Introduction to Parallel Construction of Wavelet Trees

José Fuentes

May 9, 2017

Wavelet Tree

- A wavelet tree maintains a sequence S of symbols $s_1, s_2, \ldots, s_n$
- Symbols in S belongs to an alphabet $\Sigma = [1 \ldots \sigma]$
- Support operations in $O(\lg \sigma)$ time:
  - $access(S, i)$ returns the symbol s_i
  - $rank_c(S, i)$ counts the times symbol c appears up to position i
  - $select_c(S, i)$ returns the position in S of the i-th appearance of symbol c

Wavelet Tree

Construction

S = once_upon_a_time_a_PhD_student **Σ** = {o,n,c, e,_,u,p,a,t ,i,m,P,h,D,s,d}

Wavelet Tree

Construction

S = once_upon_a_time_a_PhD_student

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

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

Wavelet Tree

Construction

S = once_upon_a_time_a_PhD_student

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

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

o n c e _ u p o n _ a _ t i m e _ a _ P h D _ s t u d e n t
0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 1 1 1 0 1 1 0 1 0 0 1

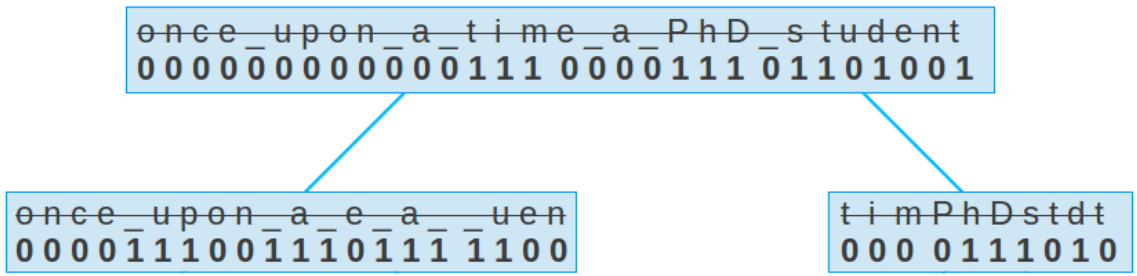
Wavelet Tree

Construction

S = once_upon_a_time_a_PhD_student

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

```
0 0 0 0 0 0 0 0 1 1 1 1 1  1 1 1
0 0 0 0 1 1 1 1 0 0 0 0 1  1 1 1
0 0 1 1 0 0 1 1 0 0 1 1 0  0 1 1
0 1 0 1 0 1 0 1 0 1 0 1 0  1 0 1
```



