

# Introduction to Parallel Construction of Wavelet Trees

José Fuentes

May 8, 2017

# Wavelet Tree

---

- A wavelet tree maintains a sequence  $S$  of symbols  $s_1, s_2, \dots, s_n$
- Symbols in  $S$  belongs to an alphabet  $\Sigma = [1 \dots \sigma]$

# Wavelet Tree

## Construction

---

$S = \text{once\_upon\_a\_time\_a\_PhD\_student}$        $\Sigma = \{\text{o,n,c, e,_,u,p,a,t ,i,m,P,h,D,s,d}\}$

# Wavelet Tree

## Construction

---

$S = \text{once\_upon\_a\_time\_a\_PhD\_student}$

$\Sigma = \{\text{o,n,c, e,_,u,p,a,t,i,m,P,h,D,s,d}\}$

```
0000000011111111
0000111100001111
0011001100110011
0101010101010101
```

# Wavelet Tree

## Construction

---

S = once\_upon\_a\_time\_a\_PhD\_student

$\Sigma = \{o,n,c, e,_,u,p,a,t,i,m,P,h,D,s,d\}$

0000000011111111  
0000111100001111  
0011001100110011  
0101010101010101



once\_upon\_a\_time\_a\_PhD\_student  
000000000000111 0000111 01101001

# Wavelet Tree

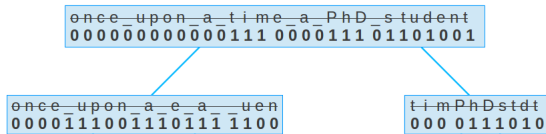
## Construction

---

S = once\_upon\_a\_time\_a\_PhD\_student

$\Sigma = \{o,n,c, e,_,u,p,a,t,i,m,P,h,D,s,d\}$

```
0000000011111 111  
0000111100001 111  
0011001100110 011  
0101010101010 101
```



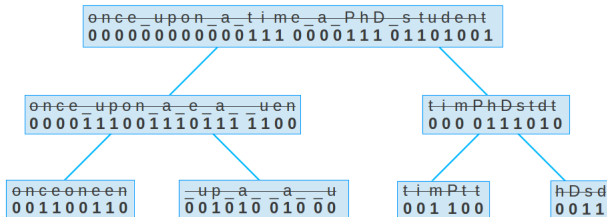
# Wavelet Tree

## Construction

$S = \text{once\_upon\_a\_time\_a\_PhD\_student}$

$\Sigma = \{o,n,c, e,_,u,p,a,t,i,m,P,h,D,s,d\}$

```
000000001111111111
0000111100001111
0011001100110011 ←
0101010101010101
```



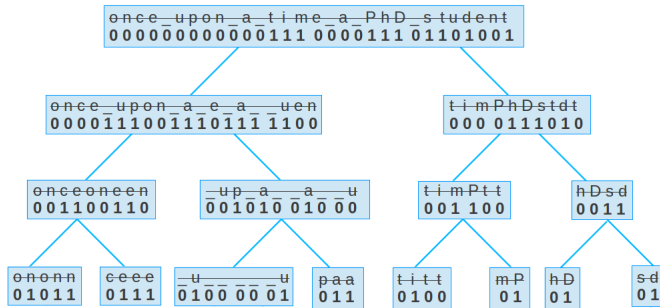
# Wavelet Tree

## Construction

S = once\_upon\_a\_time\_a\_PhD\_student

$\Sigma = \{o,n,c, e,_,u,p,a,t,i,m,P,h,D,s,d\}$

```
000000001111111111
0000111100001111
0011001100110011
0101010101010101
```





