Engineering a Distributed Full-Text Index

Johannes Fischer and *Florian Kurpicz* and Peter Sanders

Meeting on Algorithm Engineering & Experiments (ALENEX) 2017

Task

Pattern Matching Queries

Given a text T of length n and a pattern P of length m:

Existential	Does P occur in T?
Counting	How often does P occur in T?
Enumeration	Where does P occur in T?

(Full-Text) Indices

Main Memory

- Inverted Indices
- Suffix Arrays
- Suffix Trees

(Full-Text) Indices

Main Memory

- Inverted Indices
- Suffix Arrays
- Suffix Trees

(Full-Text) Indices

Main Memory

- Inverted Indices
- Suffix Arrays
- Suffix Trees

T= cabbaccab$

i	0	1	2	3	4	5	6	7	8	9
T[i..n]	c	a	b	b	a	c	c	a	b	$
	a	b	b	a	c	c	a	b	$	
	b	b	a	c	c	a	b	$		
	b	a	c	c	a	b	$			
	a	c	c	a	b	$				
	c	c	a	b	$					
	c	a	b	$						
	a	b	$							
	b	$								
	$									

(Full-Text) Indices

Main Memory

- Inverted Indices
- Suffix Arrays
- Suffix Trees

T= cabbaccab$

i	0	1	2	3	4	5	6	7	8	9
SA[i]	9	7	1	4	8	3	2	6	0	5
LCP[i]										
	$	a	a	a	b	b	b	c	c	c
		b	b	c	$	a	b	a	a	c
		$	b	c		c	a	b	b	a
			a	a		c	a	$	b	a
			c	b		a	c		a	$
			c	$		b	a		c	
			a			$	b		c	
			b				$		a	
			$						b	
									$	

(Full-Text) Indices

Main Memory

- Inverted Indices
- Suffix Arrays
- Suffix Trees

T= cabbaccab$

i	0	1	2	3	4	5	6	7	8	9
SA[i]	9	7	1	4	8	3	2	6	0	5
LCP[i]			2							
	$	a	a	a	b	b	b	c	c	c
		b	b	c	$	a	b	a	a	c
		$	b	c		c	a	b	b	a
			a	a		c	a	$	b	a
			c	b		a	c		a	$
			c	$		b	a		c	
			a			$	b		c	
			b				$		a	
			$						b	
									$	

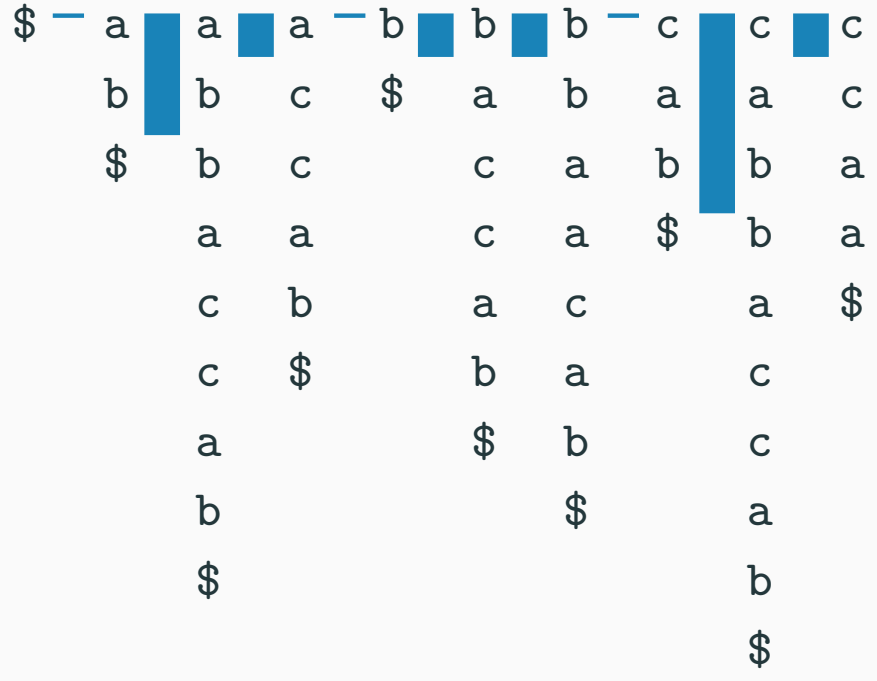
(Full-Text) Indices

Main Memory

- Inverted Indices
- Suffix Arrays
- Suffix Trees

T= cabbaccab$

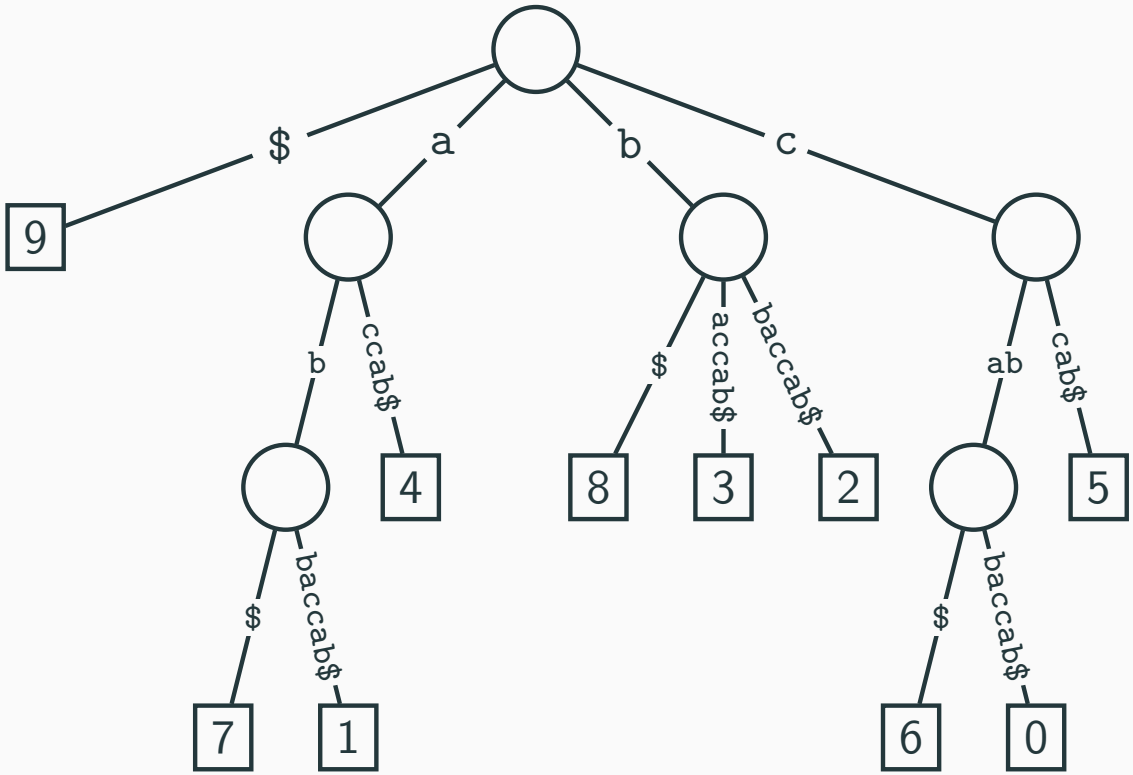
i	0	1	2	3	4	5	6	7	8	9
SA[i]	9	7	1	4	8	3	2	6	0	5
LCP[i]	0	0	2	1	0	1	1	0	3	1



(Full-Text) Indices

Main Memory

- Inverted Indices
- Suffix Arrays
- Suffix Trees

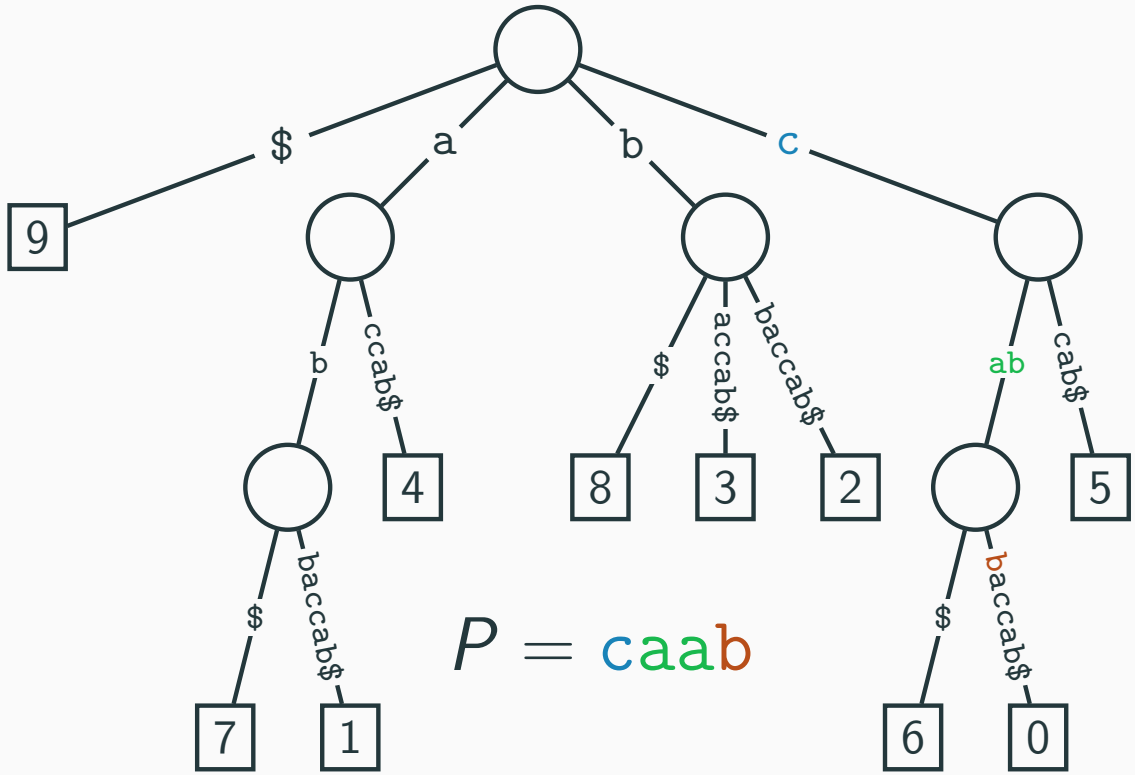


SA	9	7	1	4	8	3	2	6	0	5
LCP	0	0	2	1	0	1	1	0	3	1

(Full-Text) Indices

Main Memory

- Inverted Indices
- Suffix Arrays
- Suffix Trees



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

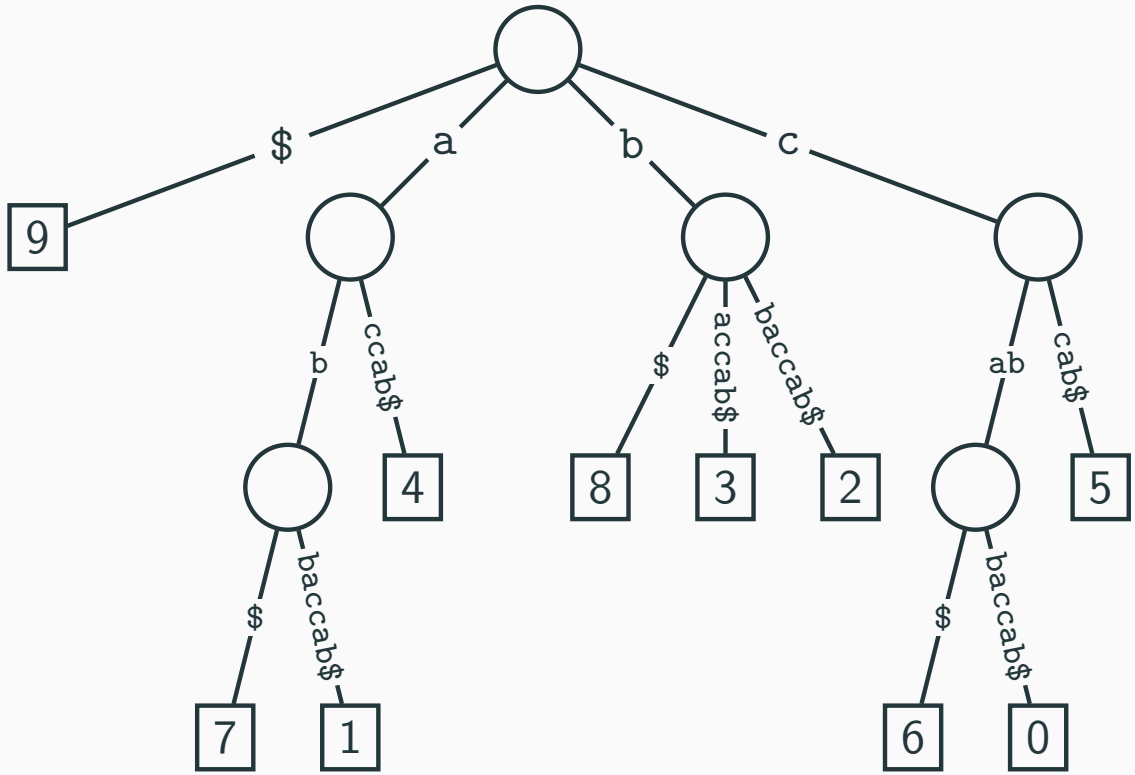
(Full-Text) Indices

Main Memory

- Inverted Indices
- Suffix Arrays
- Suffix Trees

External Memory

- Inverted Indices
- String B-Trees (Patricia Tries)



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

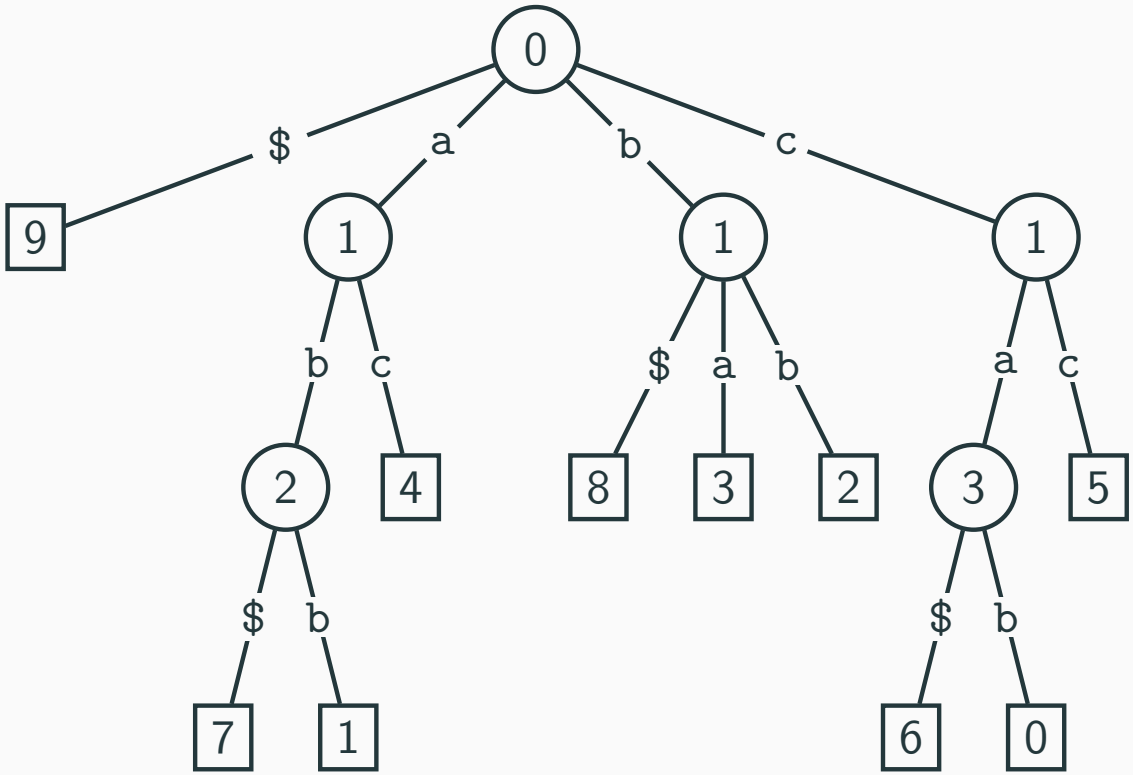
(Full-Text) Indices

Main Memory

- Inverted Indices
- Suffix Arrays
- Suffix Trees

External Memory

- Inverted Indices
- String B-Trees
  (Patricia Tries)