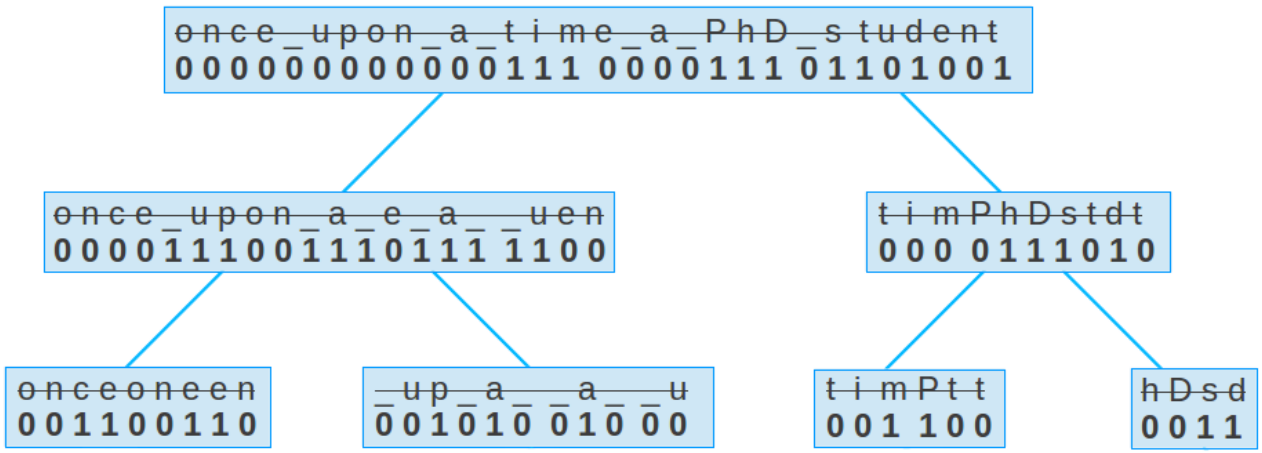
Wavelet Tree

Construction

S = once_upon_a_time_a_PhD_student

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

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



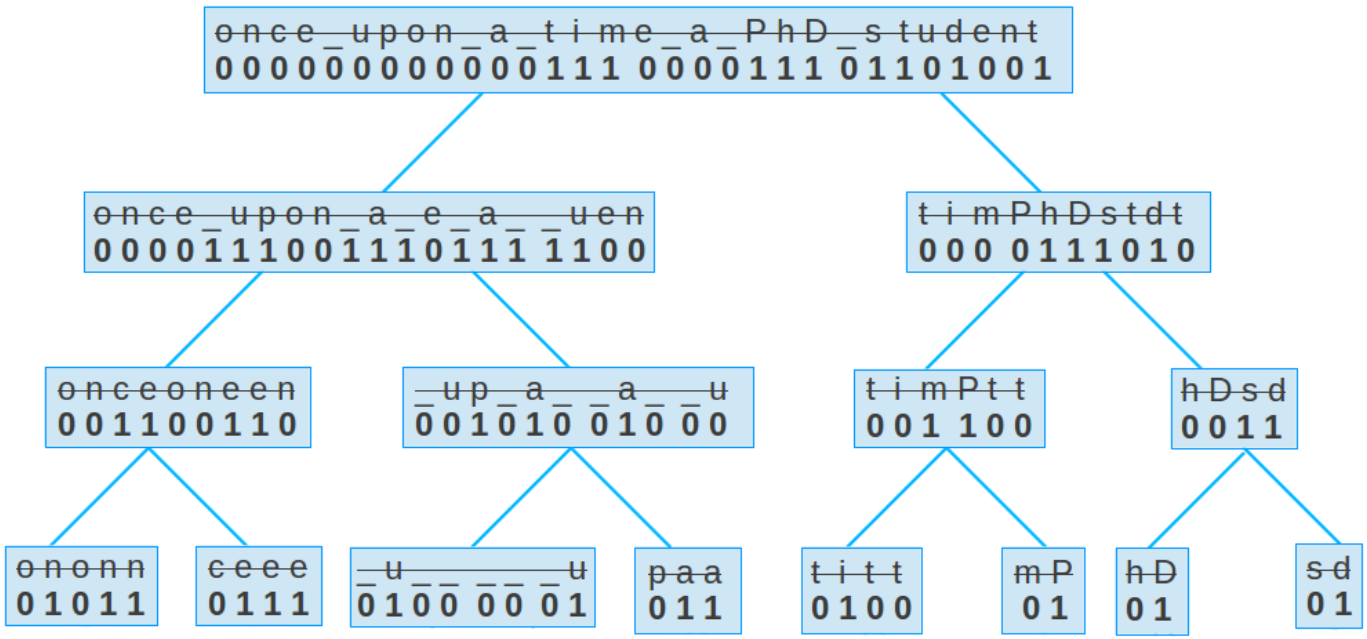
Wavelet Tree

Construction

S = once_upon_a_time_a_PhD_student

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

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



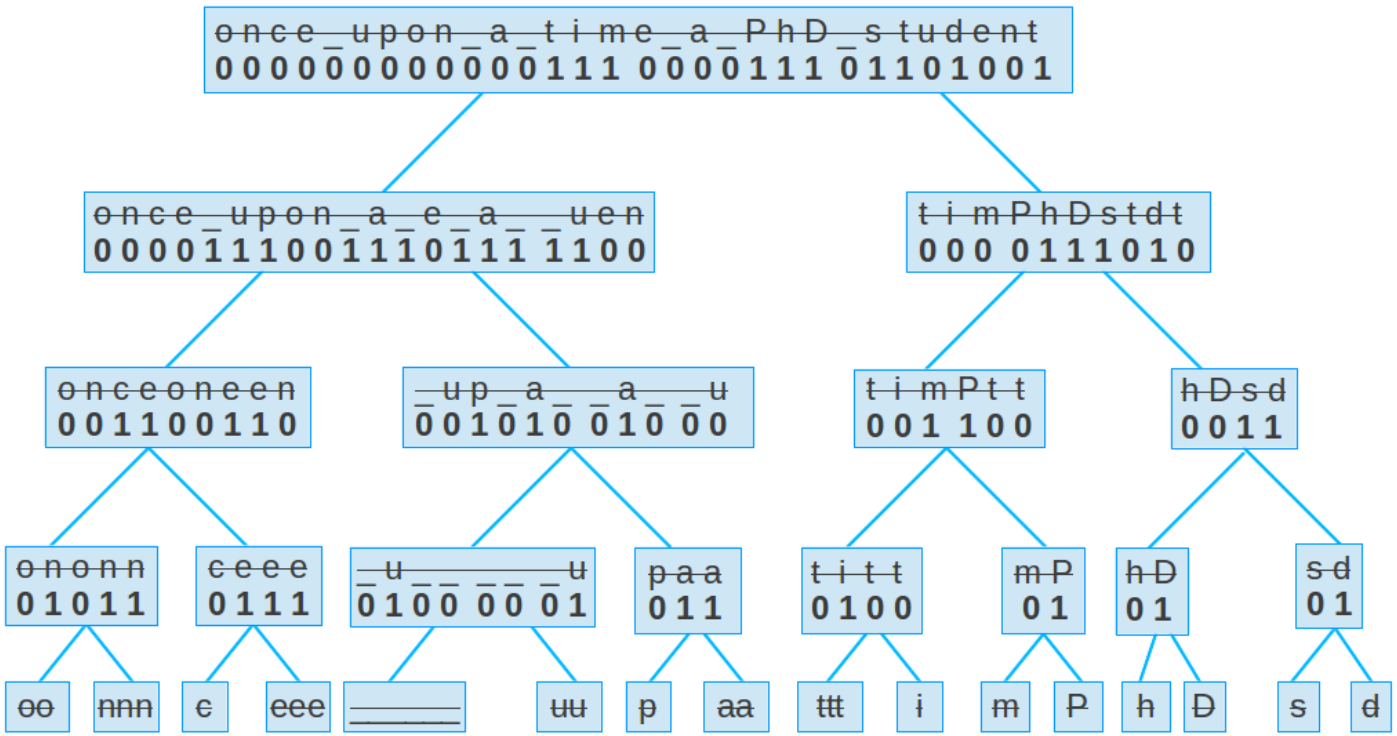
Wavelet Tree

Construction

S = once_upon_a_time_a_PhD_student

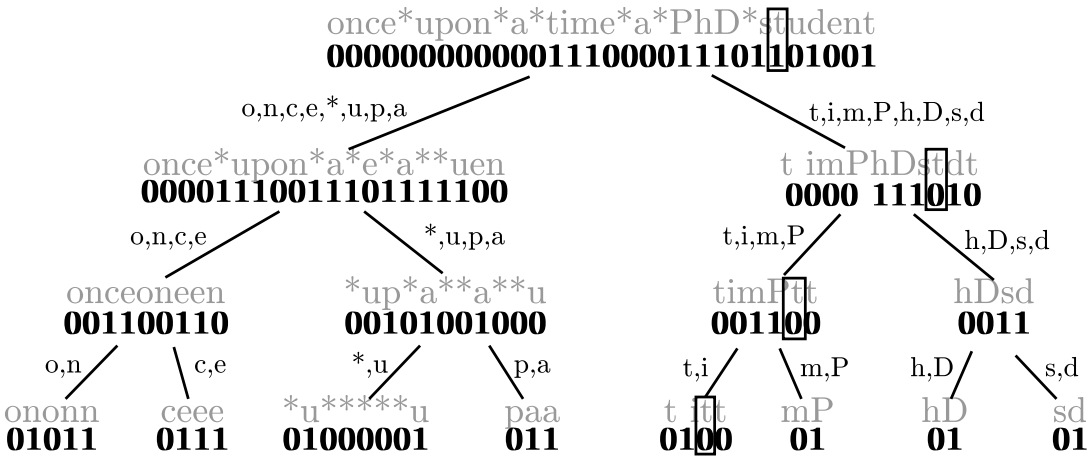