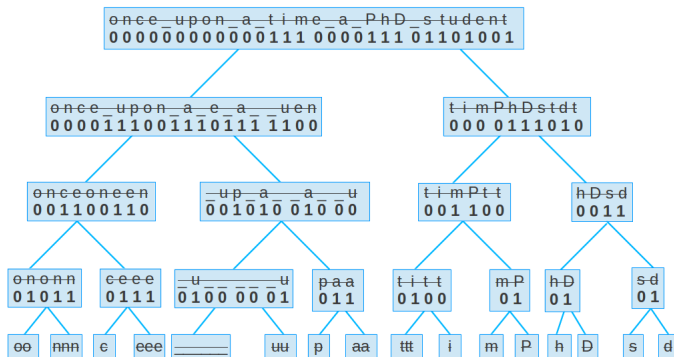
# Wavelet Tree

## Construction

S = once\_upon\_a\_time\_a\_PhD\_student

$\Sigma = \{o,n,c, e,_,u,p,a,t,i,m,P,h,D,s,d\}$

```
000000001111111111
0000111100001111
0011001100110011
0101010101010101
```



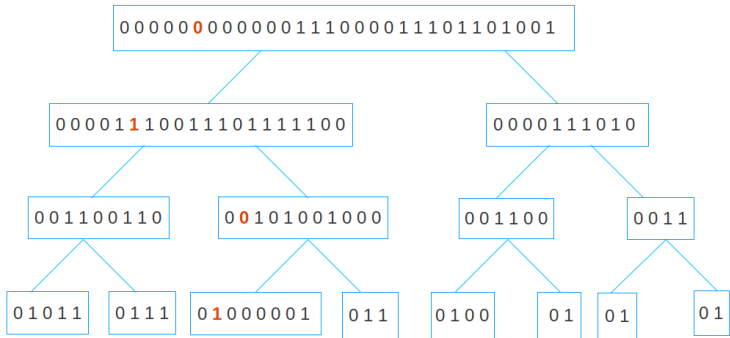
# WT querying operations

$$\text{access}(S, 5) = p$$

---

$S = \text{once\_upon\_a\_time\_a\_PhD\_student}$

$\Sigma = \{o,n,c, e,_,u,p,a,t,i,m,P,h,D,s,d\}$

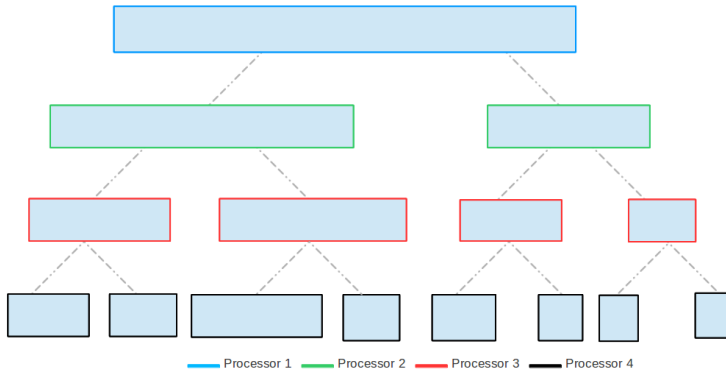


# Parallel Per-level Iterative Construction

S = once\_upon\_a\_time\_a\_PhD\_student

$\Sigma = \{o,n,c,e,_,u,p,a,t,i,m,P,h,D,s,d\}$

000000001111111111	Processor 1
000011111000011111	Processor 2
001100111001100111	Processor 3
010101010101010101	Processor 4

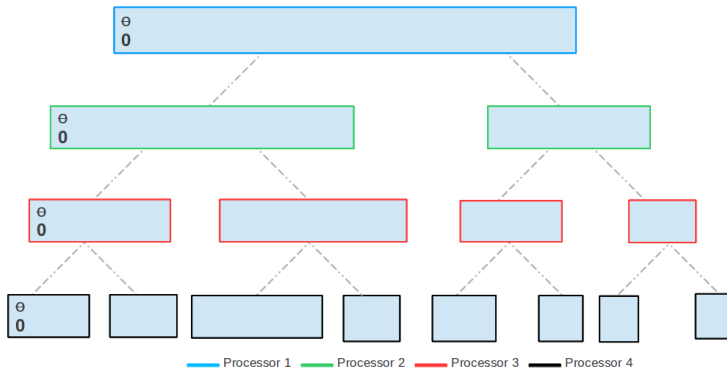


# Parallel Per-level Iterative Construction

S = Once\_upon\_a\_time\_a\_PhD\_student

$\Sigma = \{o,n,c,e,_,u,p,a,t,i,m,P,h,D,s,d\}$

0 0000000111111 111	Processor 1
0 0001111100001 111	Processor 2
0 011001100110 011	Processor 3
0 101010101010 101	Processor 4