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

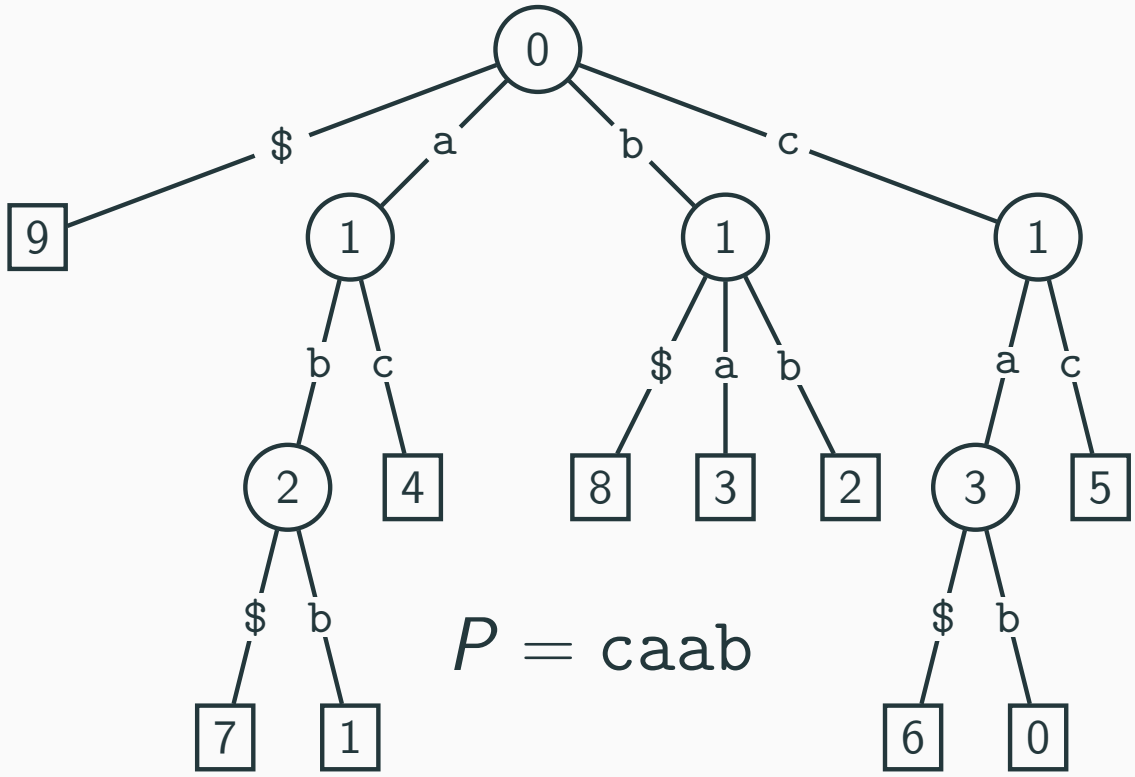
(Full-Text) Indices

Main Memory

- Inverted Indices
- Suffix Arrays
- Suffix Trees

External Memory

- Inverted Indices
- String B-Trees (Patricia Tries)



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

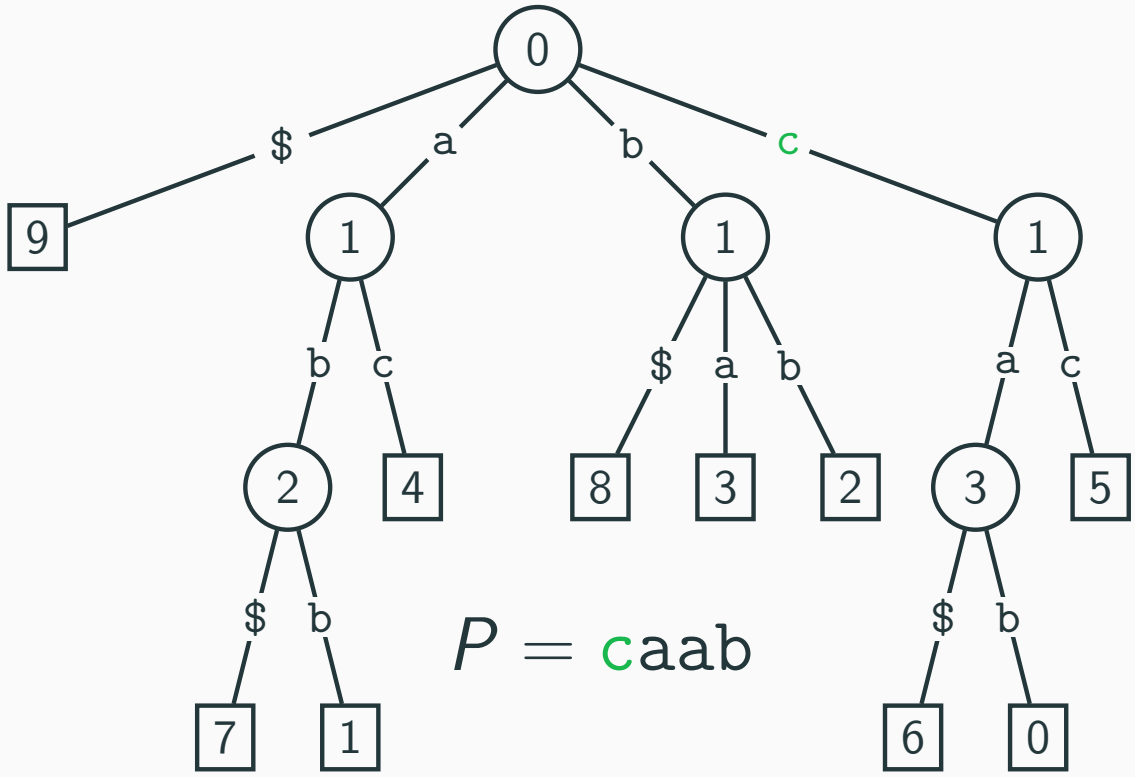
(Full-Text) Indices

Main Memory

- Inverted Indices
- Suffix Arrays
- Suffix Trees

External Memory

- Inverted Indices
- String B-Trees
  (Patricia Tries)



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

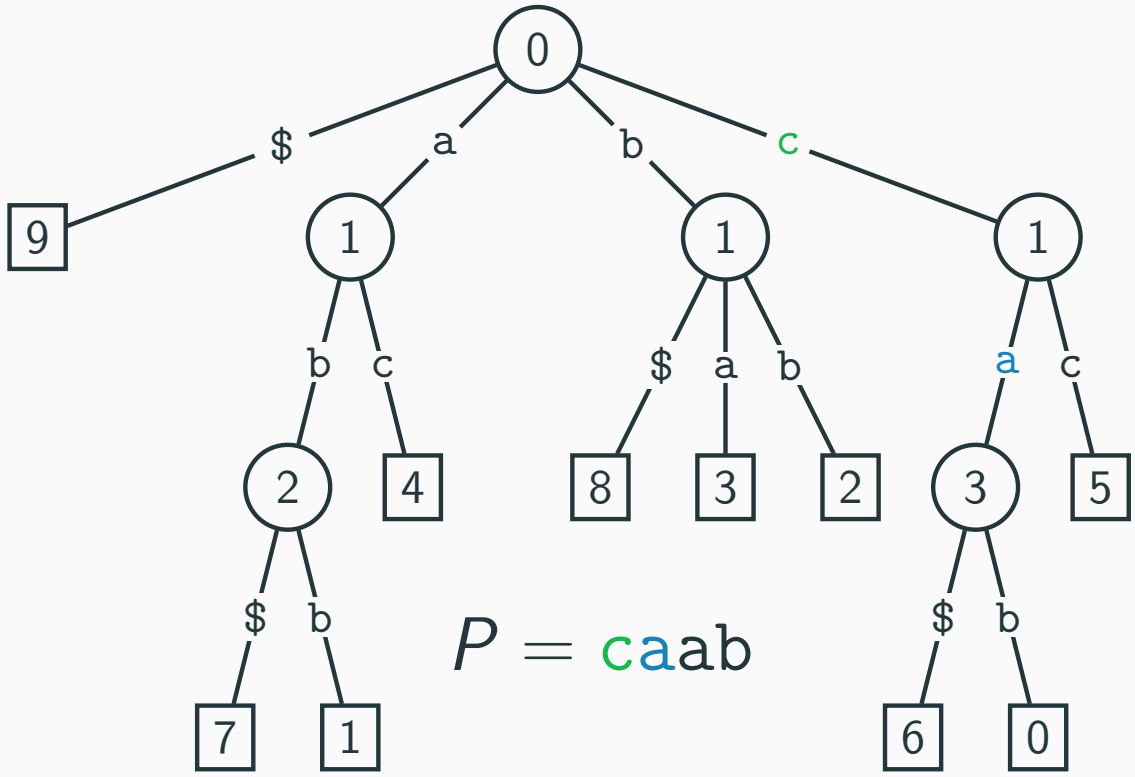
(Full-Text) Indices

Main Memory

- Inverted Indices
- Suffix Arrays
- Suffix Trees

External Memory

- Inverted Indices
- String B-Trees (Patricia Tries)



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

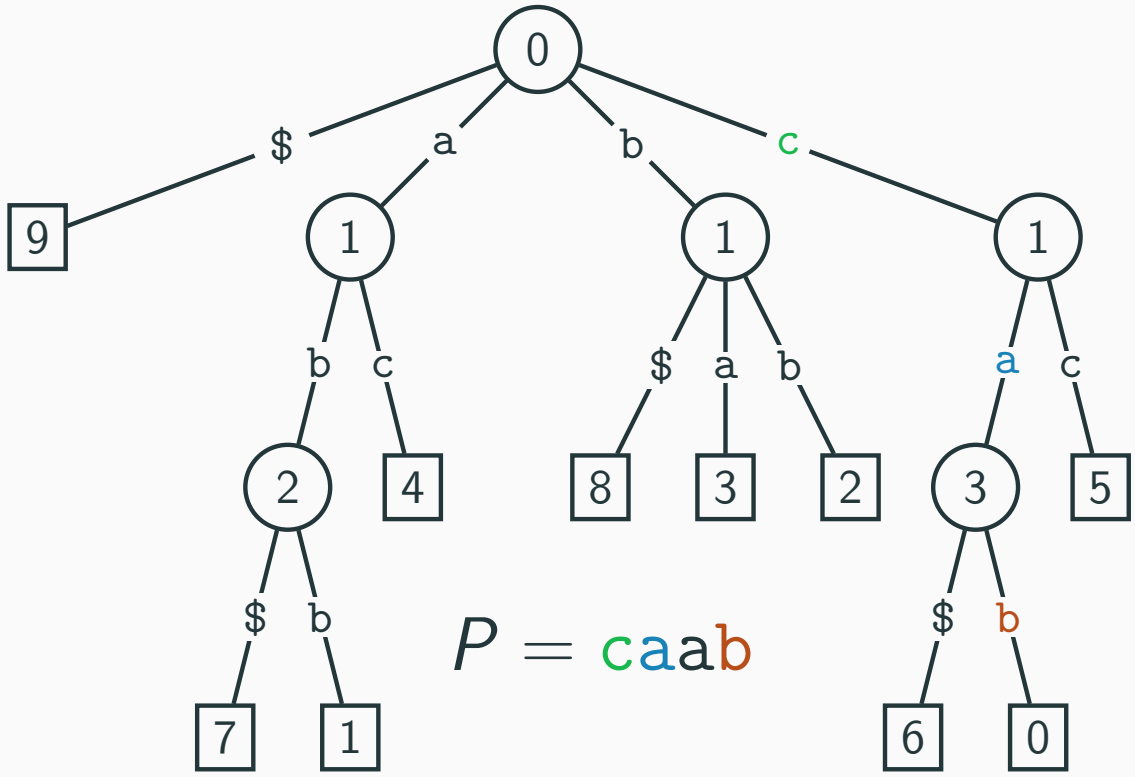
(Full-Text) Indices

Main Memory

- Inverted Indices
- Suffix Arrays
- Suffix Trees

External Memory

- Inverted Indices
- String B-Trees (Patricia Tries)



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

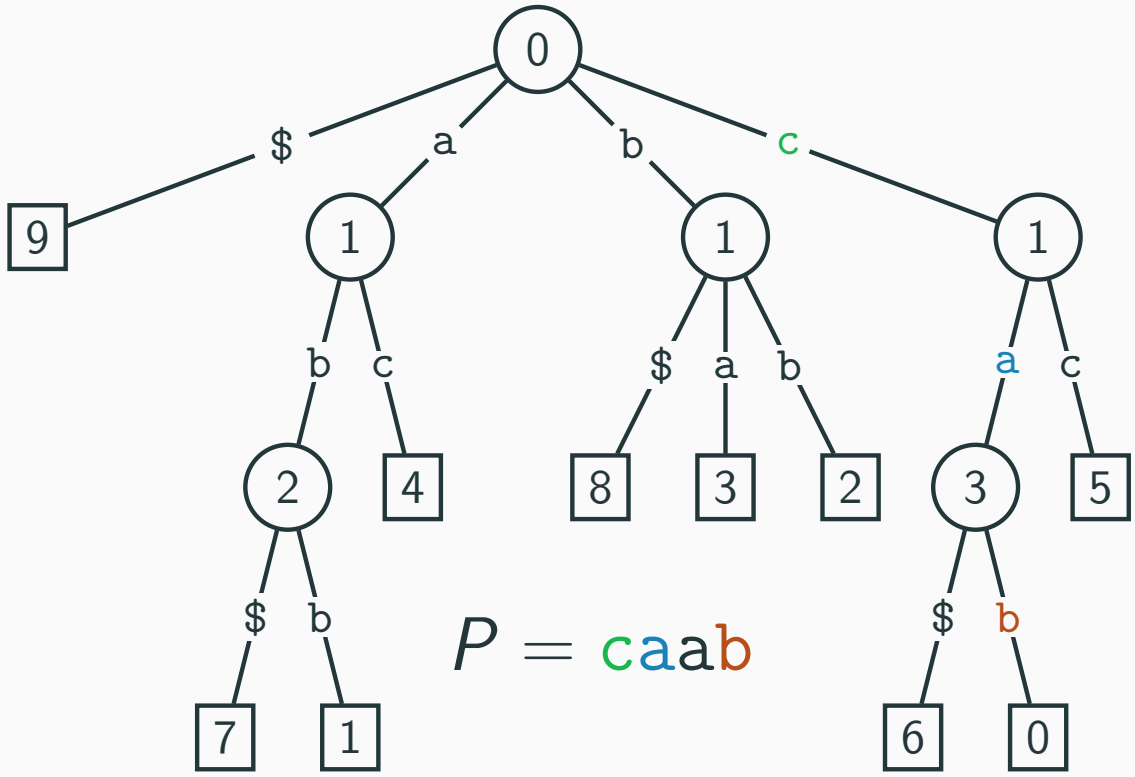
(Full-Text) Indices

Main Memory

- Inverted Indices
- Suffix Arrays
- Suffix Trees

External Memory

- Inverted Indices
- String B-Trees (Patricia Tries)



i	0	1	2	3	4	5	6	7	8	9
T	c	a	b	b	a	c	c	a	b	$

(Full-Text) Indices

Main Memory

- Inverted Indices
- Suffix Arrays
- Suffix Trees

External Memory

- Inverted Indices
- String B-Trees (Patricia Tries)

Distributed Environment

- Inverted Indices
- Distributed Suffix Array [Arroyuelo et al., 2014]

(Full-Text) Indices

Main Memory

- Inverted Indices
- Suffix Arrays
- Suffix Trees

External Memory

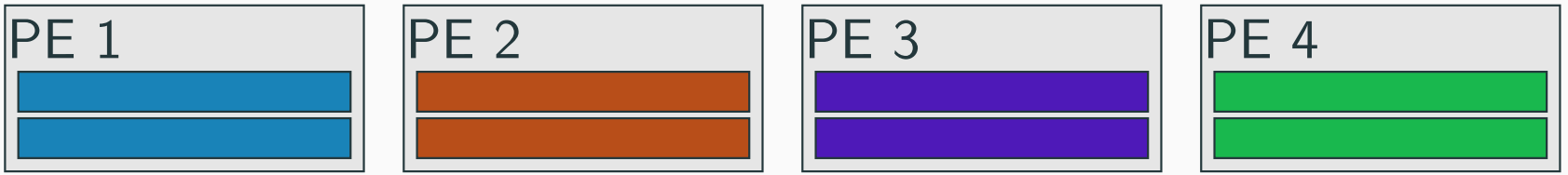
- Inverted Indices
- String B-Trees (Patricia Tries)