Σ = {o,n,c, e,_,u,p,a,t ,i,m,P,h,D,s,d}

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

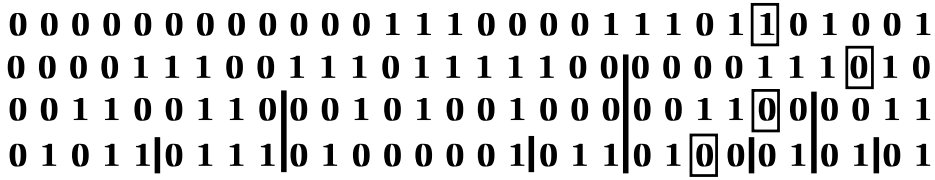


Wavelet Tree

Construction



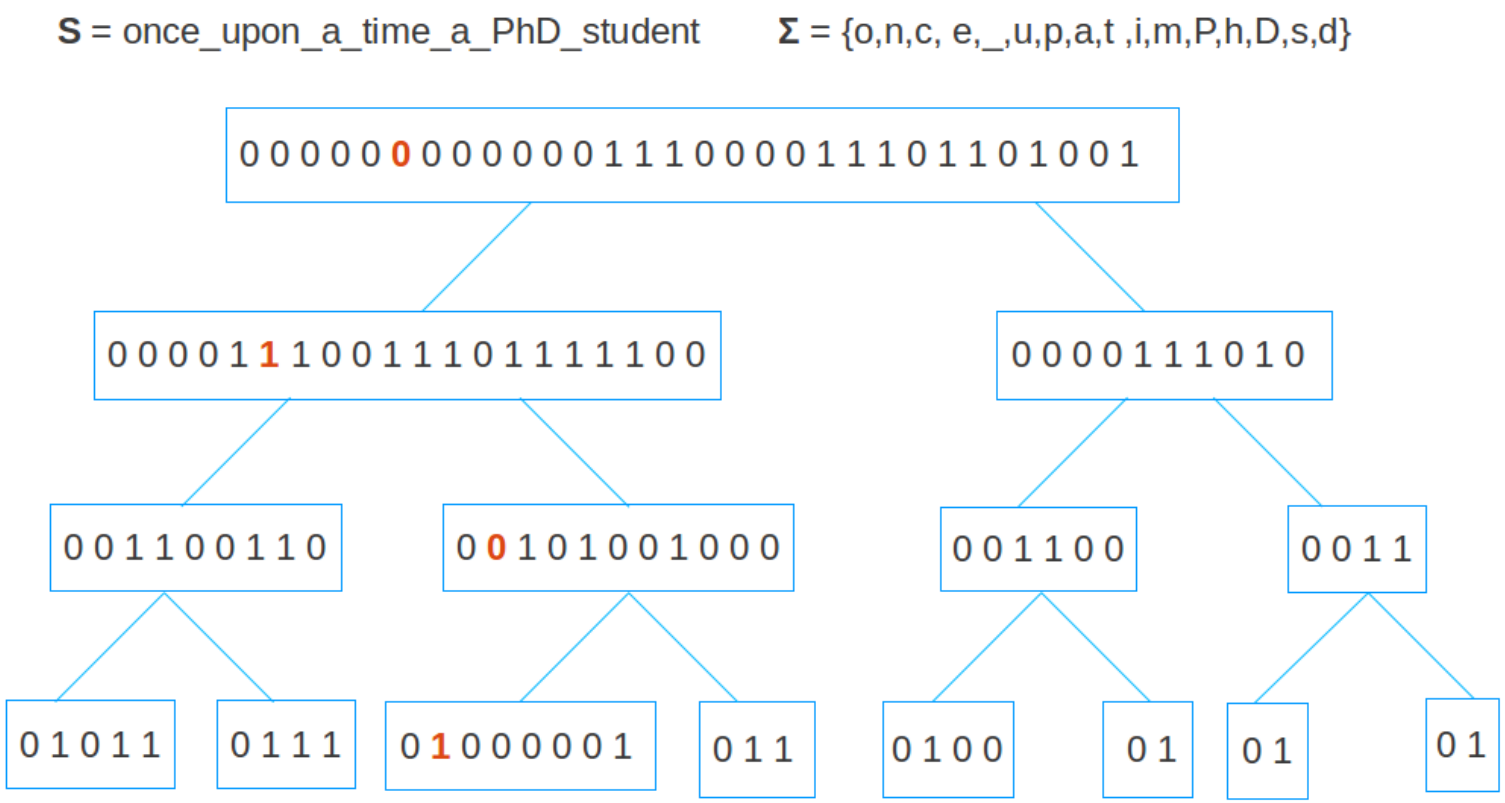
One pointer per node



One pointer per level

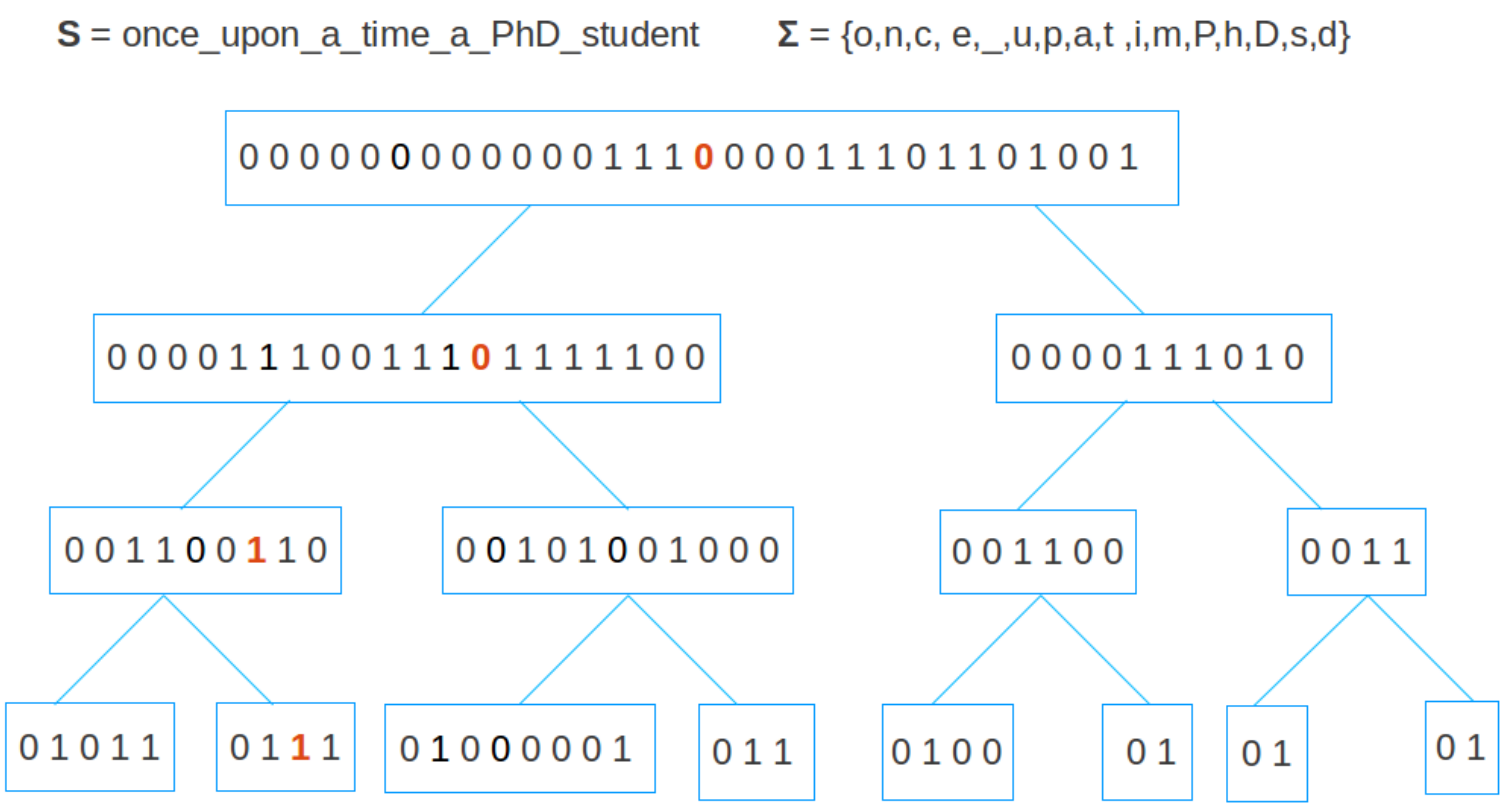
WT querying operations

$access(S, 5) = u$



WT querying operations

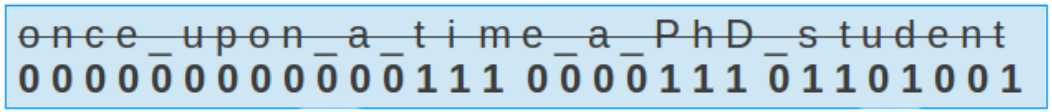
$rank_e(S, 15) = 2$



Parallel Construction

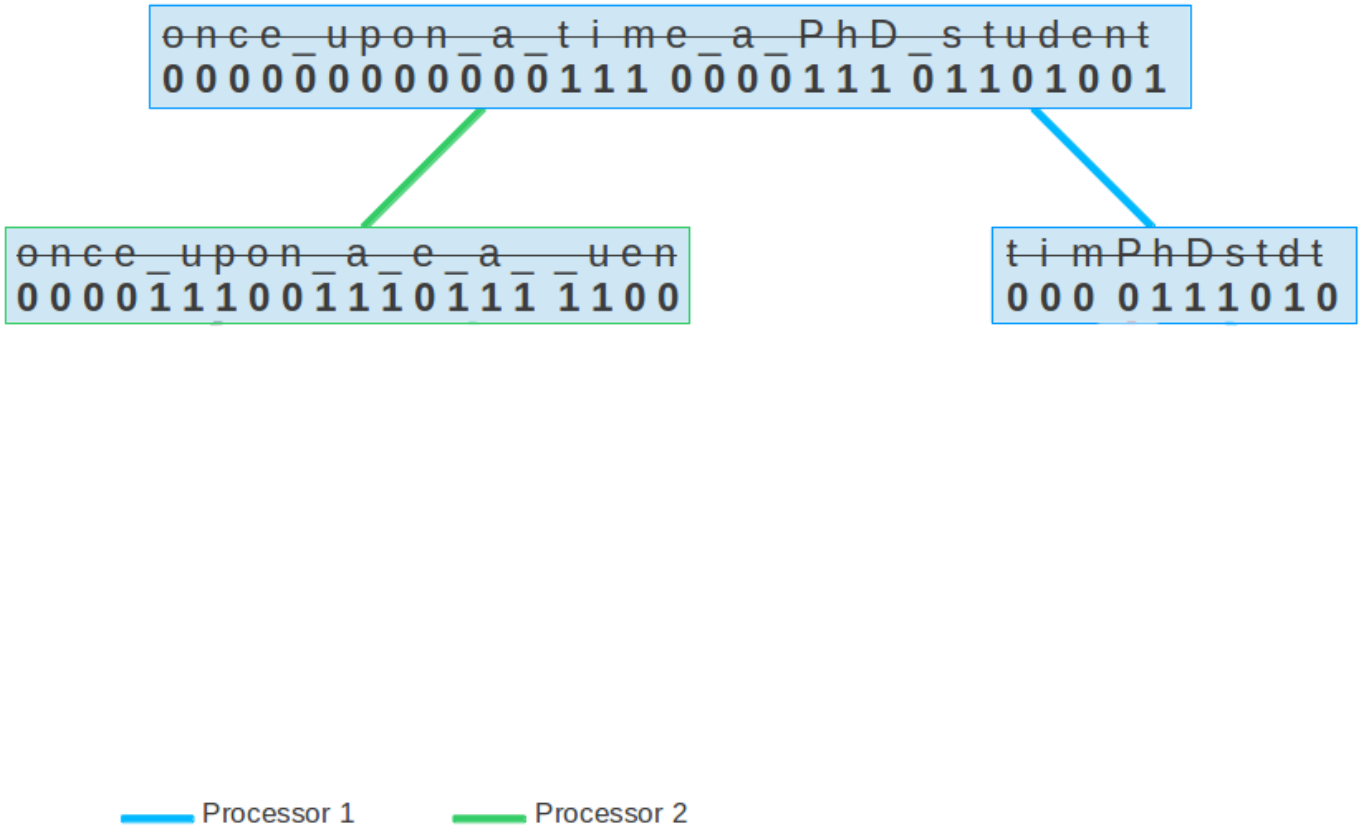
- Recursive construction
- Per-level iterative construction
- Domain decomposition construction