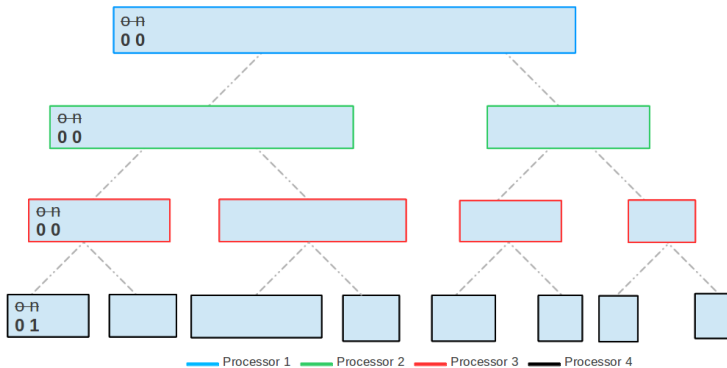


# Parallel Per-level Iterative Construction

S = once upon a time a PhD student

$\Sigma = \{o,n,c,e,_,u,p,a,t,i,m,P,h,D,s,d\}$

0 0 0 0 0 0 0 1 1 1 1 1 1 1	Processor 1
0 0 0 0 1 1 1 1 1 0 0 0 0 1	Processor 2
0 0 1 1 0 0 1 1 0 0 1 1 0 1	Processor 3
0 1 0 1 0 1 0 1 0 1 0 1 0 1	Processor 4

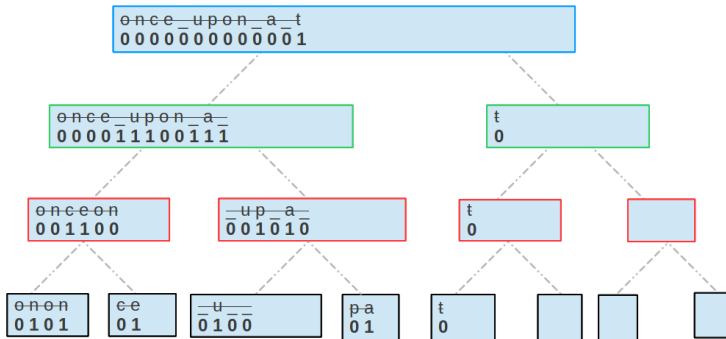


# Parallel Per-level Iterative Construction

S = once\_upon\_a\_t time\_a\_PhD\_student

$\Sigma = \{o,n,c,e,_,u,p,a,t,i,m,P,h,D,s,d\}$

000000000111111111	Processor 1
00001111100001111	Processor 2
0011001100110011	Processor 3
0101010101010101	Processor 4



Processor 1 Processor 2 Processor 3 Processor 4

## Parallel Per-level Iterative Construction

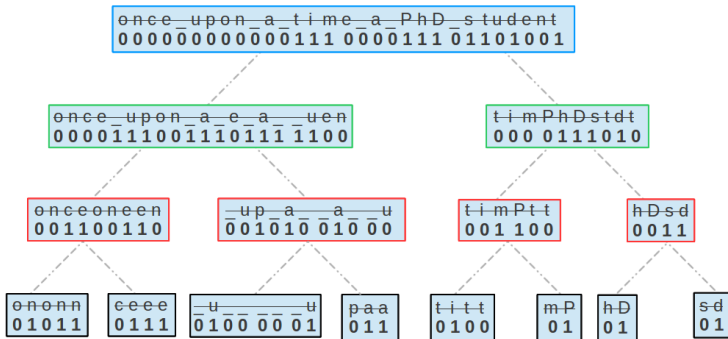
S = once\_upon\_a\_time\_a\_PhD\_student