Distributed Environment

- Inverted Indices
- Distributed Suffix Array [Arroyuelo et al., 2014]
- Distributed Patricia Trie [This talk]

Bulk-Synchronous Parallel Model

Given p Processing Elements (PEs)



Computation = Sequence of Supersteps

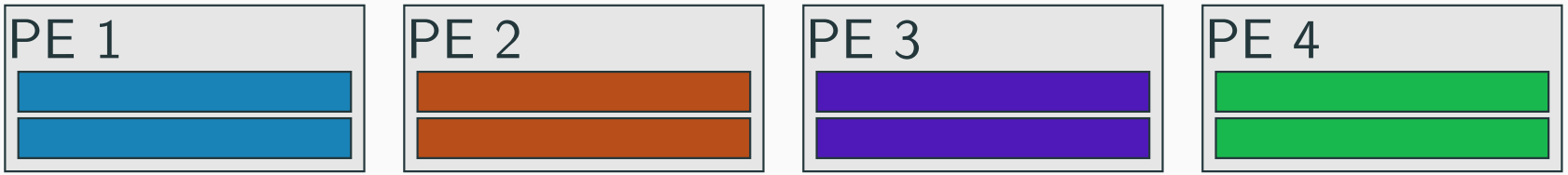
1. Operations of local data
2. Communication among PEs
3. Synchronization

Cost of Superstep

$$\mathcal{O}(\ w\ +\ hG\ +\ L\ )$$

Bulk-Synchronous Parallel Model

Given p Processing Elements (PEs)



Computation = Sequence of Supersteps

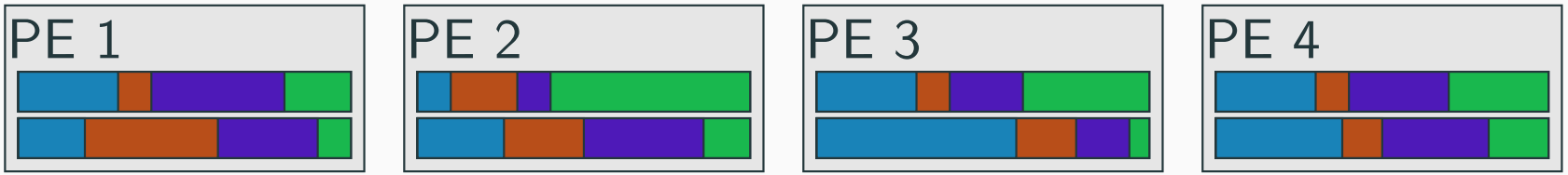
1. Operations of local data
2. Communication among PEs
3. Synchronization

Cost of Superstep

$$\mathcal{O}(\ w + hG + L\ )$$

Bulk-Synchronous Parallel Model

Given p Processing Elements (PEs)



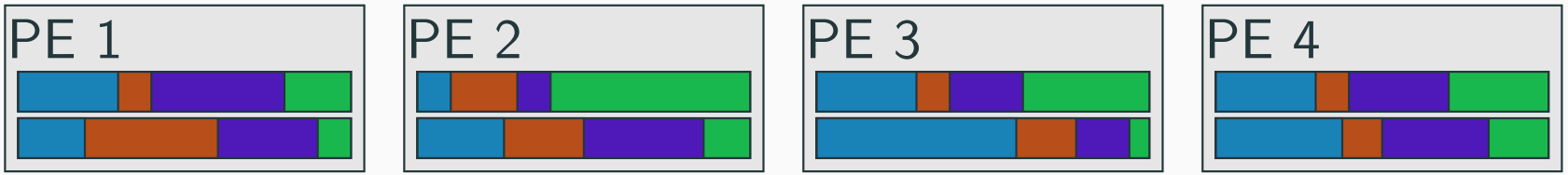
Computation = Sequence of Supersteps **Cost of Superstep**

1. Operations of local data
2. Communication among PEs
3. Synchronization

$$\mathcal{O}(\ w\ +\ hG\ +\ L\ )$$

Bulk-Synchronous Parallel Model

Given p Processing Elements (PEs)



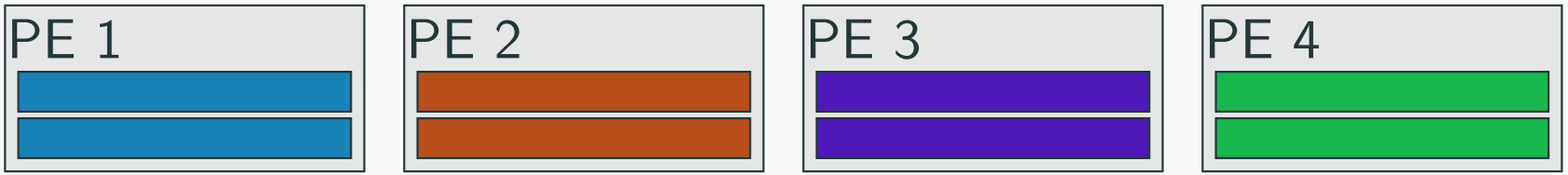
Computation = Sequence of Supersteps **Cost of Superstep**

1. Operations of local data
2. Communication among PEs
3. Synchronization

$$\mathcal{O}(\ w + hG + L\ )$$

Bulk-Synchronous Parallel Model

Given p Processing Elements (PEs)



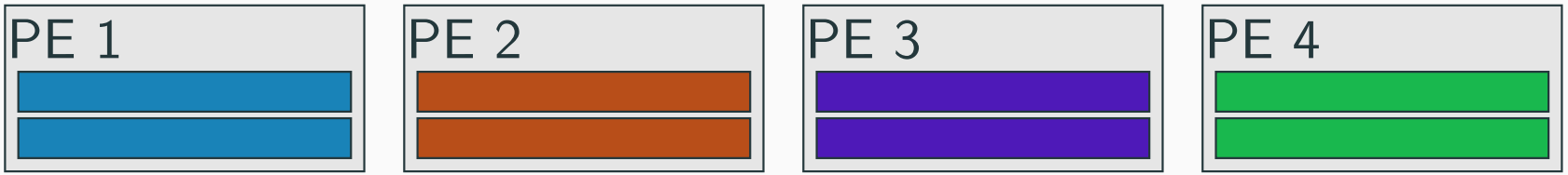
Computation = Sequence of Supersteps **Cost of Superstep**

1. Operations of local data
2. Communication among PEs
3. Synchronization

$$\mathcal{O}(\ w + hG + L\ )$$

Bulk-Synchronous Parallel Model

Given p Processing Elements (PEs)



Computation = Sequence of Supersteps **Cost of Superstep**

1. Operations of local data
2. Communication among PEs
3. Synchronization

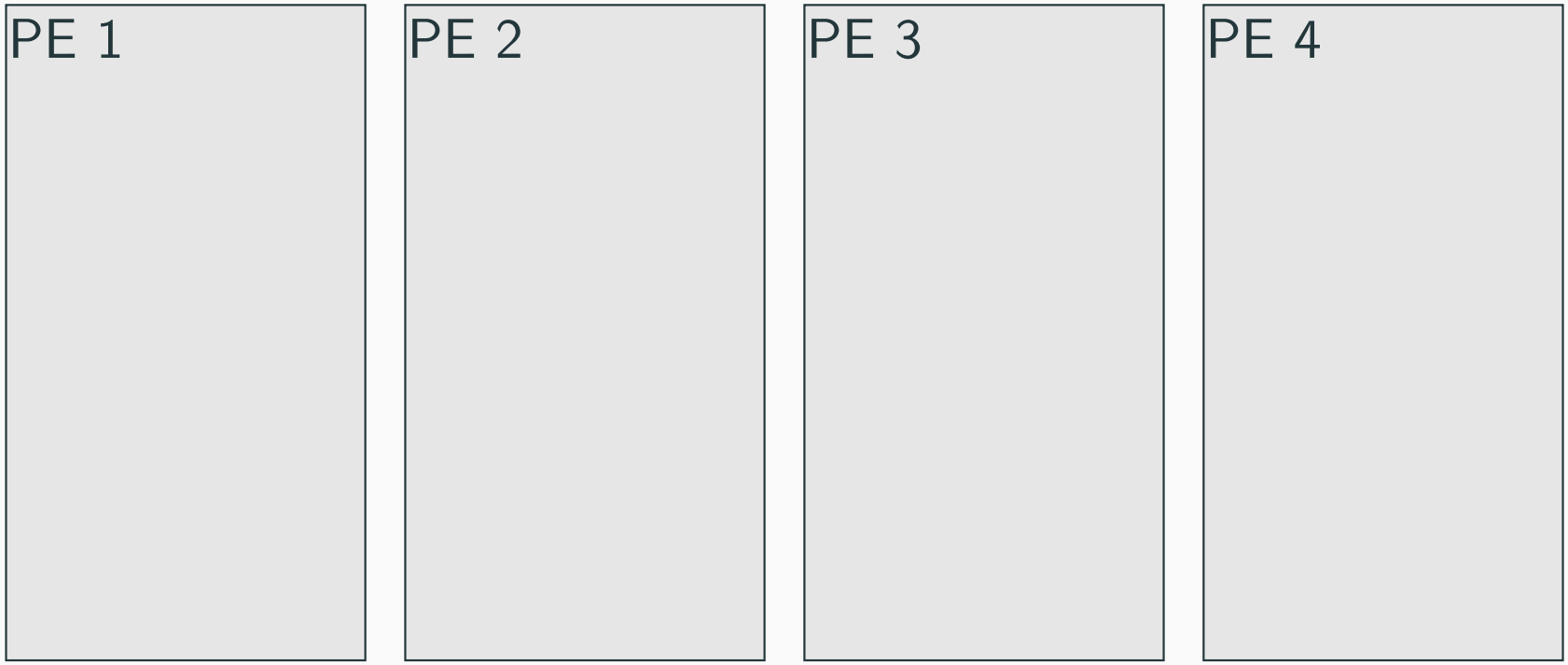
$$\mathcal{O}(\ w\ +\ hG\ +\ L\ )$$

Batch Q of Counting Queries

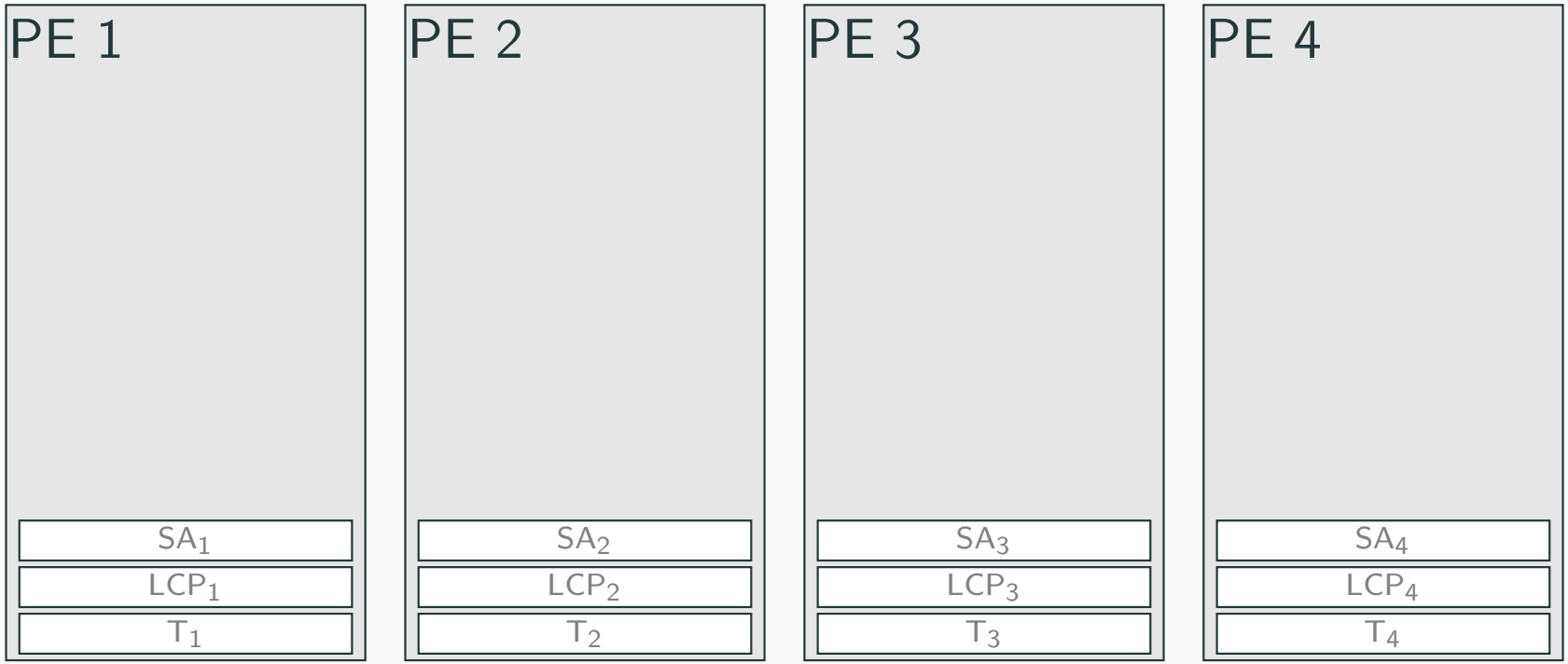
DSA $\mathcal{O}\left(\frac{1}{p}\left(\sum_{\mathrm{P}\in Q} t_{\mathrm{bin}}(\mathrm{P}) + (|\mathrm{P}|\ \lg \frac{n}{|Q|} + \lg p)\,G\right) + \lg \frac{pn}{|Q|}\,L\right)$

DPT $\mathcal{O}\left(\frac{1}{p}\left(\sum_{\mathrm{P}\in Q} t_{\mathrm{trie}}(\mathrm{P}) + |\mathrm{P}|\ G\right) + L\right)$