Parallel Recursive Construction

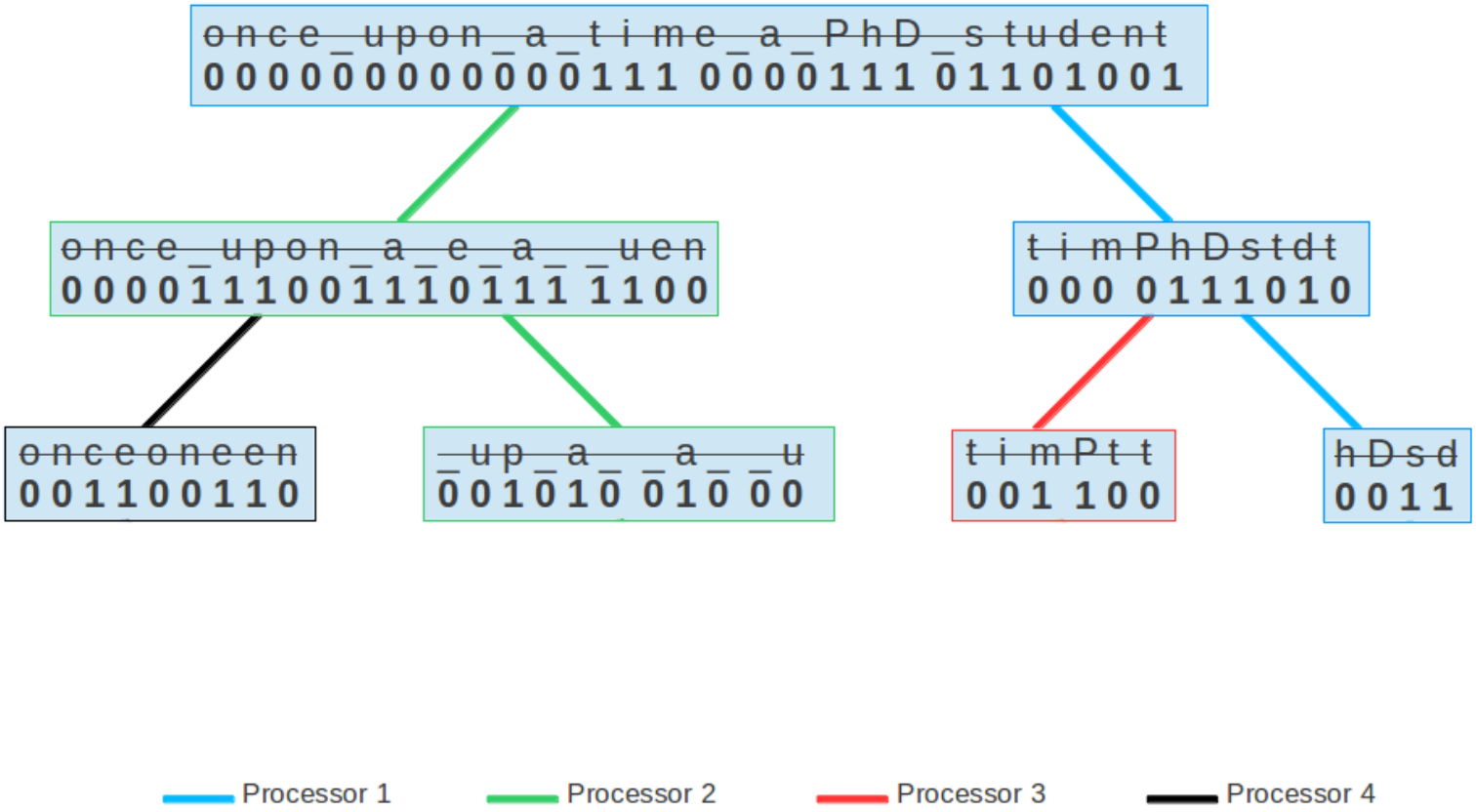


Processor 1

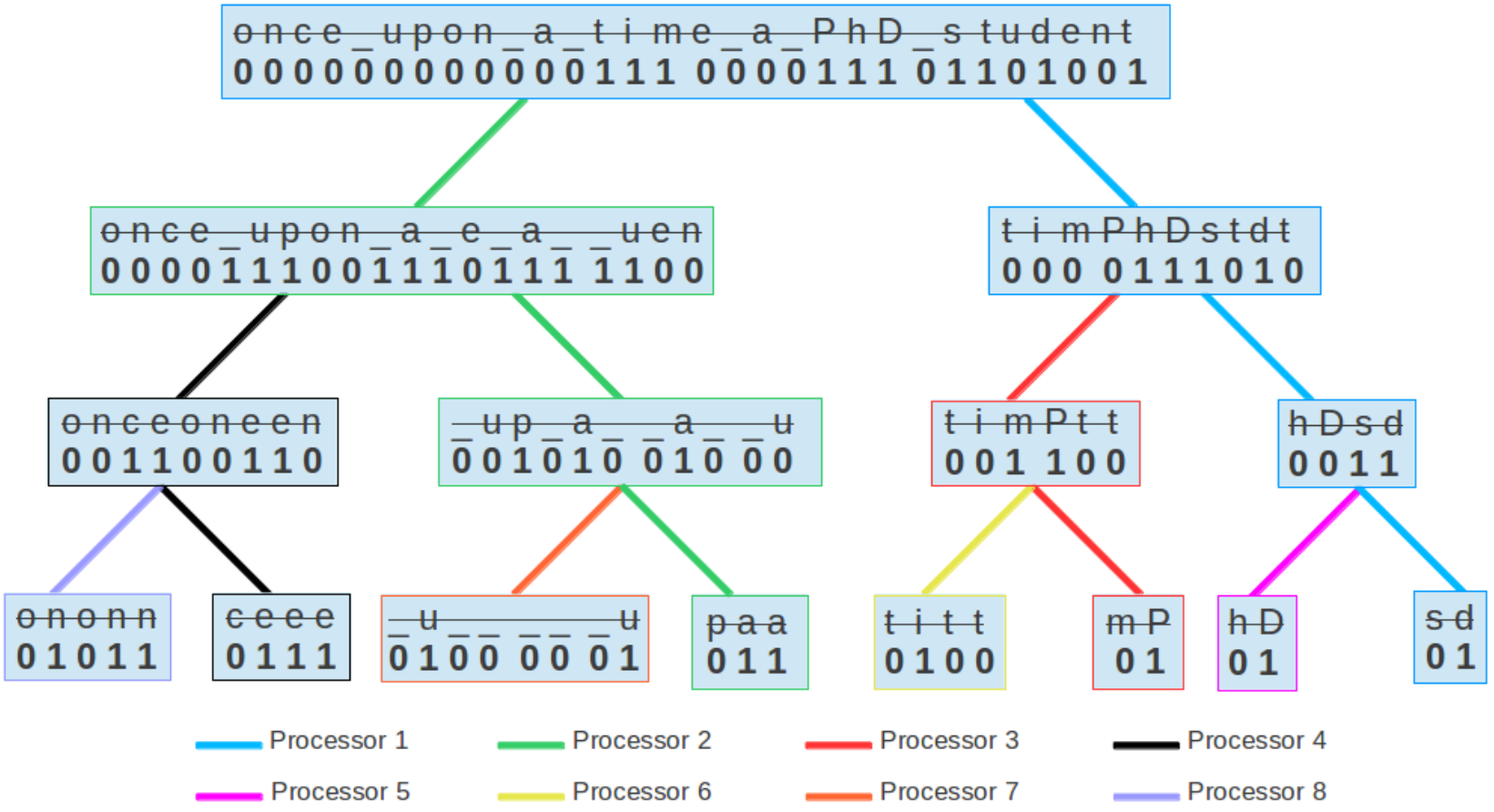
Parallel Recursive Construction



Parallel Recursive Construction



Parallel Recursive Construction



Parallel Recursive Construction

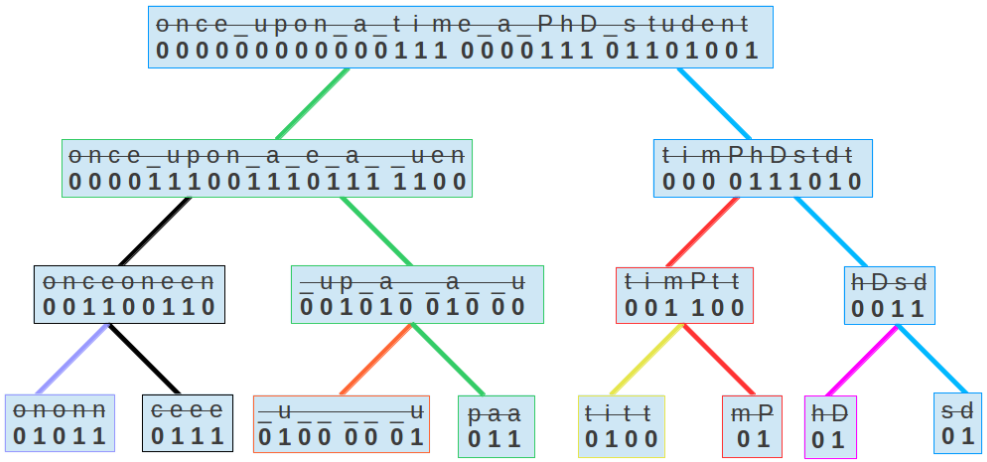
$T_1 = O(n \lg \sigma)$

Best Case

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

Worst Case

- $T_\infty = O(n \lg \sigma)$
- Parallelism $= \frac{T_1}{T_\infty} = O(1)$

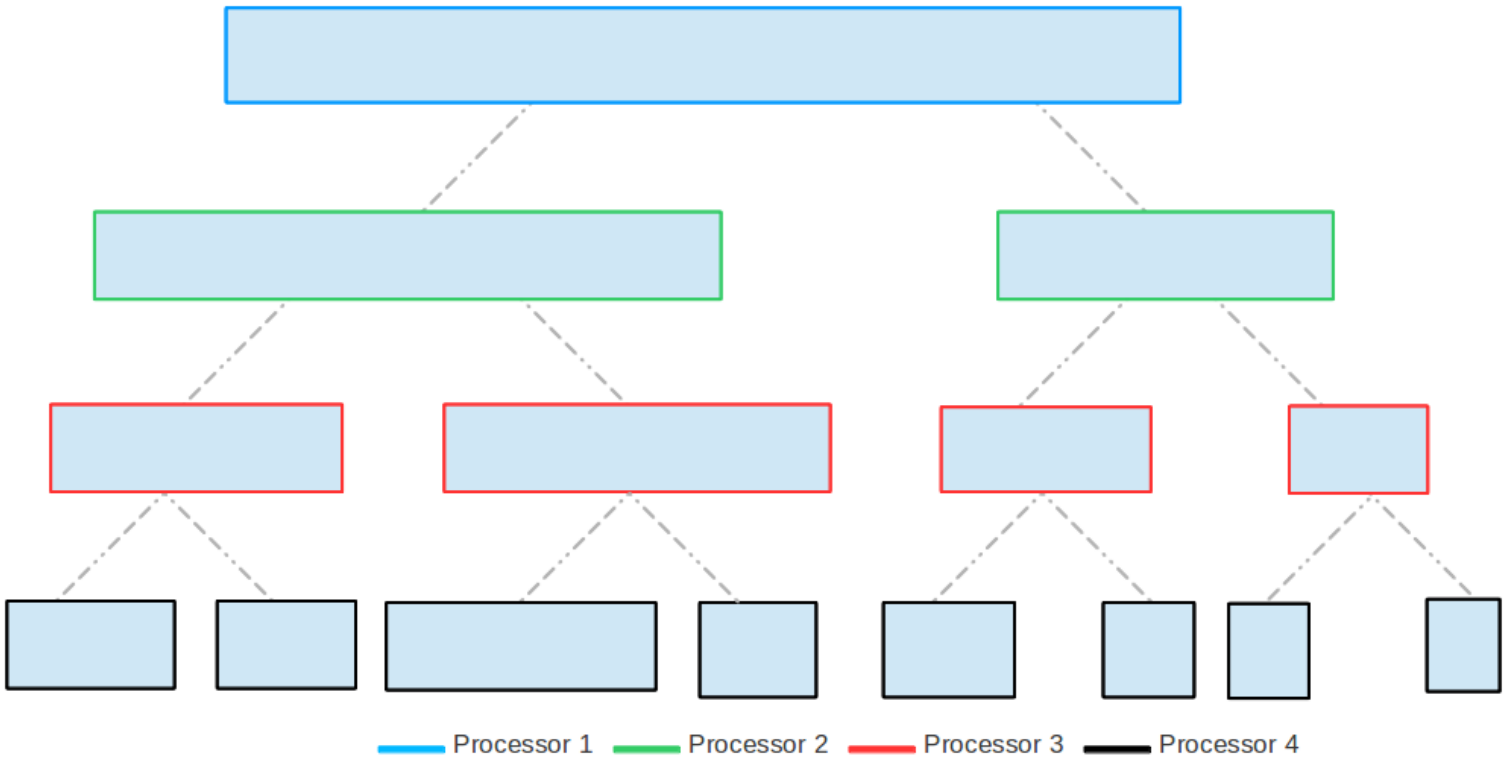


