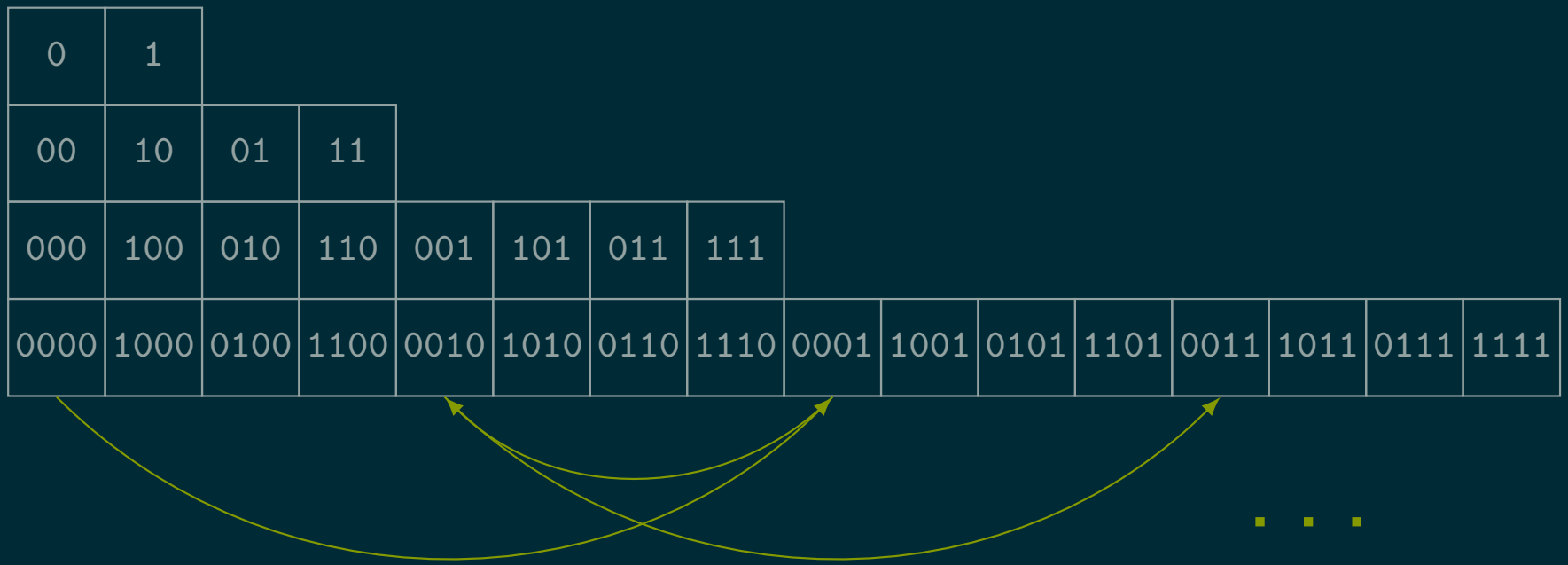
- Bit-reversal permutation
- Compute borders in this order