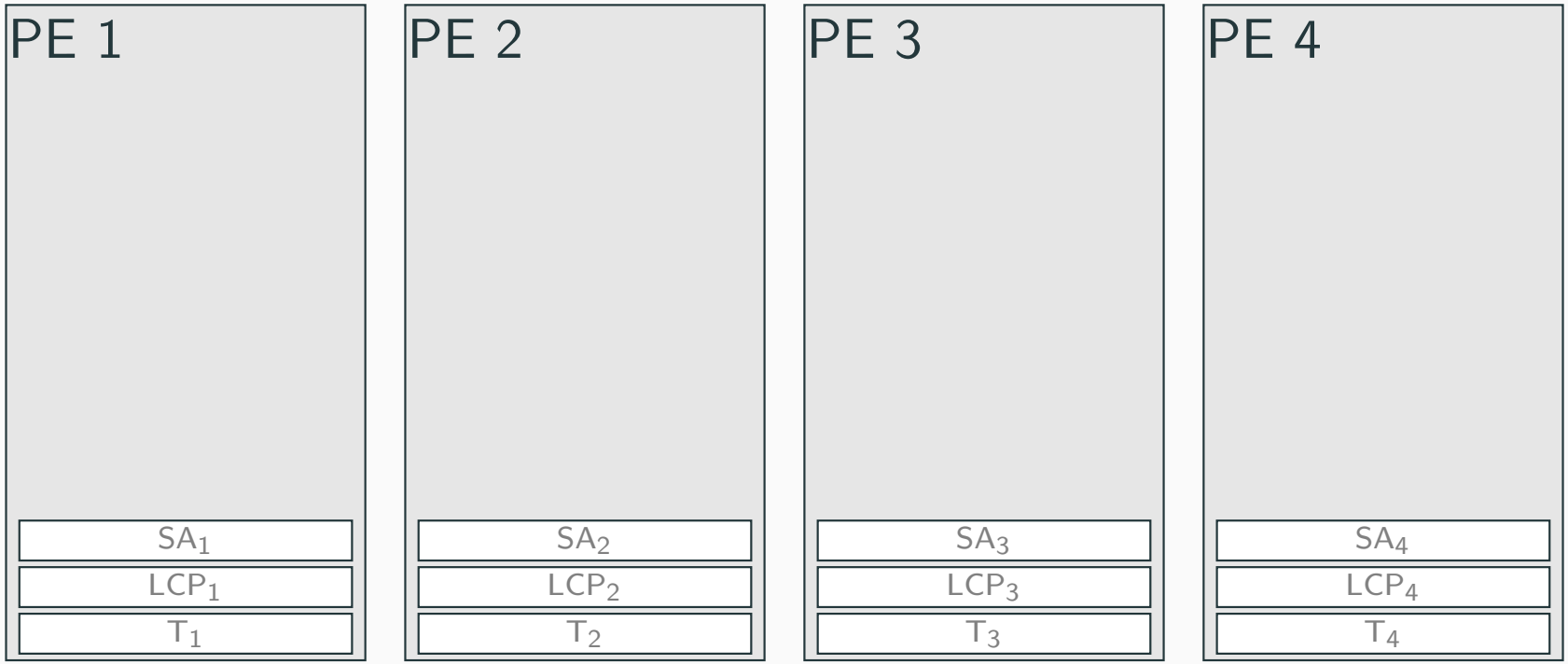
Distributed Patricia Trie



Distributed Patricia Trie



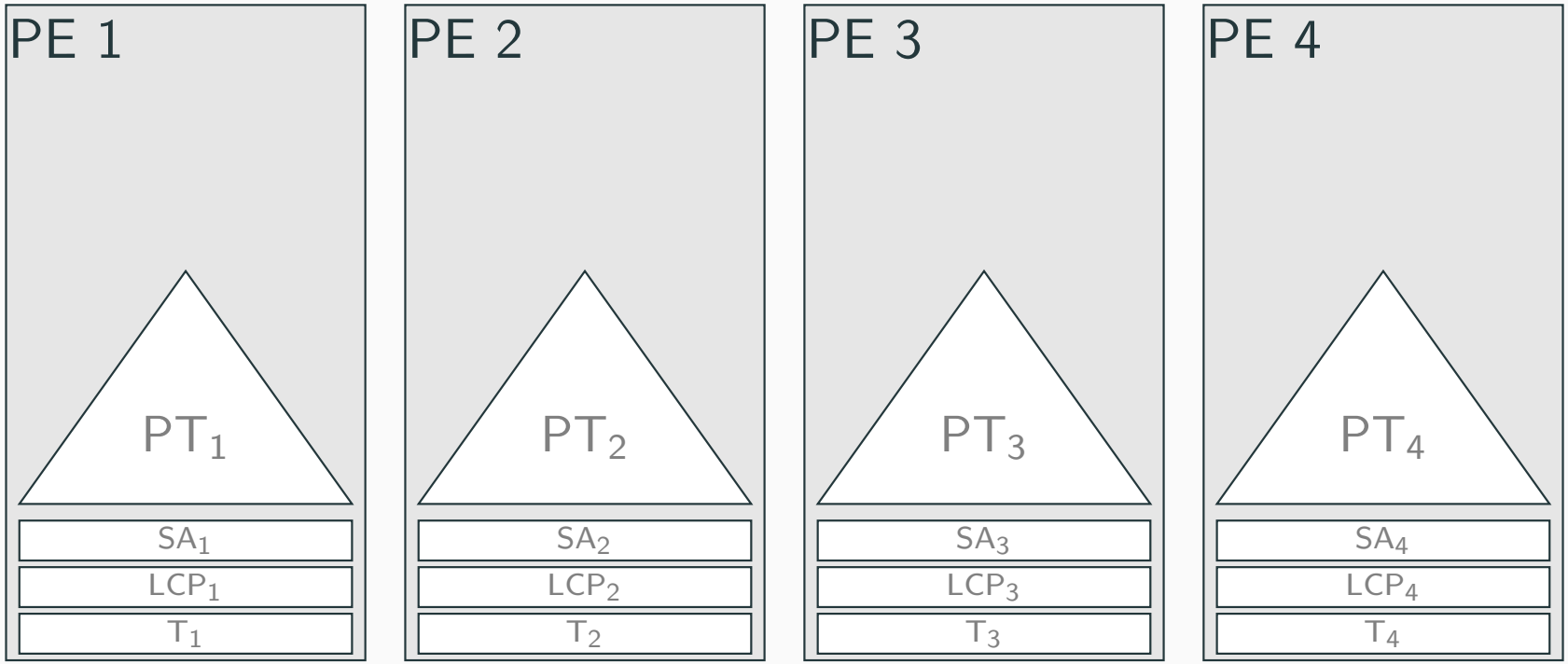
Distributed Patricia Trie



Two Levels

- Lower level to answer queries
- Upper level for query distribution

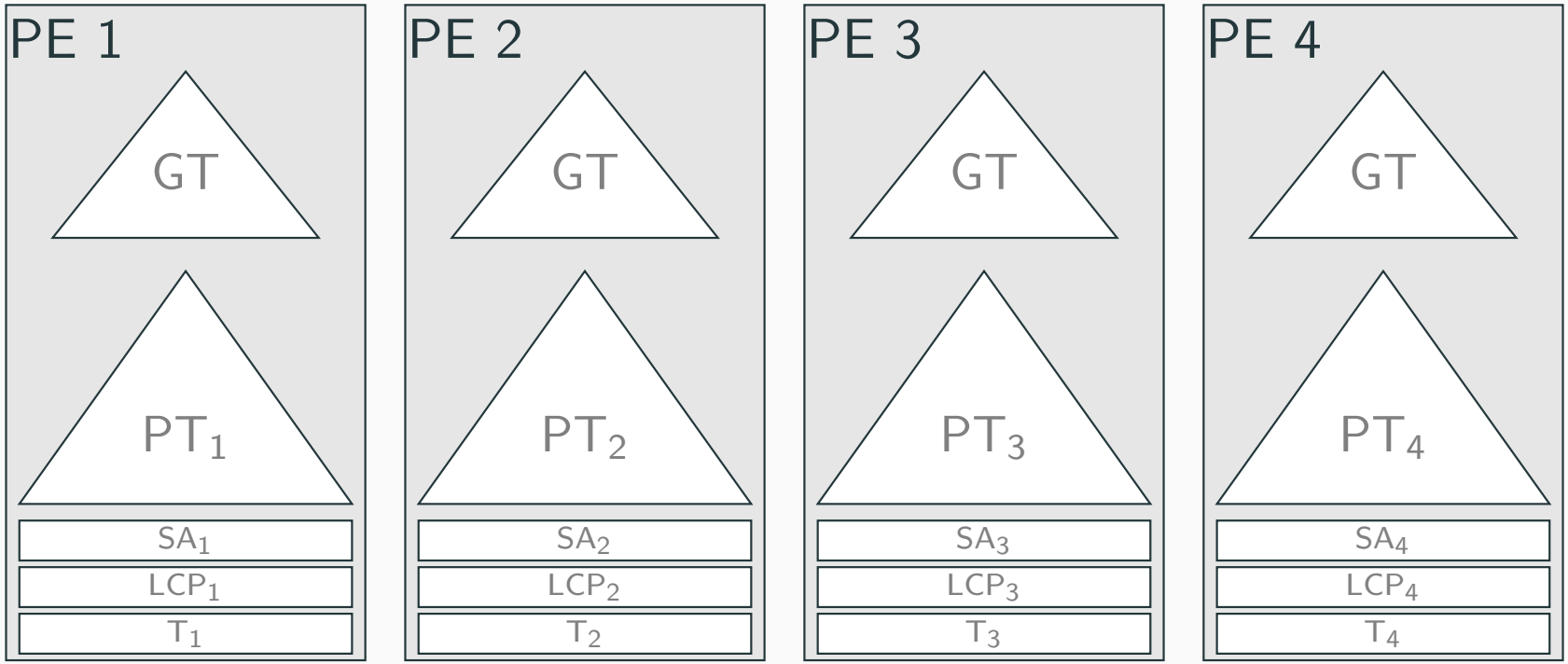
Distributed Patricia Trie



Two Levels

- Lower level to answer queries
- Upper level for query distribution

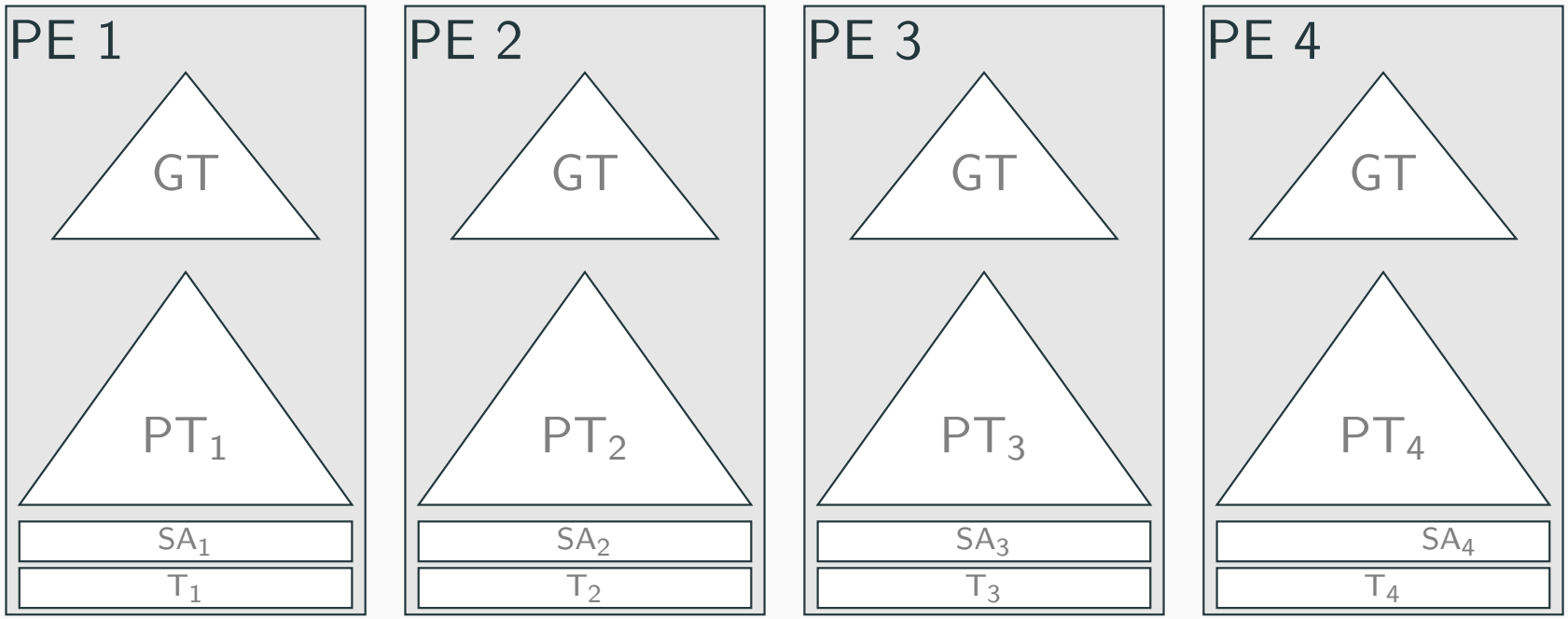
Distributed Patricia Trie



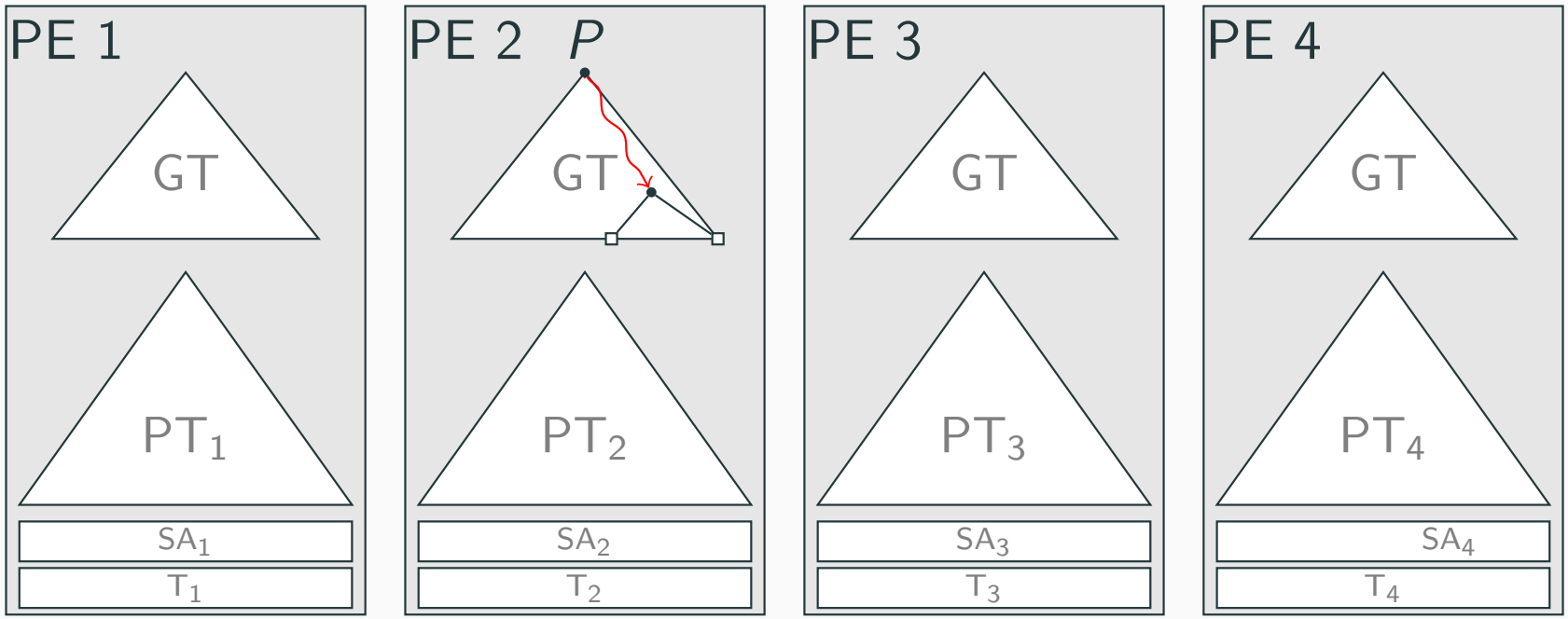
Two Levels

- Lower level to answer queries
- Upper level for query distribution

Querying – Counting Query

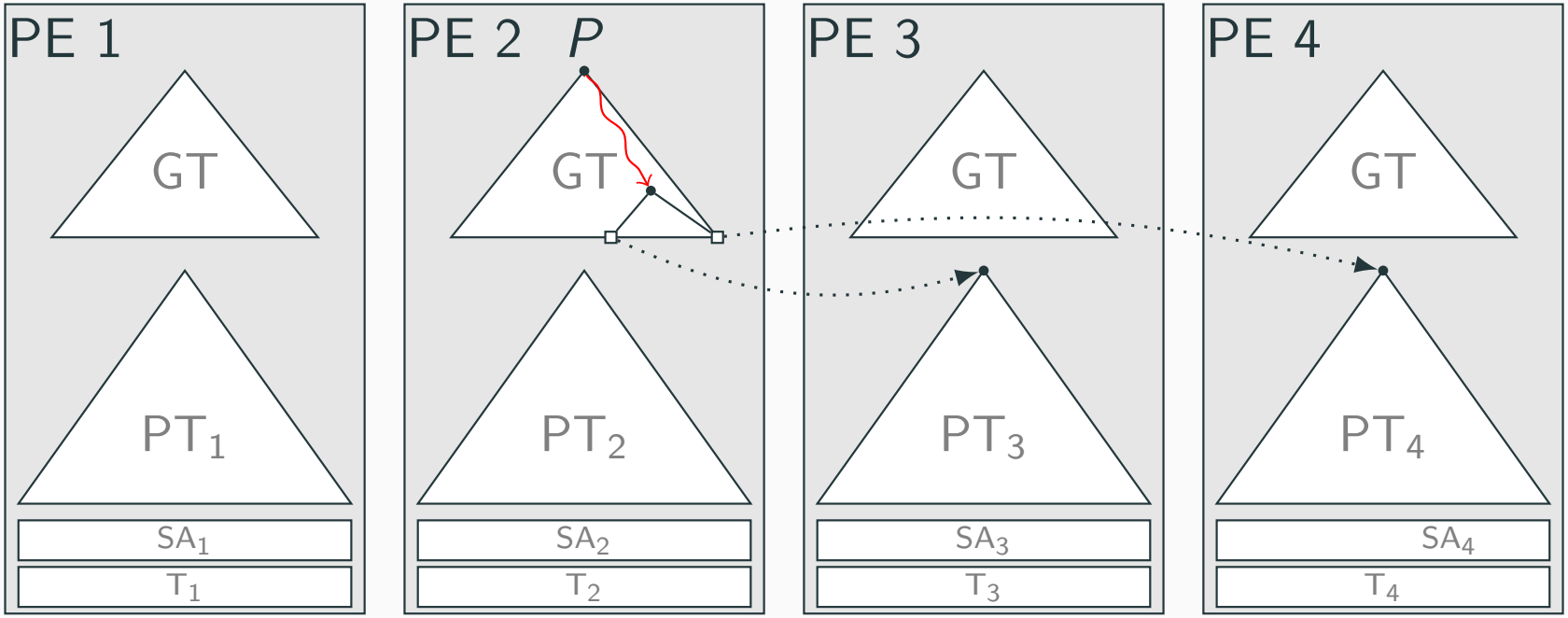


Querying – Counting Query