Parallel Per-level Iterative Construction

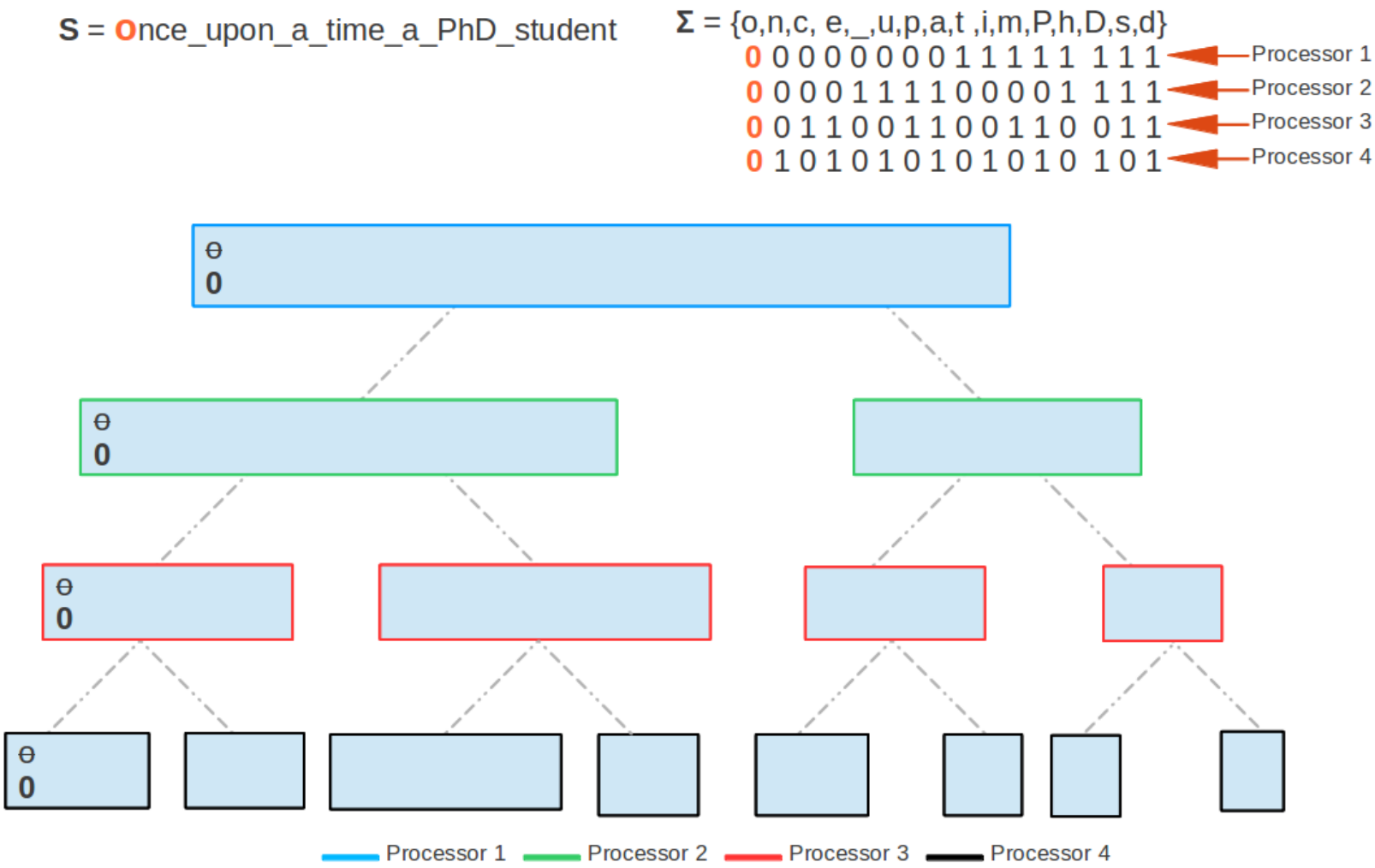
S = once_upon_a_time_a_PhD_student

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

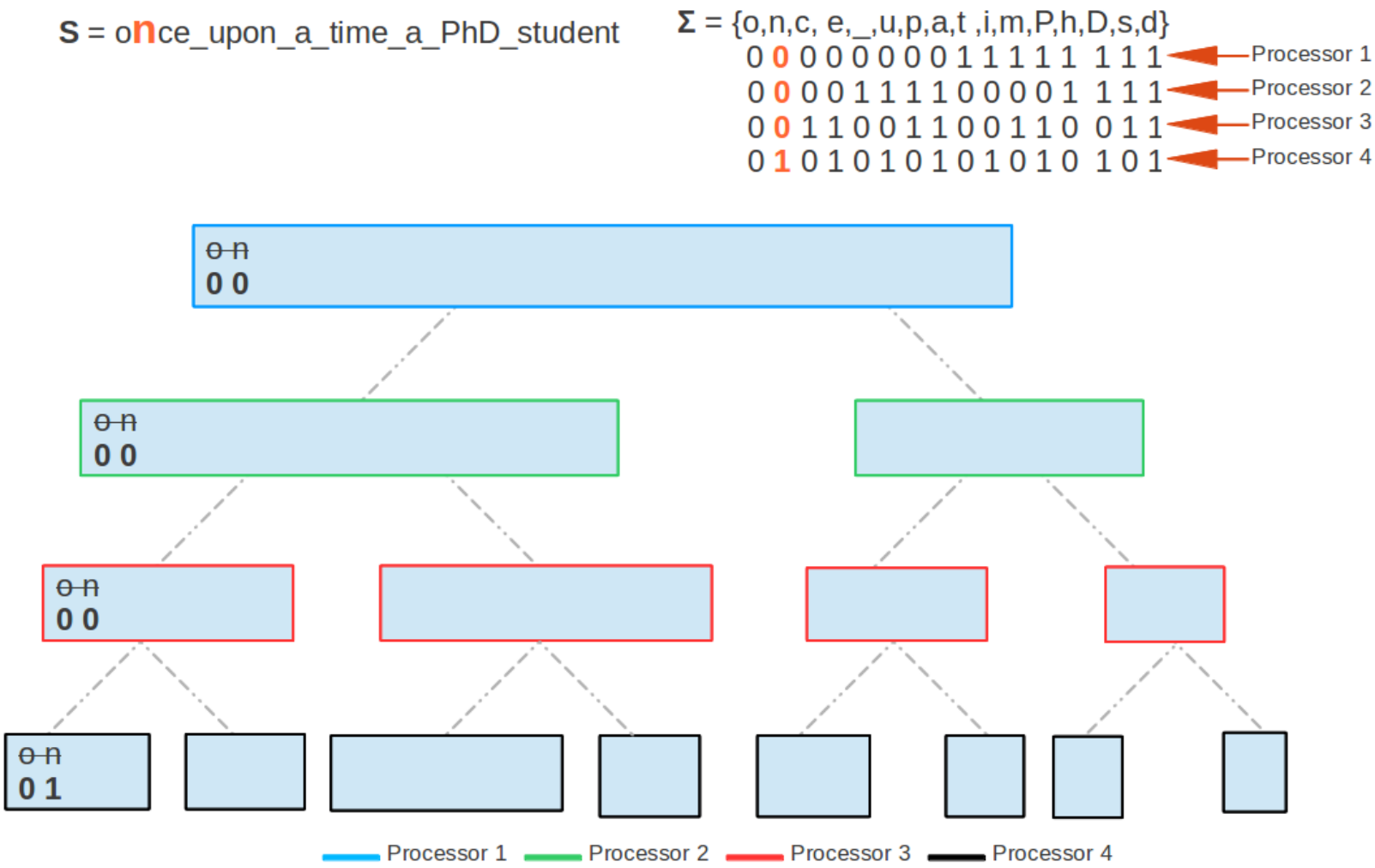
0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 ← Processor 1
0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 ← Processor 2
0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 ← Processor 3
0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 ← Processor 4