Wavelet Matrix

Considered Bit Prefixes

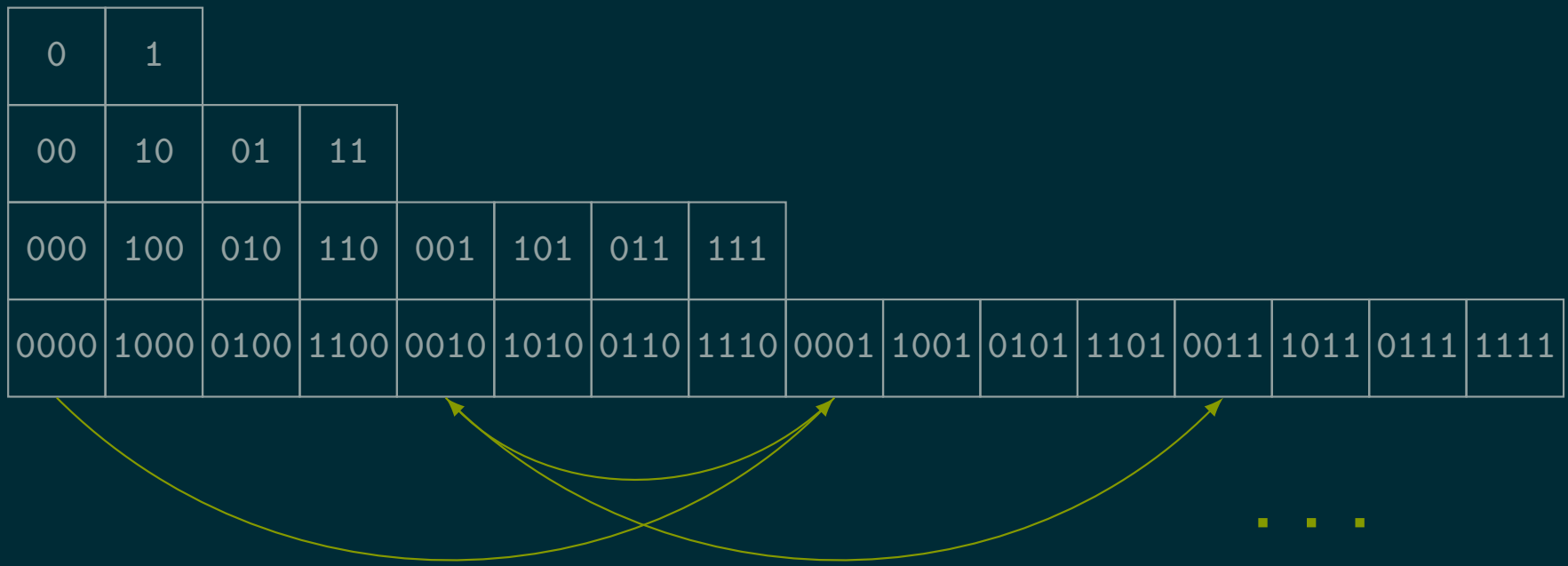
- Bit-reversal permutation
- Compute borders in this order