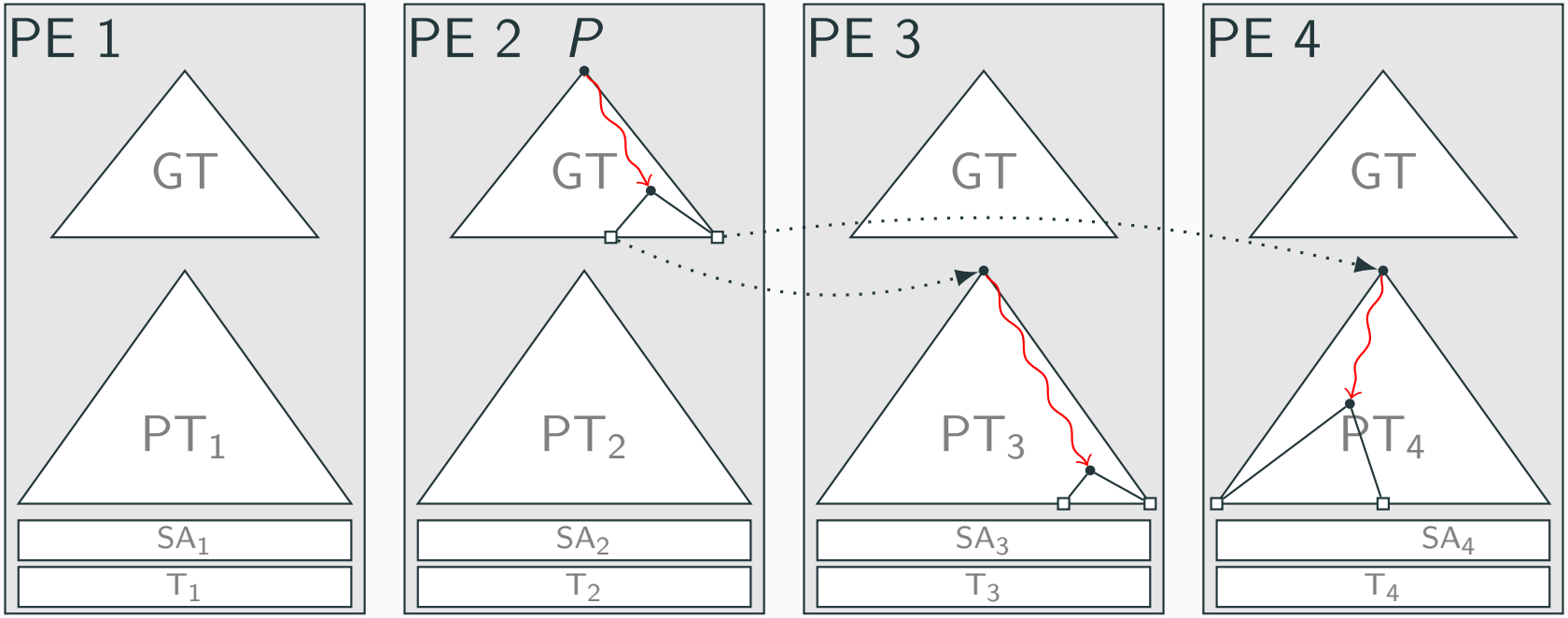
1. Pattern arrives
2. Identify PTs $\mathcal{O}(t_{\text{trie}}(P))$

Querying – Counting Query



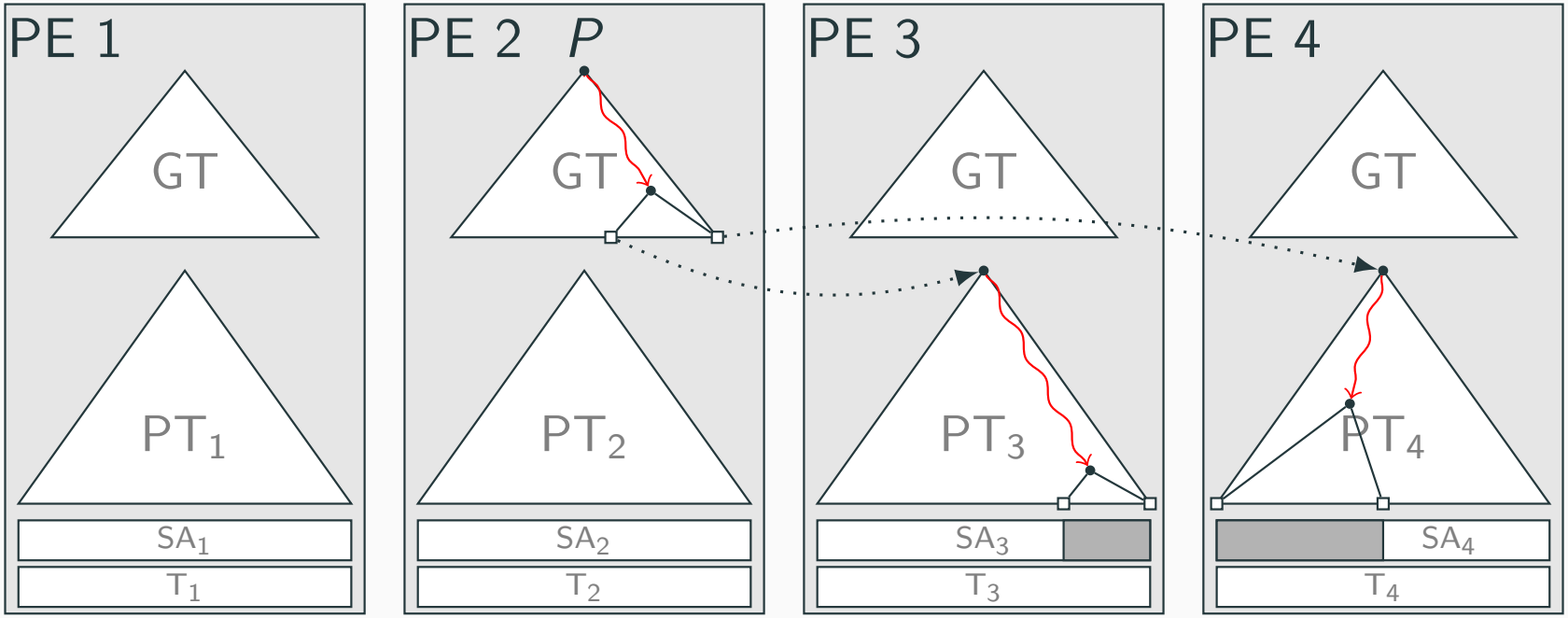
1. Pattern arrives
2. Identify PTs $\mathcal{O}(t_{\text{trie}}(P))$
3. Send to PTs $\mathcal{O}(|P|G)$

Querying – Counting Query



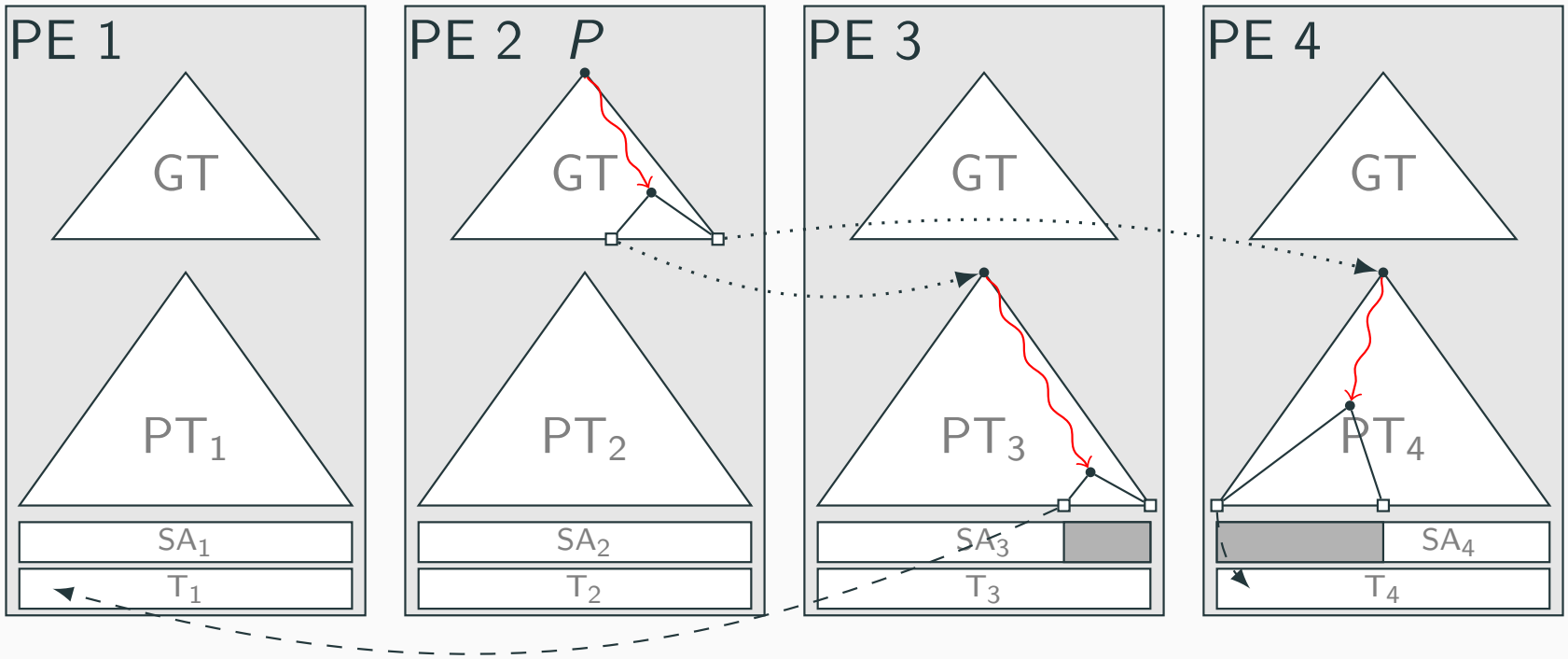
1. Pattern arrives
2. Identify PTs $\mathcal{O}(t_{\text{trie}}(P))$
3. Send to PTs $\mathcal{O}(|P|G)$
4. Blind search $\mathcal{O}(t_{\text{trie}}(P))$

Querying – Counting Query



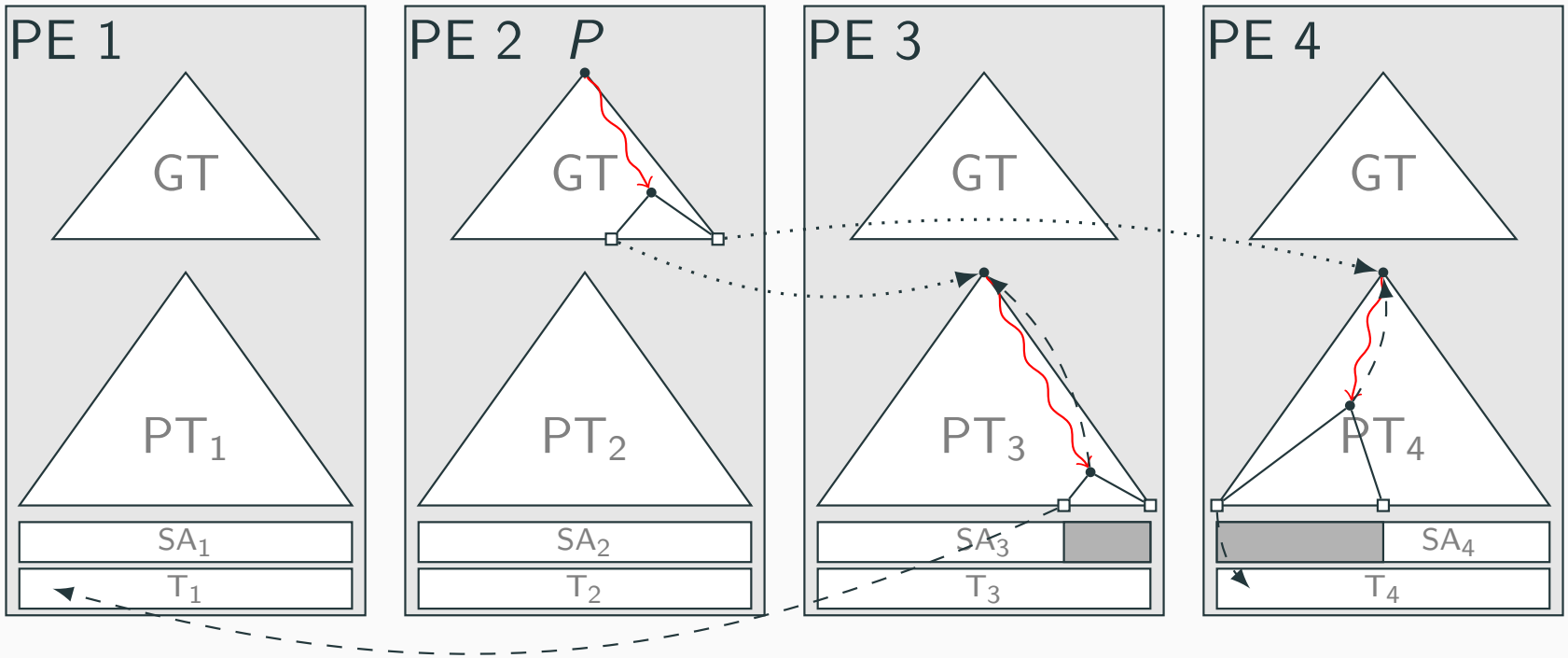
1. Pattern arrives
2. Identify PTs $\mathcal{O}(t_{\text{trie}}(P))$
3. Send to PTs $\mathcal{O}(|P|G)$
4. Blind search $\mathcal{O}(t_{\text{trie}}(P))$