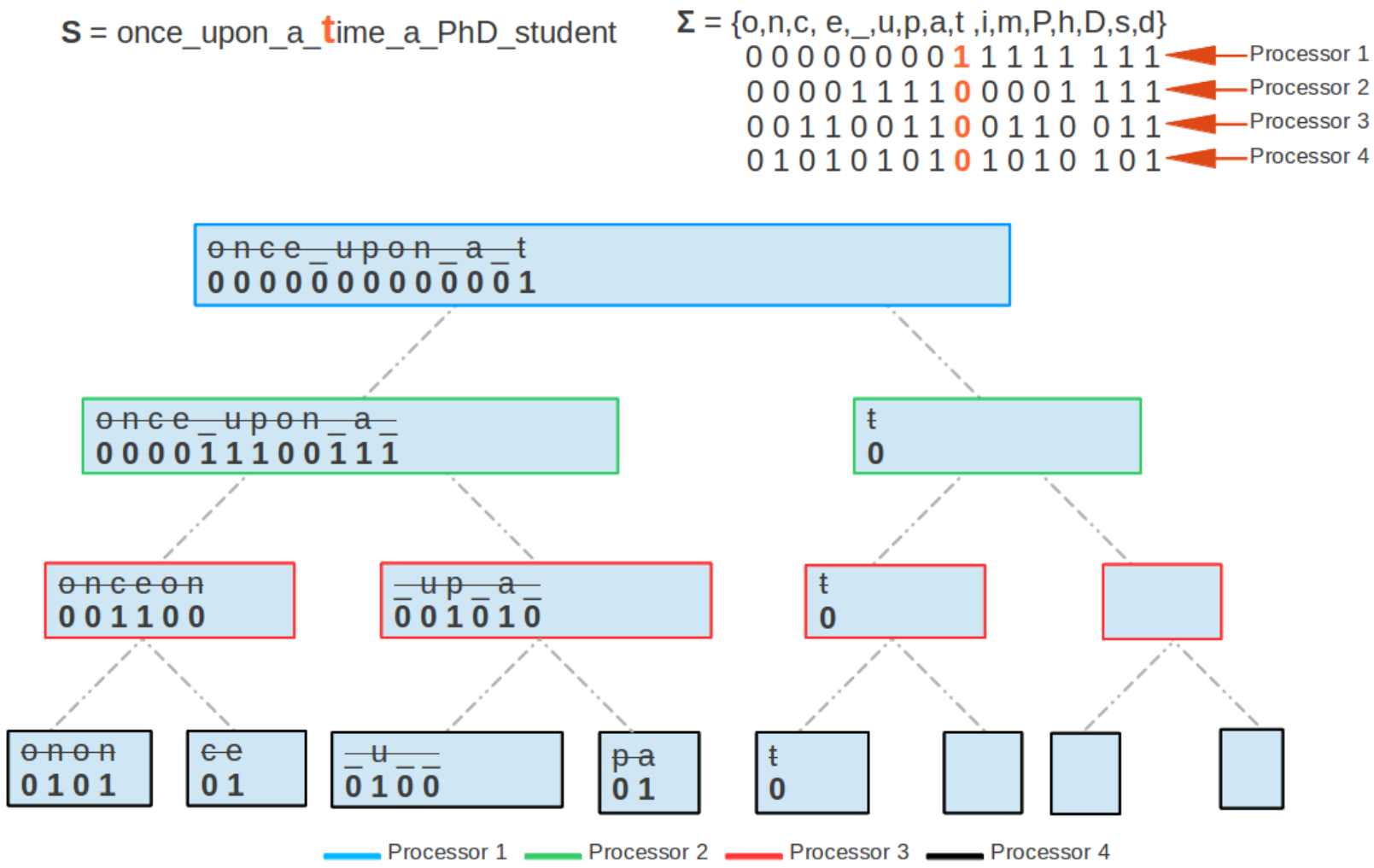
Parallel Per-level Iterative Construction



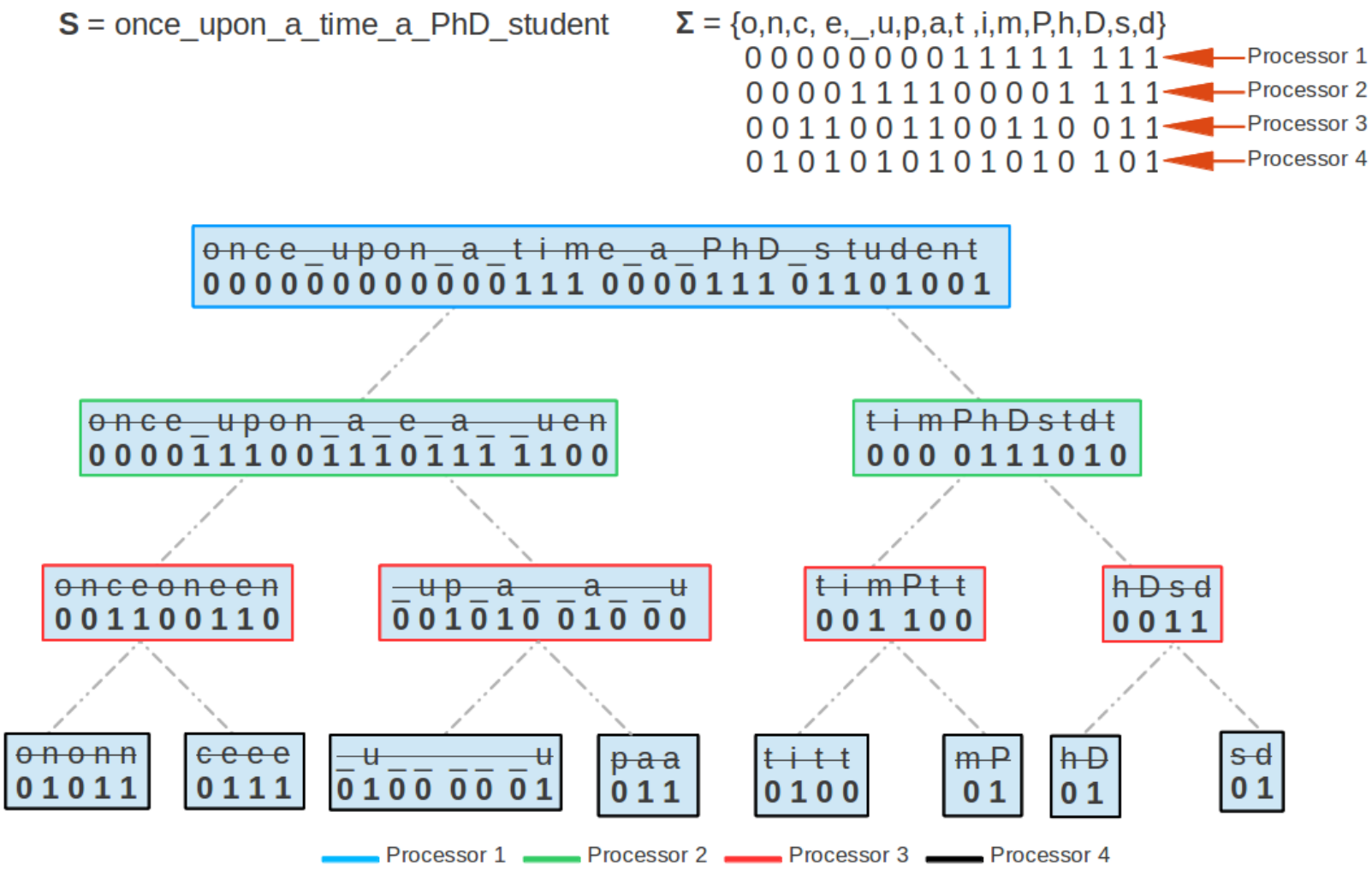
Parallel Per-level Iterative Construction