$$\Sigma = \{o,n,c, e, \_,u,p,a,t, i,m,P,h,D,s,d\}$$

0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 ← Processor 1

0 0 0 0 1 1 1 1 0 0 0 0 1 1 1  Processor 2

0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 ◀ Processor 3

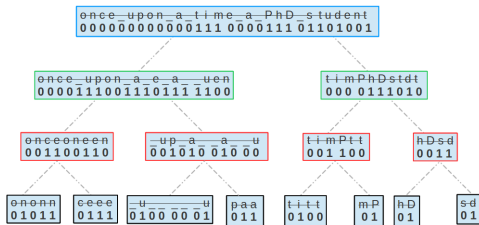
[illegible]

— Processor 1 — Processor 2 — Processor 3 — Processor 4

# Parallel Per-level Iterative Construction

$$T_1 = O(n \lg \sigma)$$

- $T_\infty = \Theta(n)$
- Parallelism =  $\frac{T_1}{T_\infty} = \Theta(\lg \sigma)$



We need only  $P = \lg \sigma$  to obtain the optimal speedup



# Domain Decomposition Construction

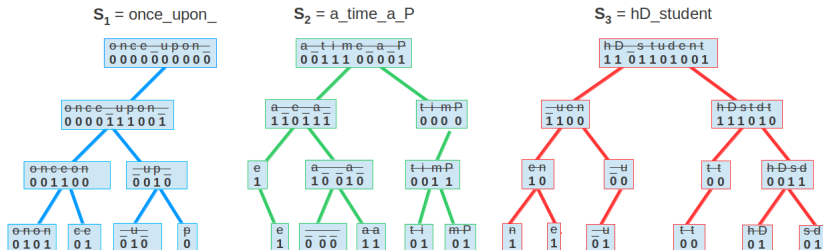
---

$S_1 = \text{once\_upon\_}$

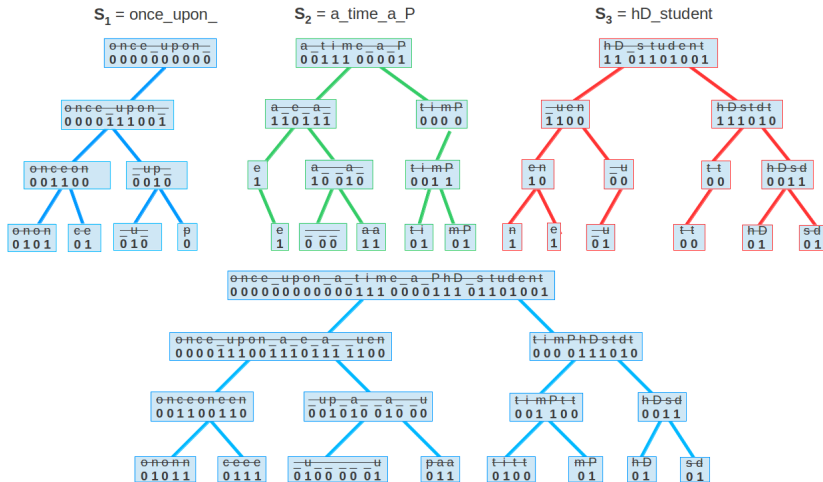
$S_2 = \text{a\_time\_a\_P}$

$S_3 = \text{hD\_student}$

# Domain Decomposition Construction



# Domain Decomposition Construction



# Domain Decomposition Construction

$$T_1 = O(n \lg \sigma)$$

- $T_\infty = O(\lg n)$ 
  - for  $O(p/\lg \sigma)$  segments
  - $O(n \lg \sigma/p)$  time for partial wavelet tree construction
  - $O(\sigma/\lg \sigma + \lg p)$  time for prefix sum
  - $O(n \lg \sigma/pw)$  time for merge, where  $w$  is the word size of that architecture
- $\frac{T_1}{T_\infty} = O(n \lg \sigma / \lg n)$