Querying – Counting Query



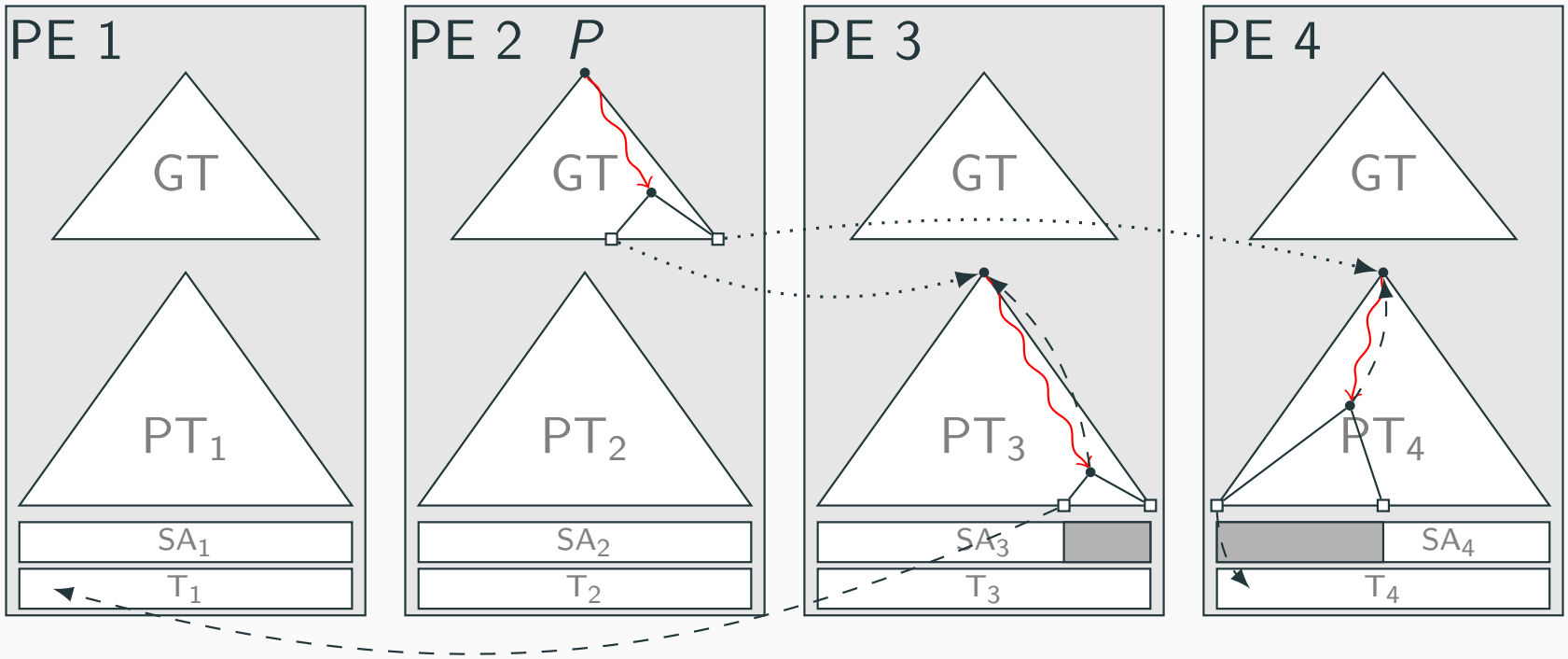
1. Pattern arrives
2. Identify PTs $\mathcal{O}(t_{\text{trie}}(P))$
3. Send to PTs $\mathcal{O}(|P|G)$
4. Blind search $\mathcal{O}(t_{\text{trie}}(P))$
5. Get substring $\mathcal{O}(|P|G)$

Querying – Counting Query



1. Pattern arrives
2. Identify PTs $\mathcal{O}(t_{\text{trie}}(P))$
3. Send to PTs $\mathcal{O}(|P|G)$
4. Blind search $\mathcal{O}(t_{\text{trie}}(P))$
5. Get substring $\mathcal{O}(|P|G)$
6. Return Result $\mathcal{O}(G)$

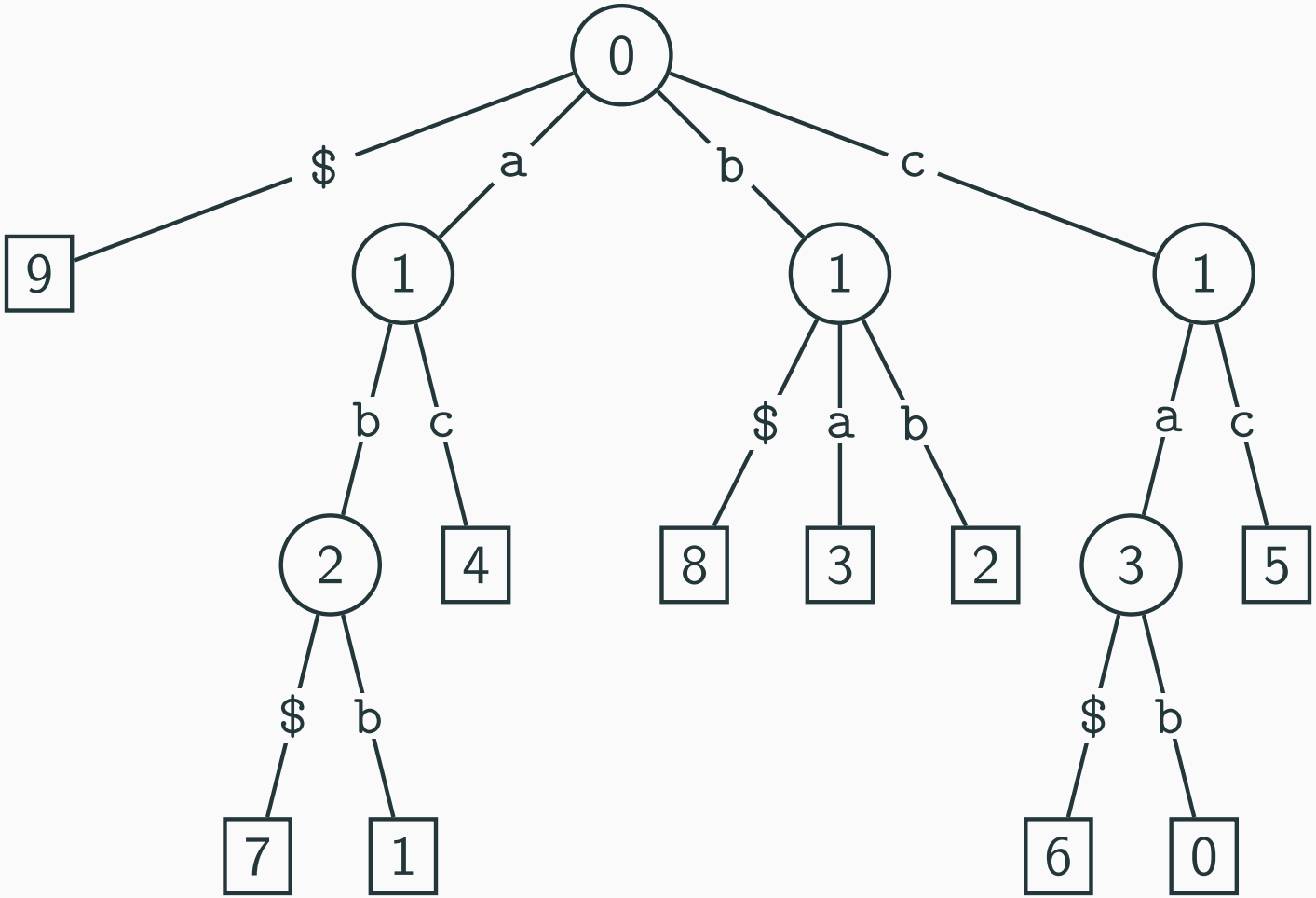
Querying – Counting Query



1. Pattern arrives
2. Identify PTs $\mathcal{O}(t_{\text{trie}}(P))$
3. Send to PTs $\mathcal{O}(|P|G)$
4. Blind search $\mathcal{O}(t_{\text{trie}}(P))$
5. Get substring $\mathcal{O}(|P|G)$
6. Return Result $\mathcal{O}(G)$

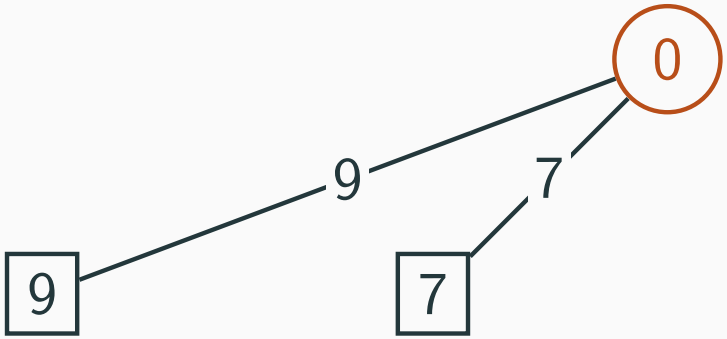
Total time: $\mathcal{O}(t_{\text{trie}}(P) + |P|G + L)$

Construction – Local Patricia Trie



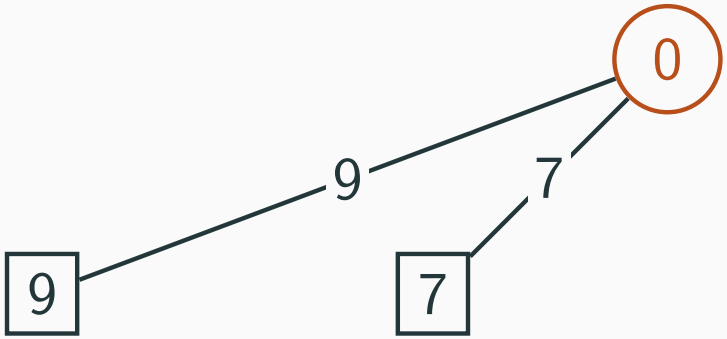
SA	9	7	1	4	8	3	2	6	0	5
LCP	0	0	2	1	0	1	1	0	3	1

Construction – Local Patricia Trie



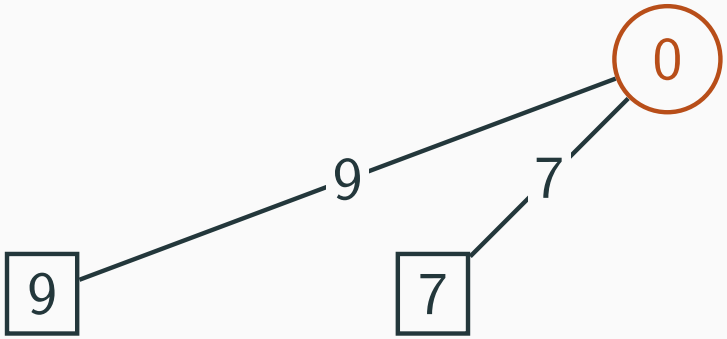
SA	9	7	1	4	8	3	2	6	0	5
LCP	0	0	2	1	0	1	1	0	3	1

Construction – Local Patricia Trie



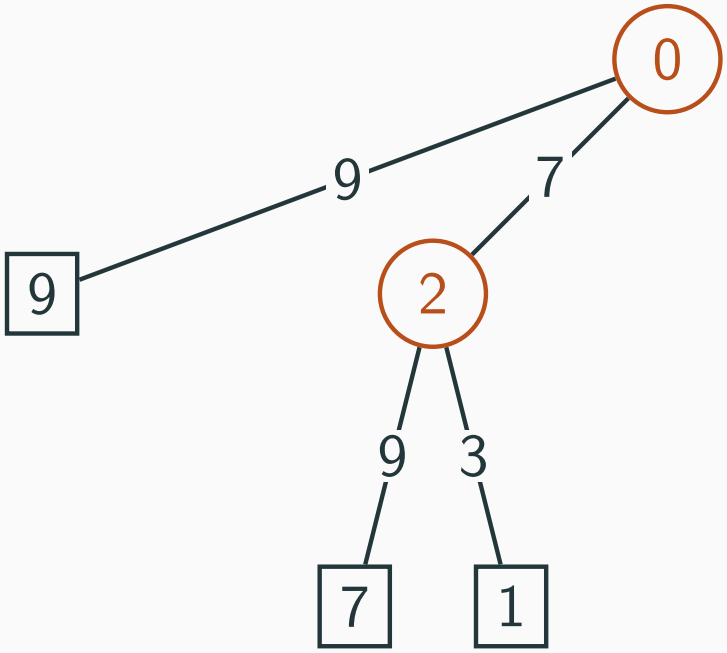
SA	9	7	1	4	8	3	2	6	0	5
LCP	0	0	2	1	0	1	1	0	3	1

Construction – Local Patricia Trie



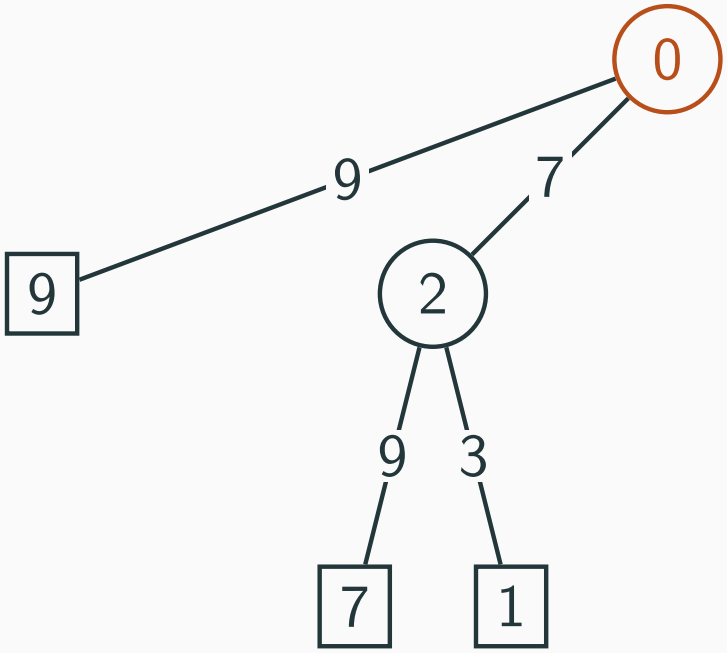
SA	9	7	1	4	8	3	2	6	0	5
LCP	0	0	2	1	0	1	1	0	3	1

Construction – Local Patricia Trie



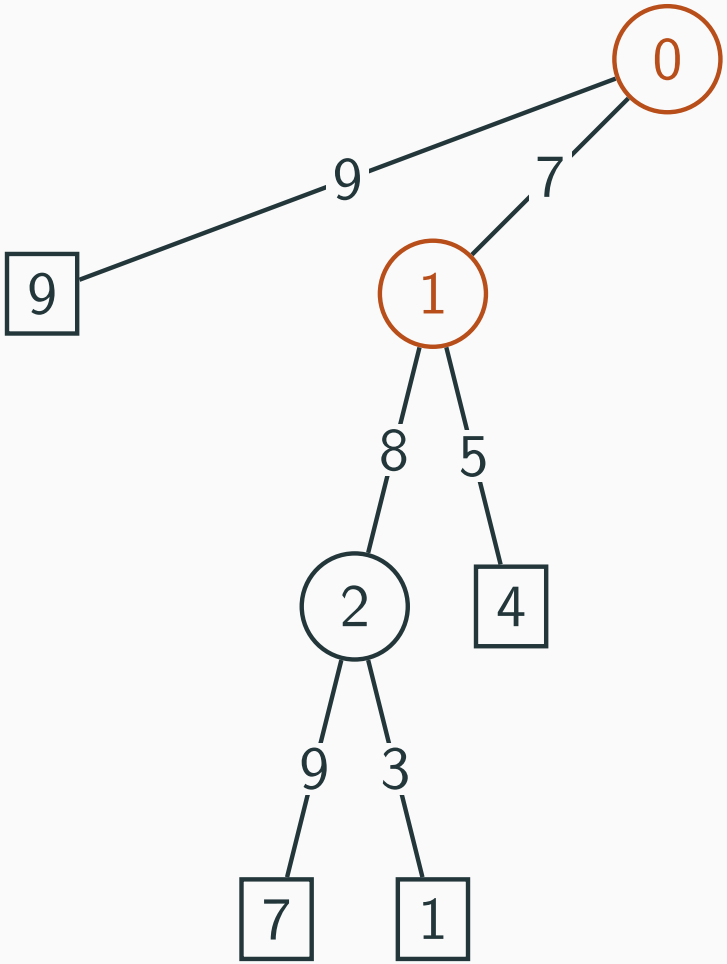
SA	9	7	1	4	8	3	2	6	0	5
LCP	0	0	2	1	0	1	1	0	3	1