Parallel Per-level Iterative Construction

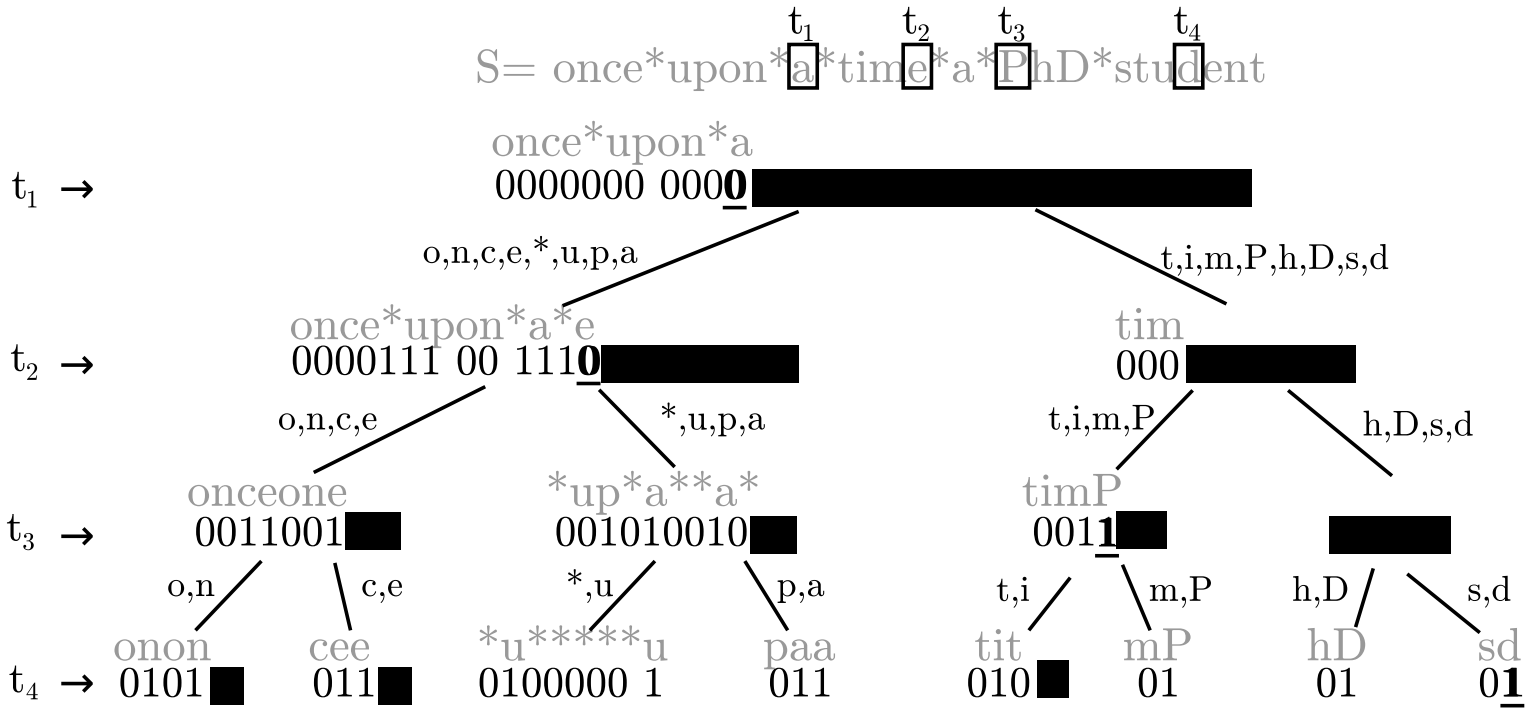


Parallel Per-level Iterative Construction



Parallel Per-level Iterative Construction

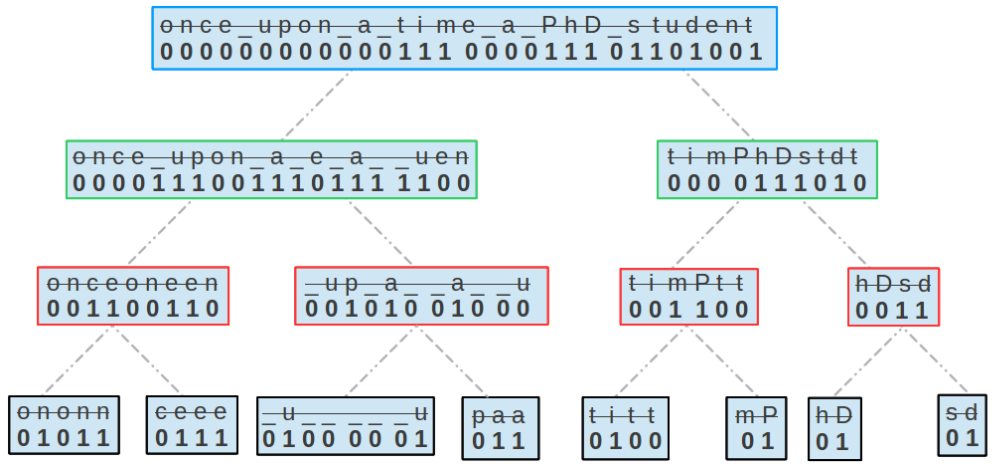
One pointer per level



Parallel Per-level Iterative Construction

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

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

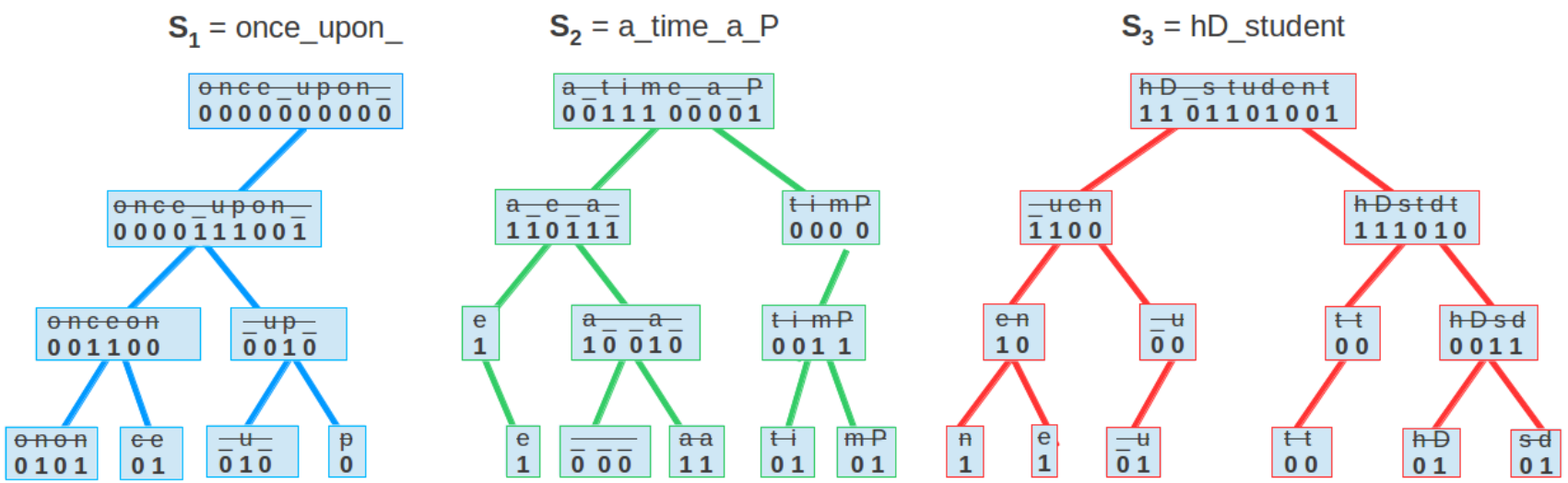


We need only $P = \lg \sigma$ to obtain the optimal speedup

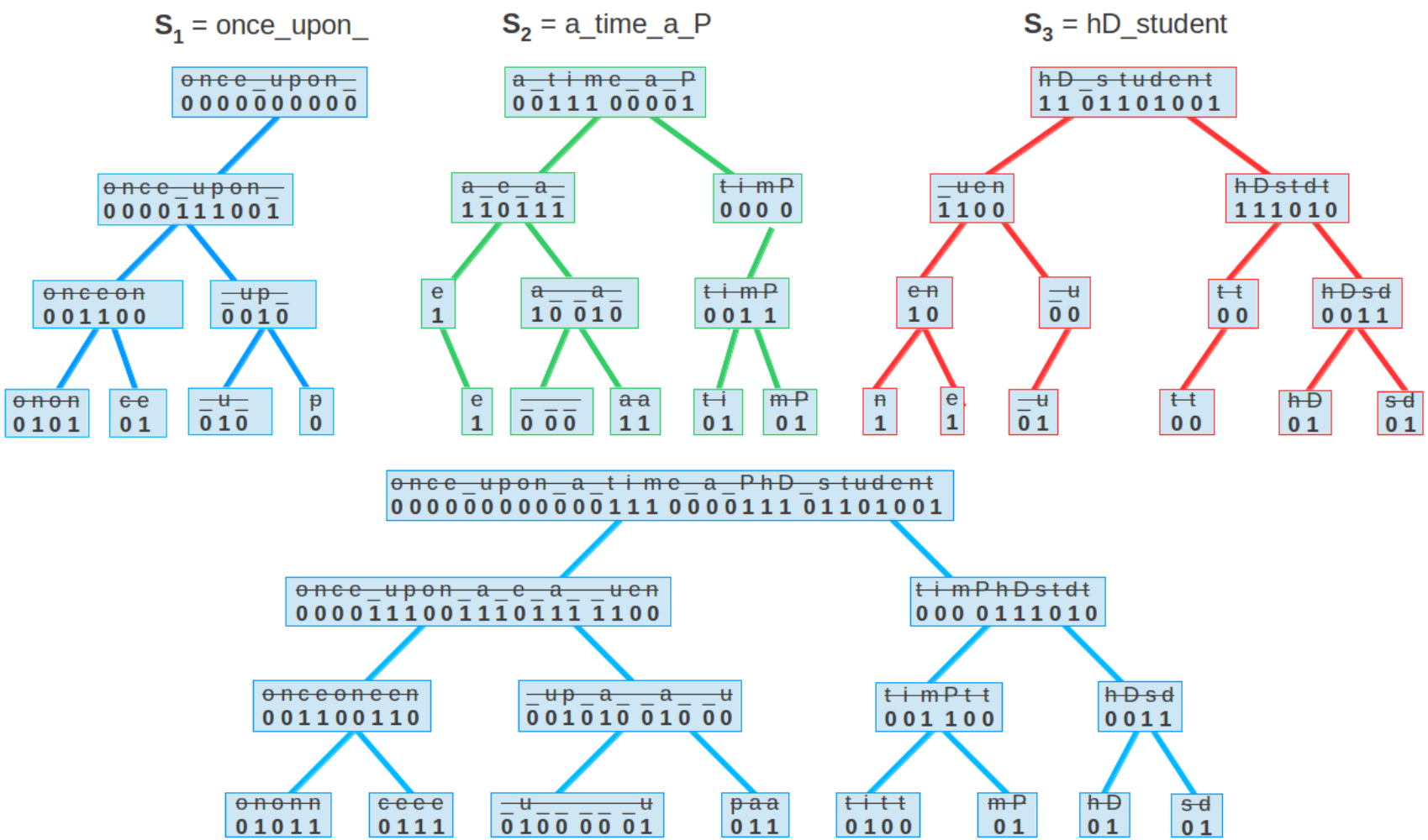
Domain Decomposition Construction

$\mathbf{S_1}$ = once_upon_ $\mathbf{S_2}$ = a_time_a_P $\mathbf{S_3}$ = hD_student

Domain Decomposition Construction

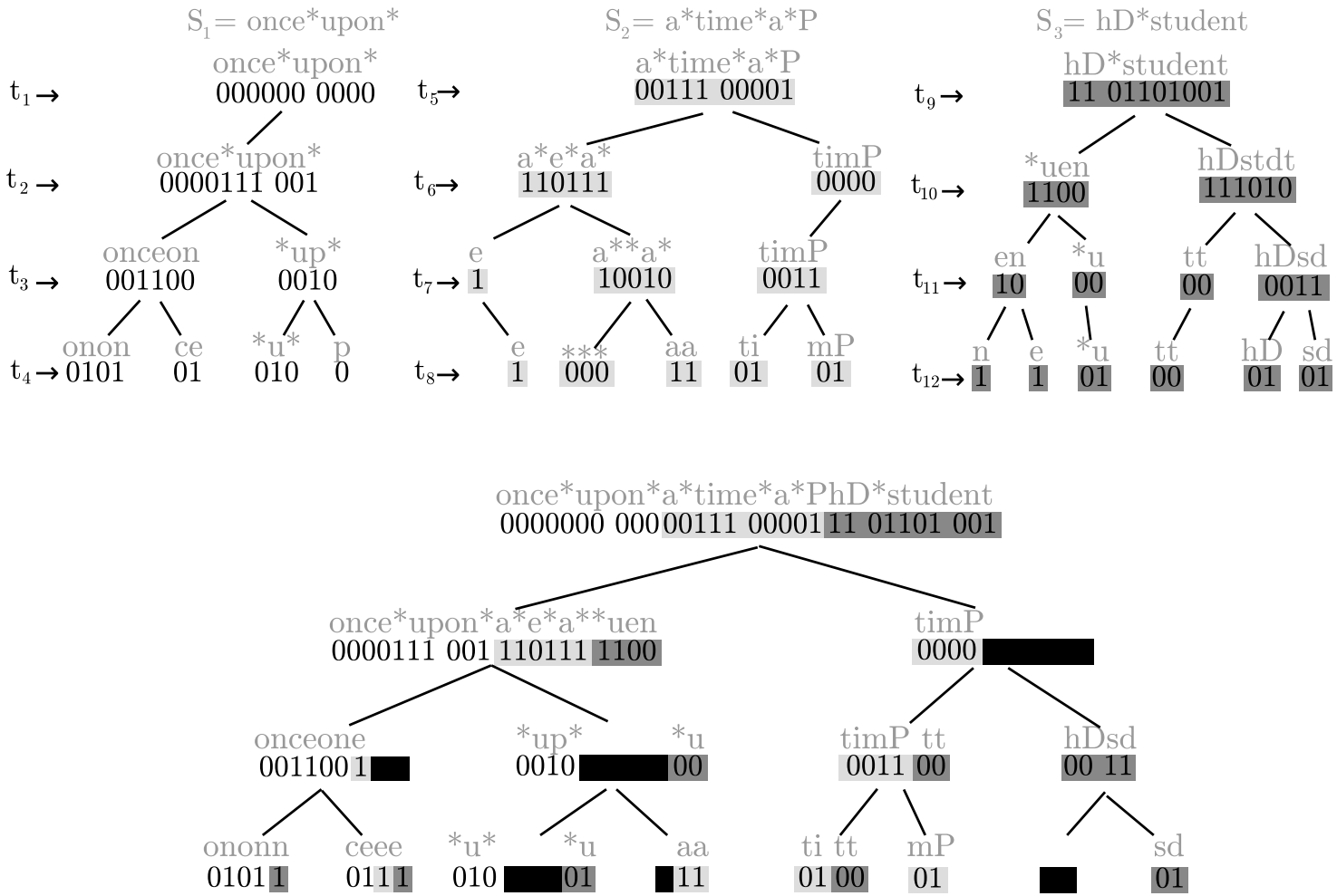


Domain Decomposition Construction