Wavelet Matrix

Considered Bit Prefixes

- Bit-reversal permutation
- Compute borders in this order



- All algorithms presented can also compute the Wavelet Matrix

Experiments

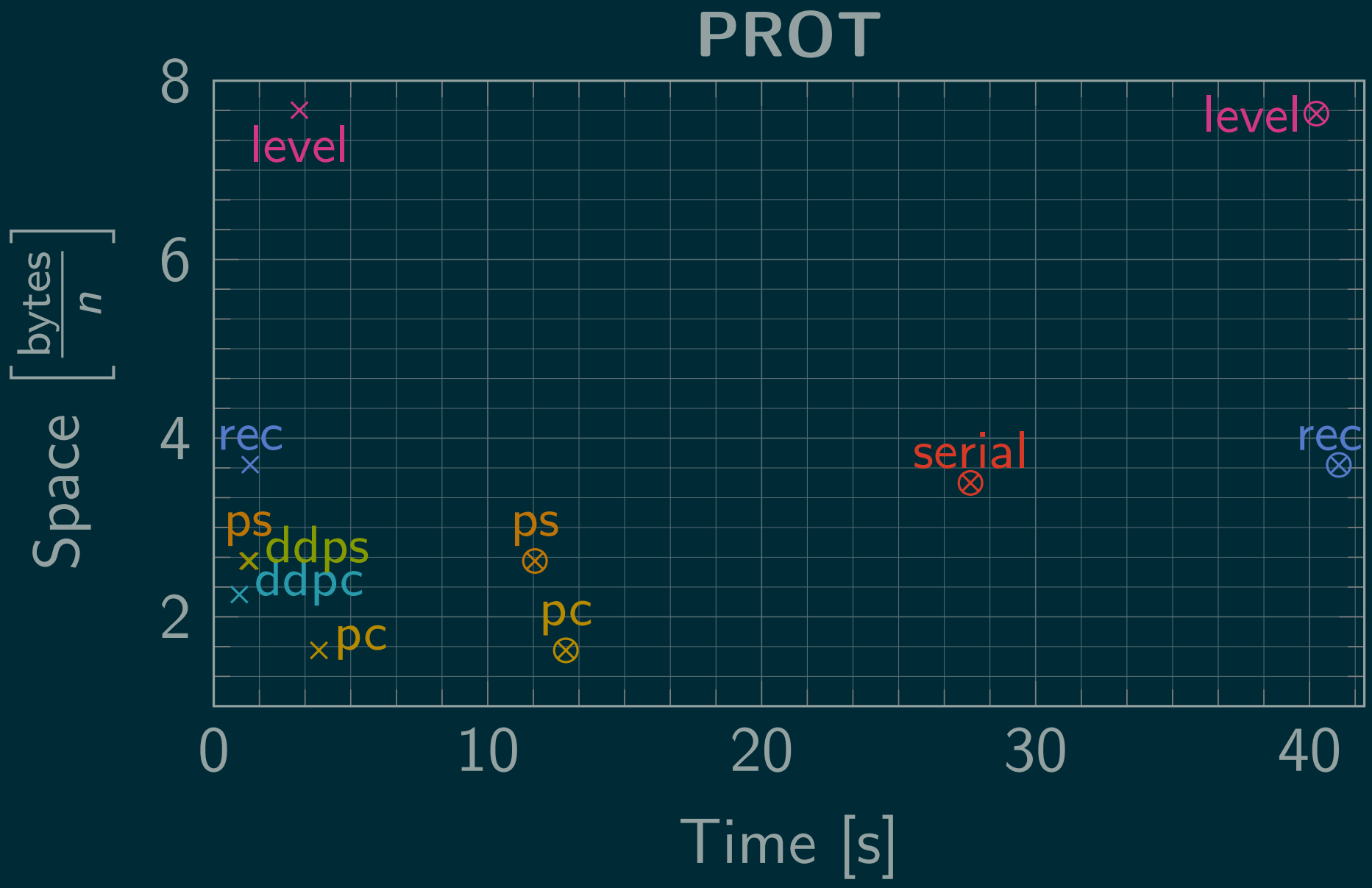
Setting

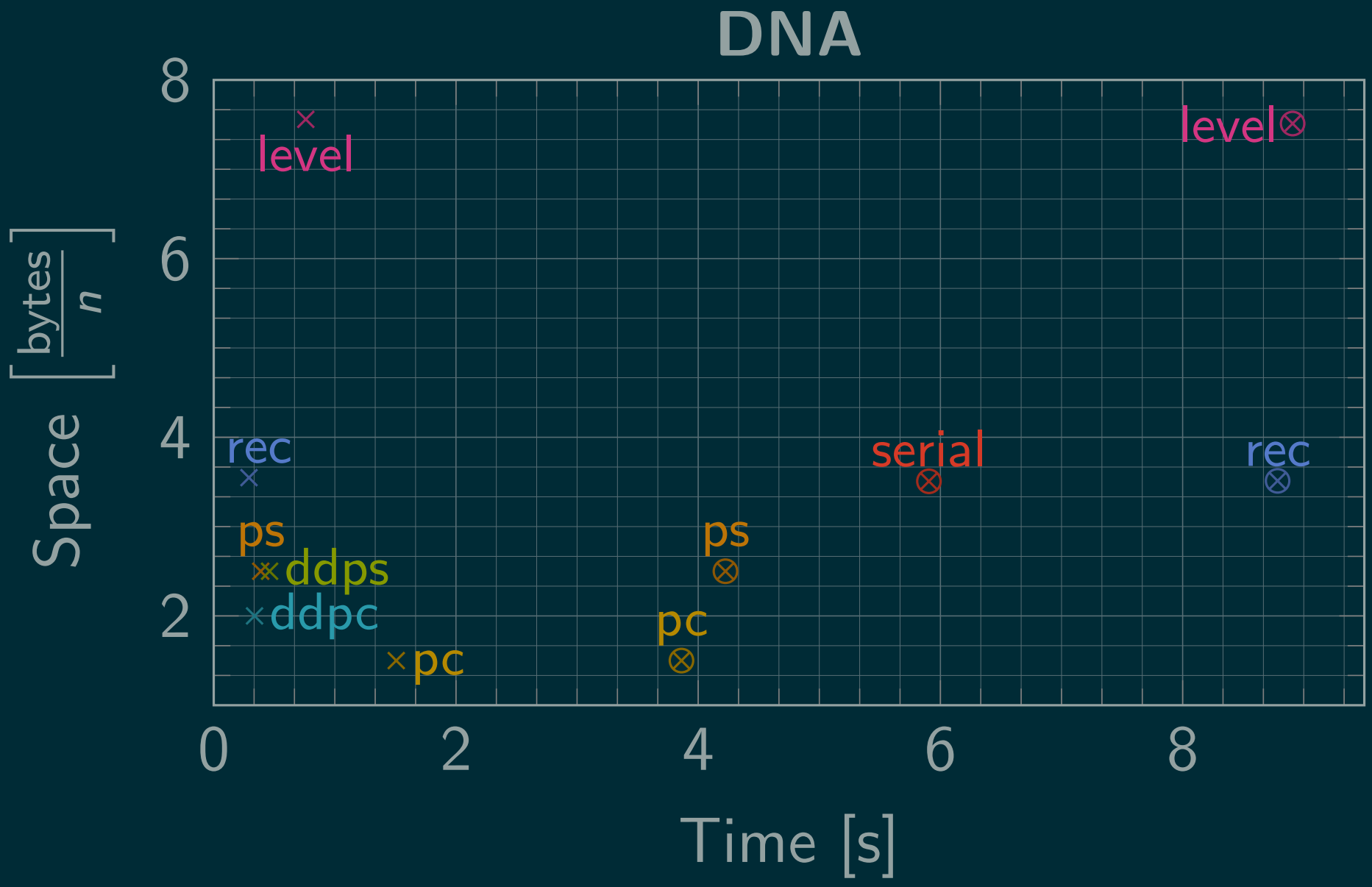
- Workstation with 32 cores and 256 GB RAM
- 11.8 GB Proteins with $\sigma = 27$
- 4 GB DNA with $\sigma = 16$

Experiments

Setting

- Workstation with 32 cores and 256 GB RAM
- 11.8 GB Proteins with $\sigma = 27$
- 4 GB DNA with $\sigma = 16$
- Test it yourself: `www.github.com/kurpicz/pwm`


PROT
level
level
rec
serial
rec
ps
ddps
ddpc
ps
pc
pc
8
6
4
2
0
10
20
30
40
Space [bytes/n]
Time [s]


DNA
level
level
rec
serial
rec
ps
ddps
ddpc
pc
ps
pc
Space $\left[\frac{\text{bytes}}{n}\right]$
Time [s]
0
2
4
6
8
2
4
6
8

Conclusion & Outlook

- New algorithms for the Wavelet Tree
- First practical algorithms for the Wavelet Matrix
- Fast, simple and lightweight

Conclusion & Outlook

- New algorithms for the Wavelet Tree
- First practical algorithms for the Wavelet Matrix
- Fast, simple and lightweight
- Better scaling possible?

Conclusion & Outlook

- New algorithms for the Wavelet Tree
- First practical algorithms for the Wavelet Matrix
- Fast, simple and lightweight
- Better scaling possible?

Thank You!