Construction – Local Patricia Trie



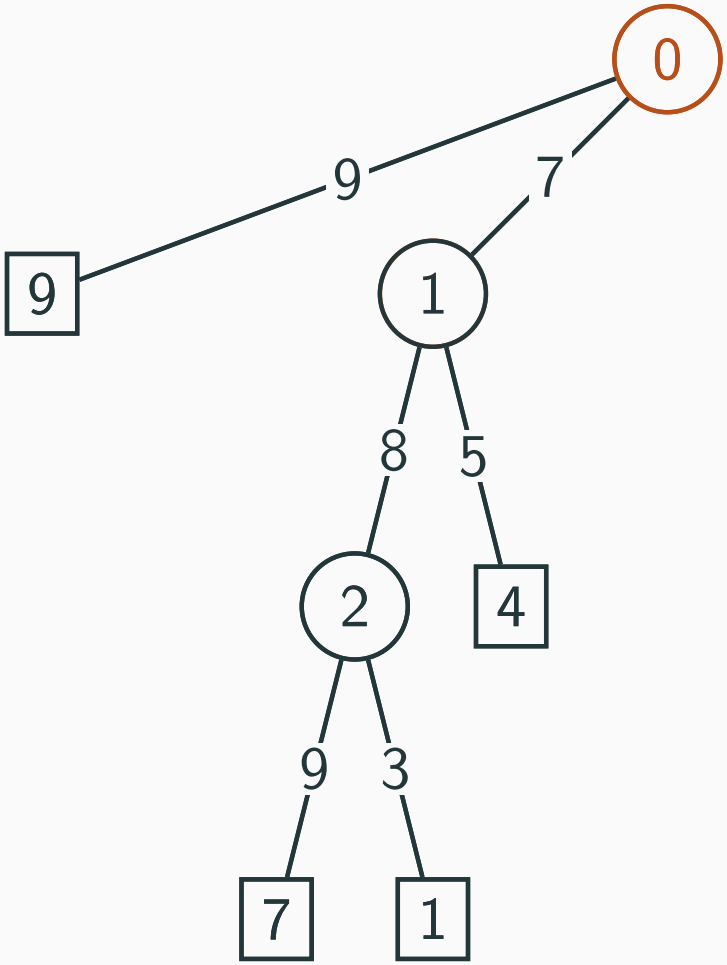
SA	9	7	1	4	8	3	2	6	0	5
LCP	0	0	2	1	0	1	1	0	3	1

Construction – Local Patricia Trie



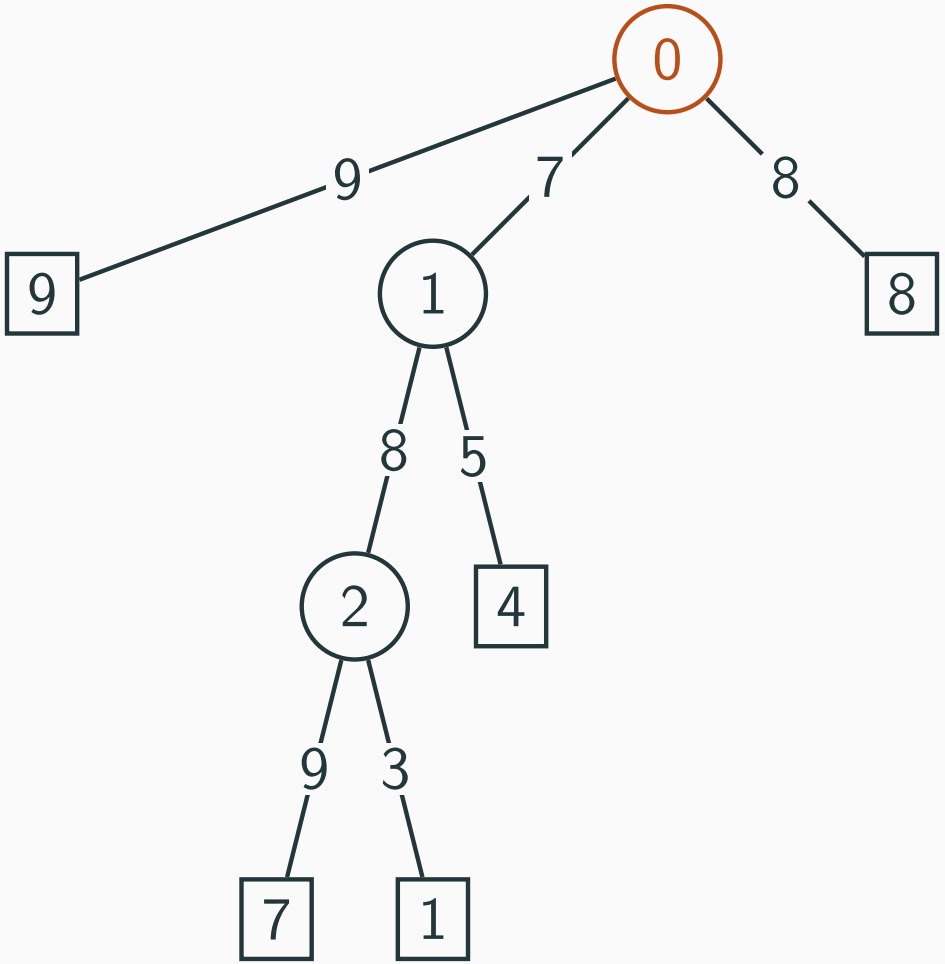
SA	9	7	1	4	8	3	2	6	0	5
LCP	0	0	2	1	0	1	1	0	3	1

Construction – Local Patricia Trie



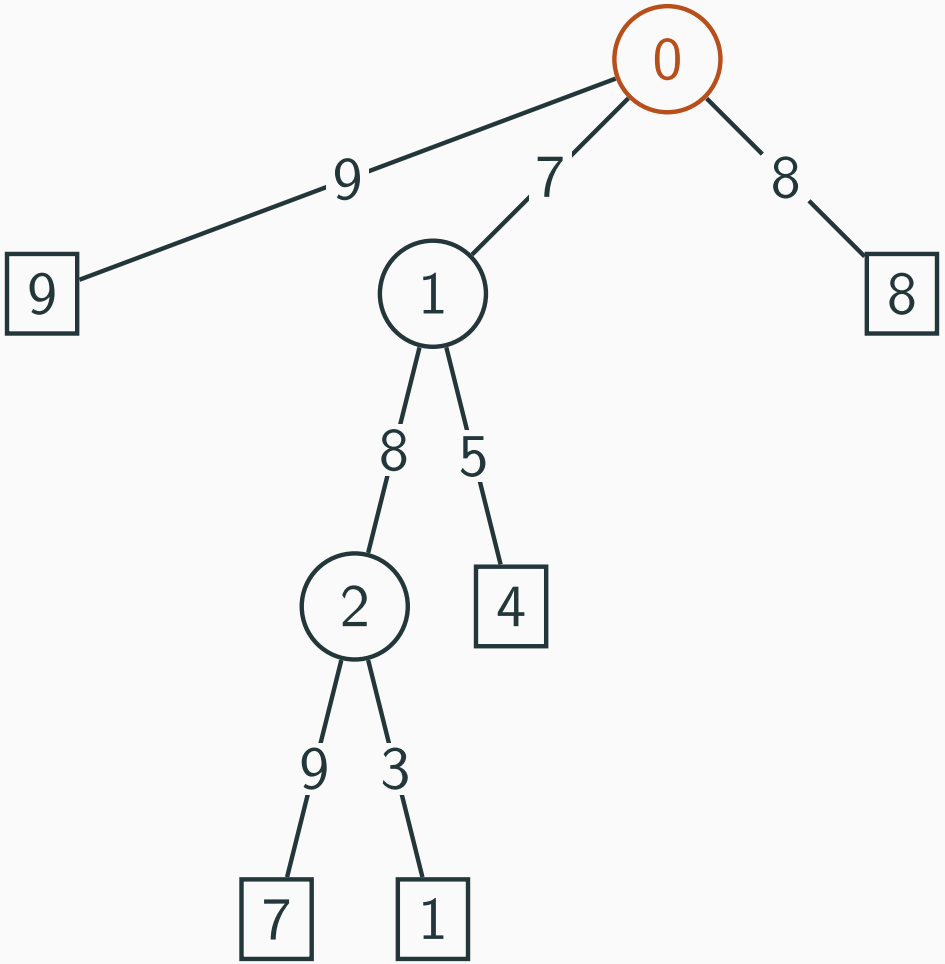
SA	9	7	1	4	8	3	2	6	0	5
LCP	0	0	2	1	0	1	1	0	3	1

Construction – Local Patricia Trie



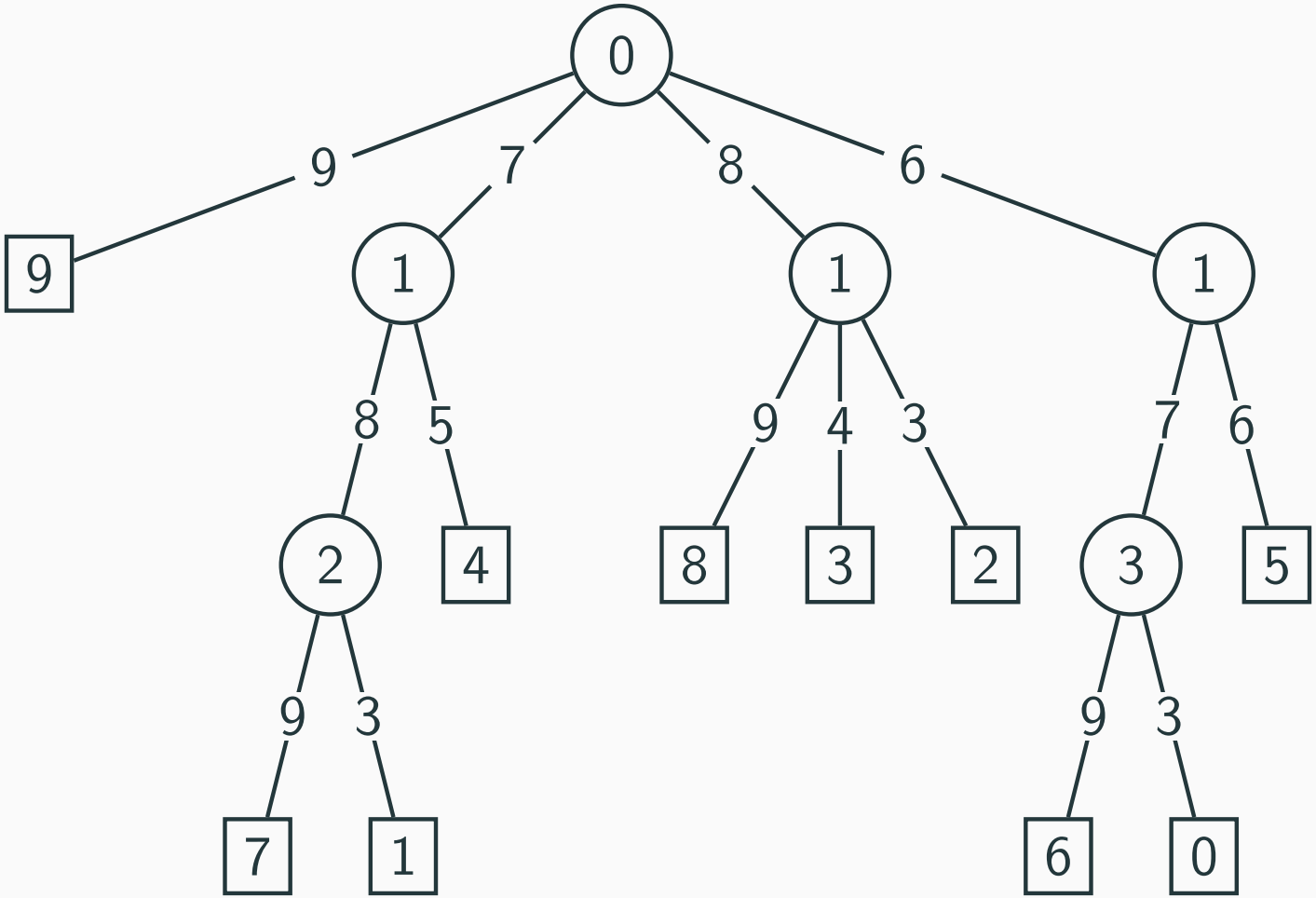
SA	9	7	1	4	8	3	2	6	0	5
LCP	0	0	2	1	0	1	1	0	3	1

Construction – Local Patricia Trie



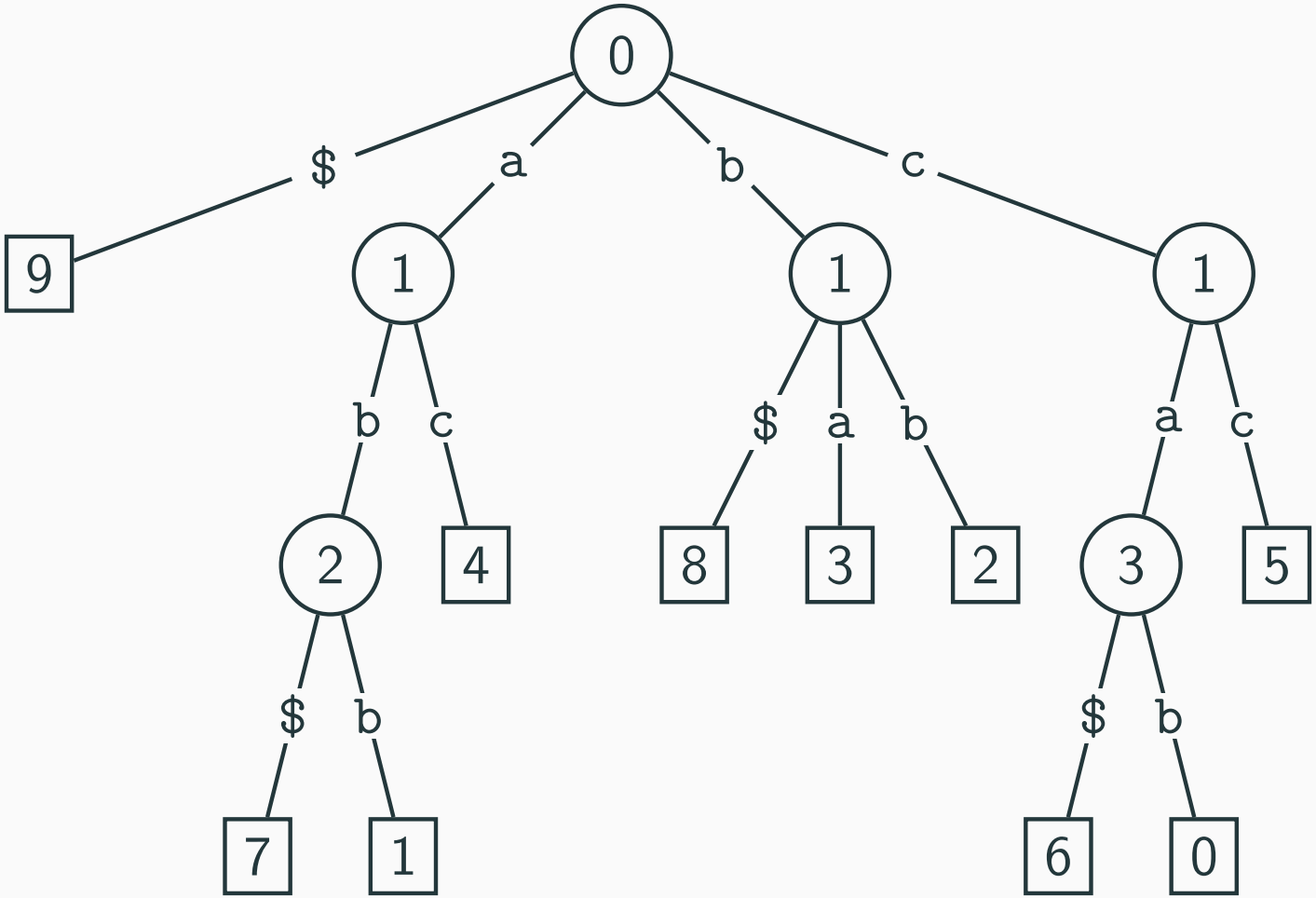
SA	9	7	1	4	8	3	2	6	0	5
LCP	0	0	2	1	0	1	1	0	3	1