Domain Decomposition Construction

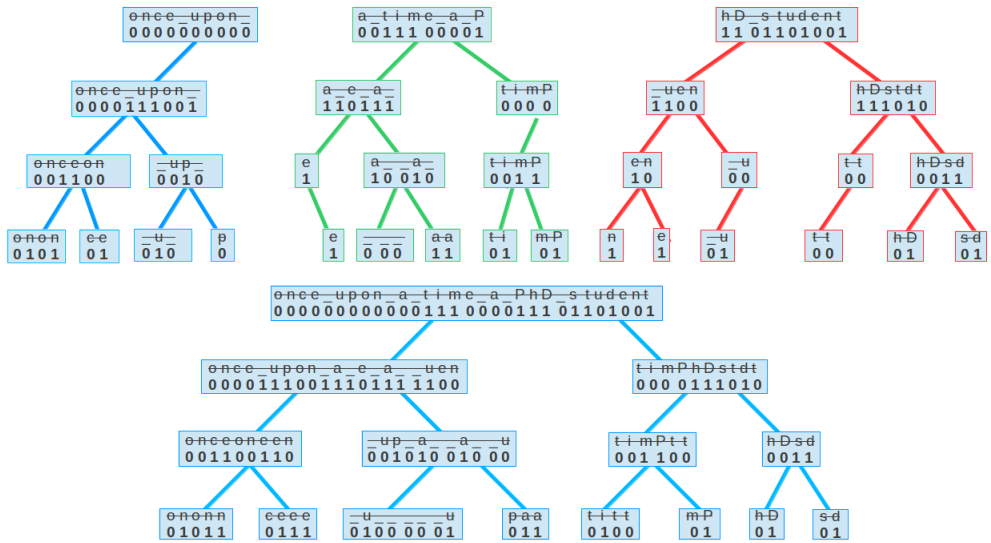
One pointer per level



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



References

- Gonzalo Navarro. *Wavelet Trees for All*. In Combinatorial Pattern Matching, volume 7354 of Lecture Notes in Computer Science, pages 2–26. Springer Berlin Heidelberg, 2012
- José Fuentes-Sepúlveda, Erick Elejalde, Leo Ferres, and Diego Seco. *Parallel construction of wavelet trees on multicore architectures*. Knowledge and Information Systems, 51(3):1043–1066, 2016.
- Julian Labeit, Julian Shun, and Guy E. Blelloch. *Parallel lightweight wavelet tree, suffix array and FM-index construction*. In Proceedings of the 2016 Data Compression Conference, DCC'16, pages 33–42, 2016.