Construction – Local Patricia Trie



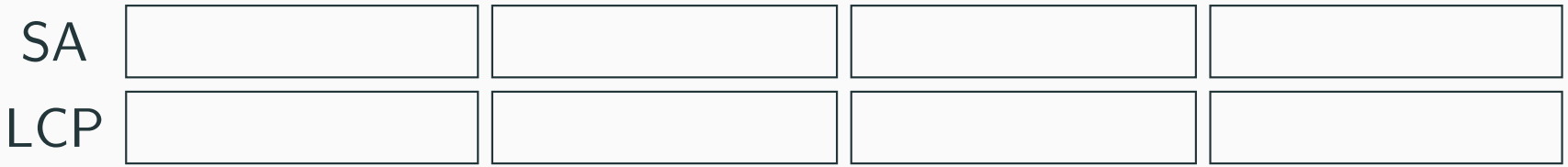
SA	9	7	1	4	8	3	2	6	0	5
LCP	0	0	2	1	0	1	1	0	3	1

Construction – Local Patricia Trie

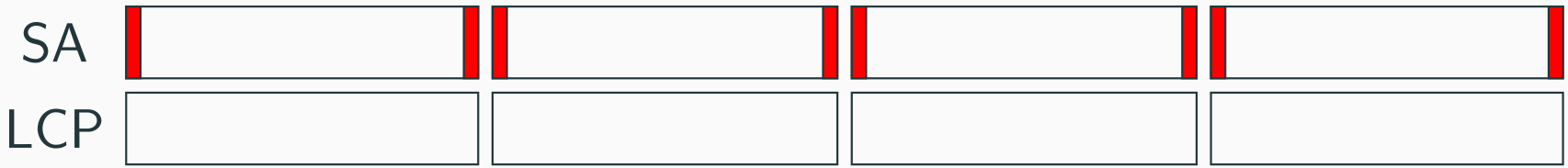


SA	9	7	1	4	8	3	2	6	0	5
LCP	0	0	2	1	0	1	1	0	3	1

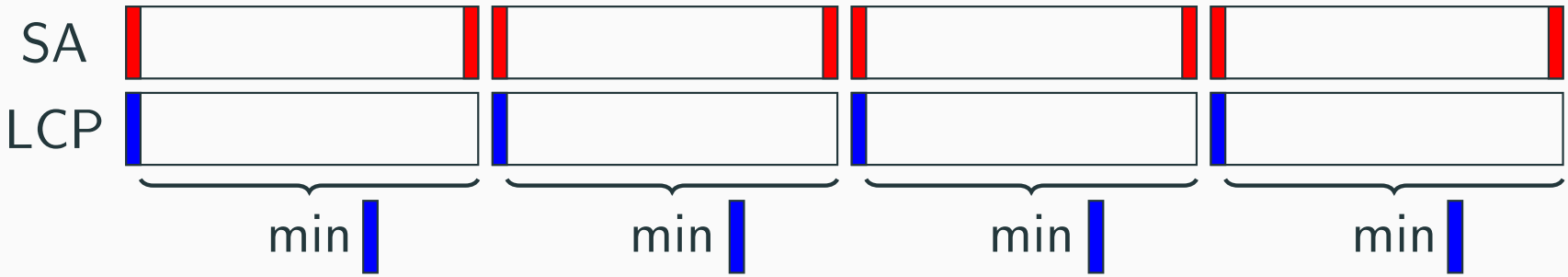
Construction – Global Trie



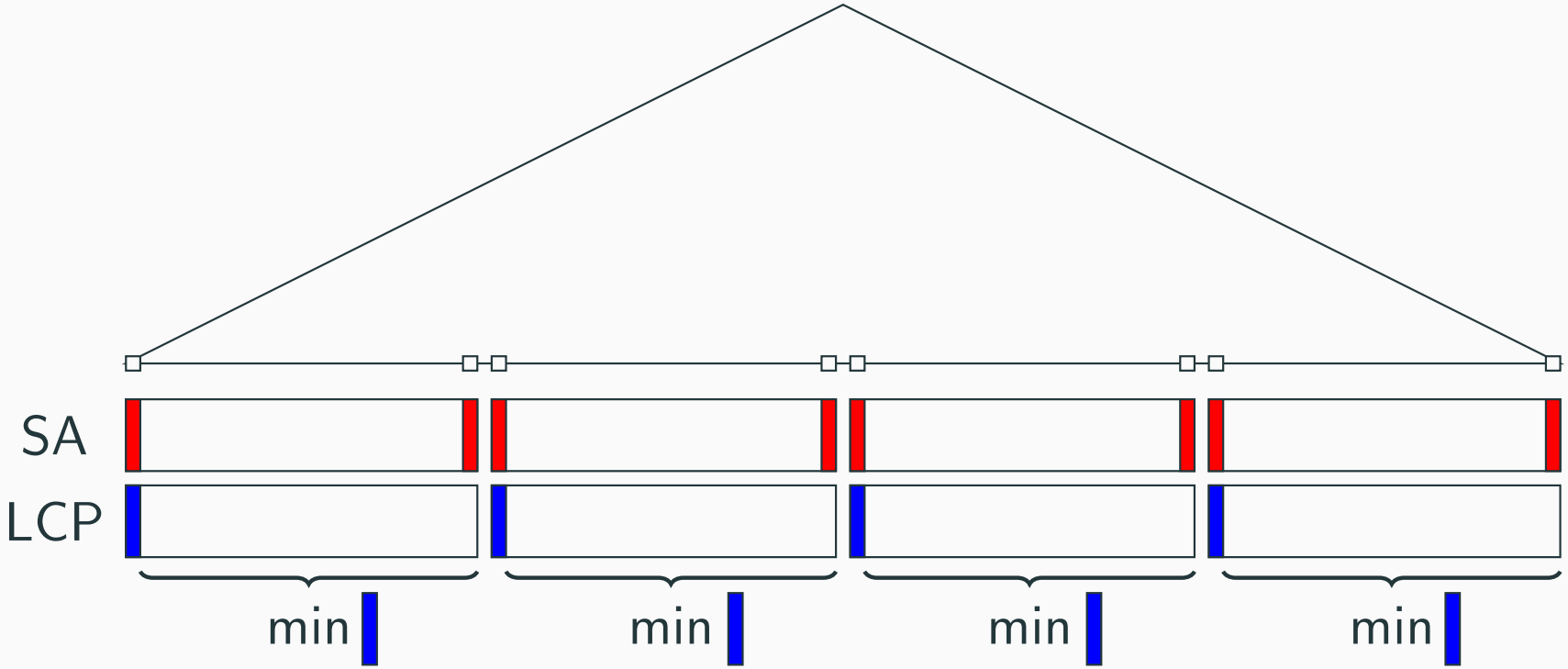
Construction – Global Trie



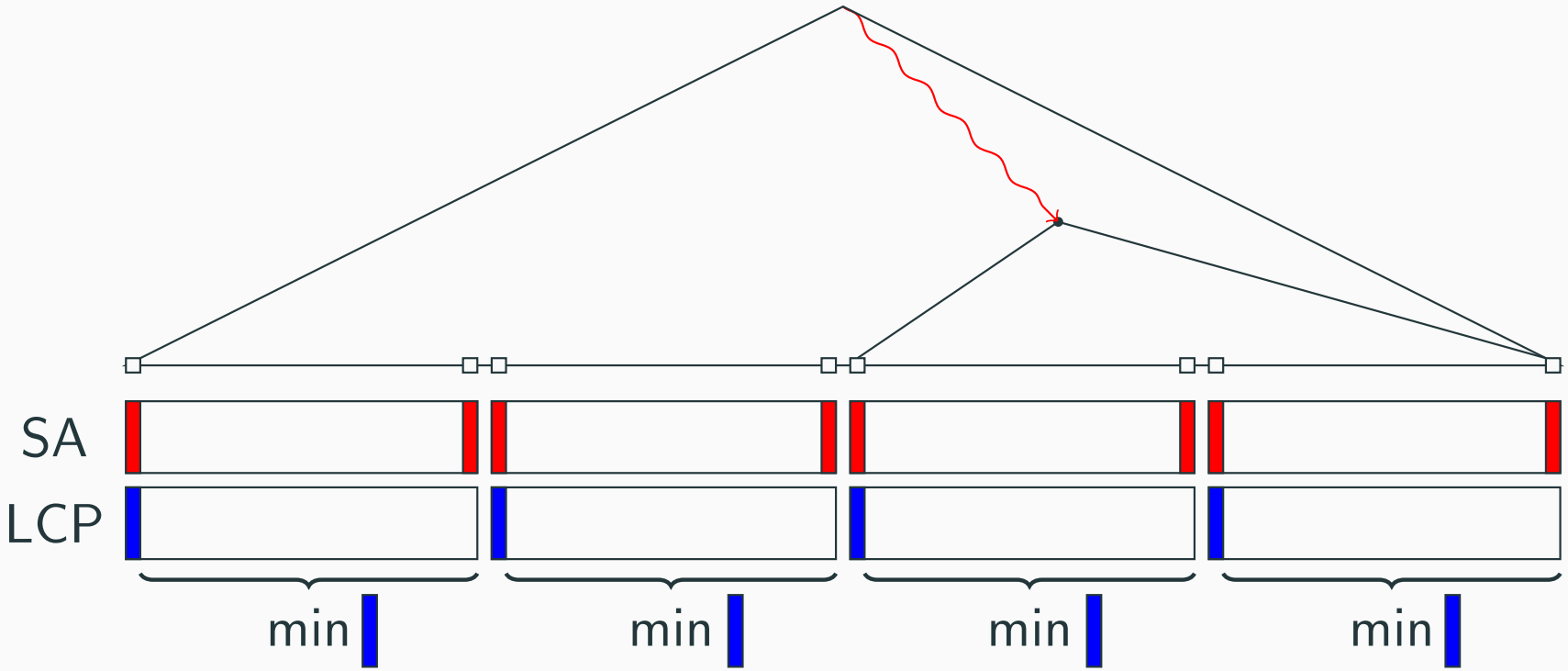
Construction – Global Trie



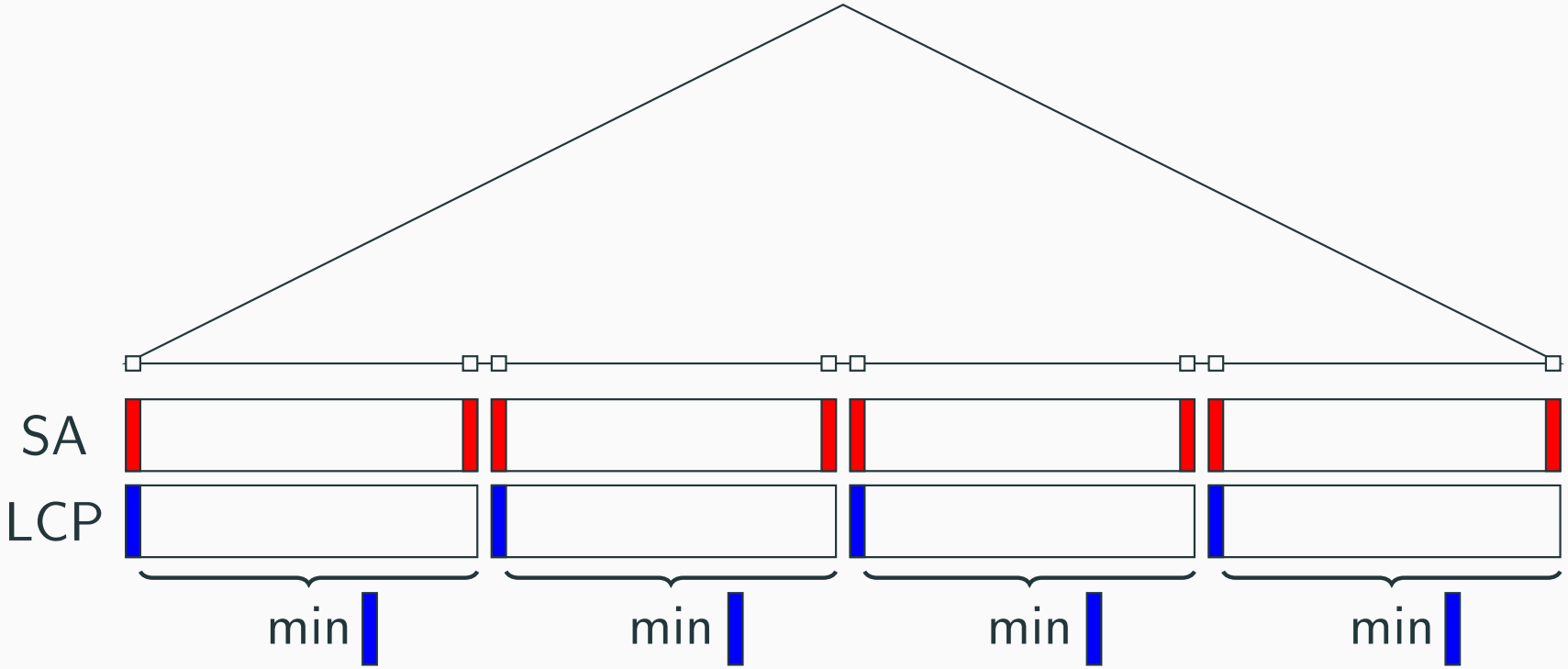
Construction – Global Trie



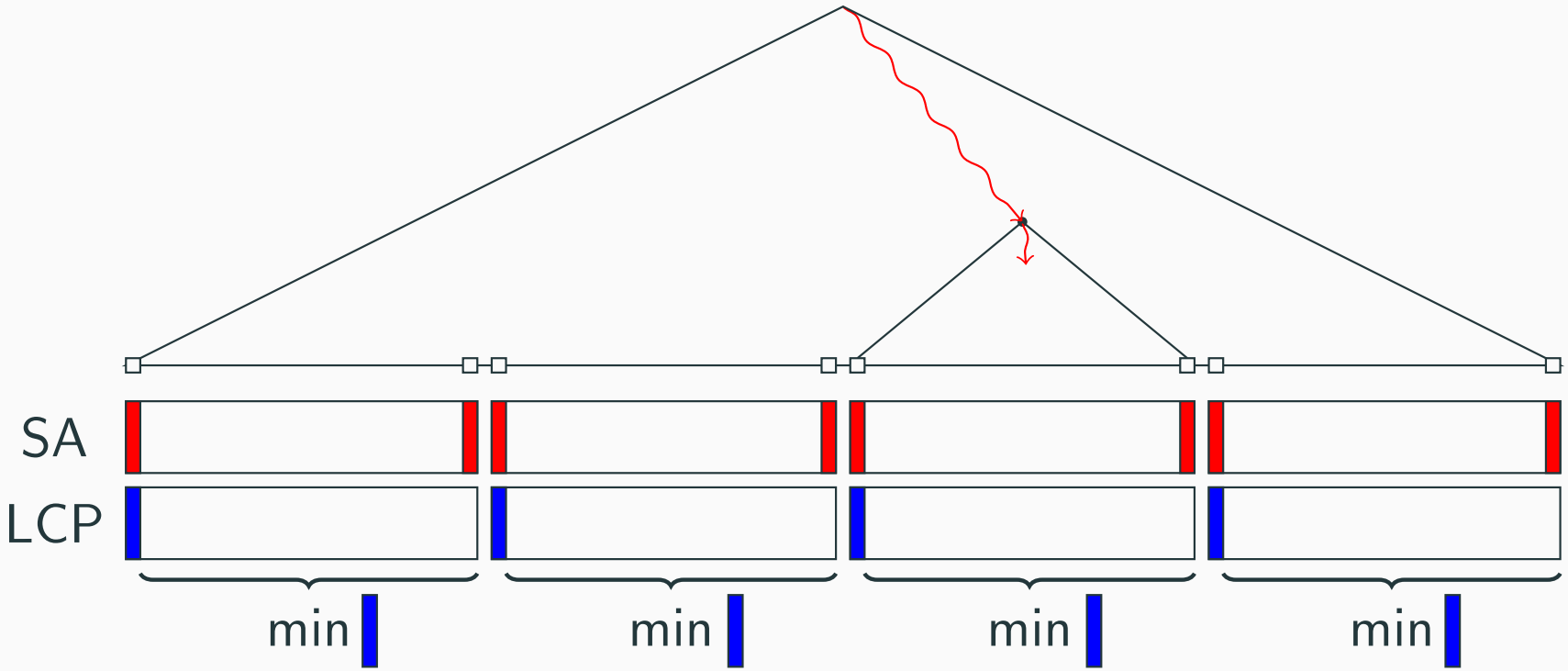
Construction – Global Trie



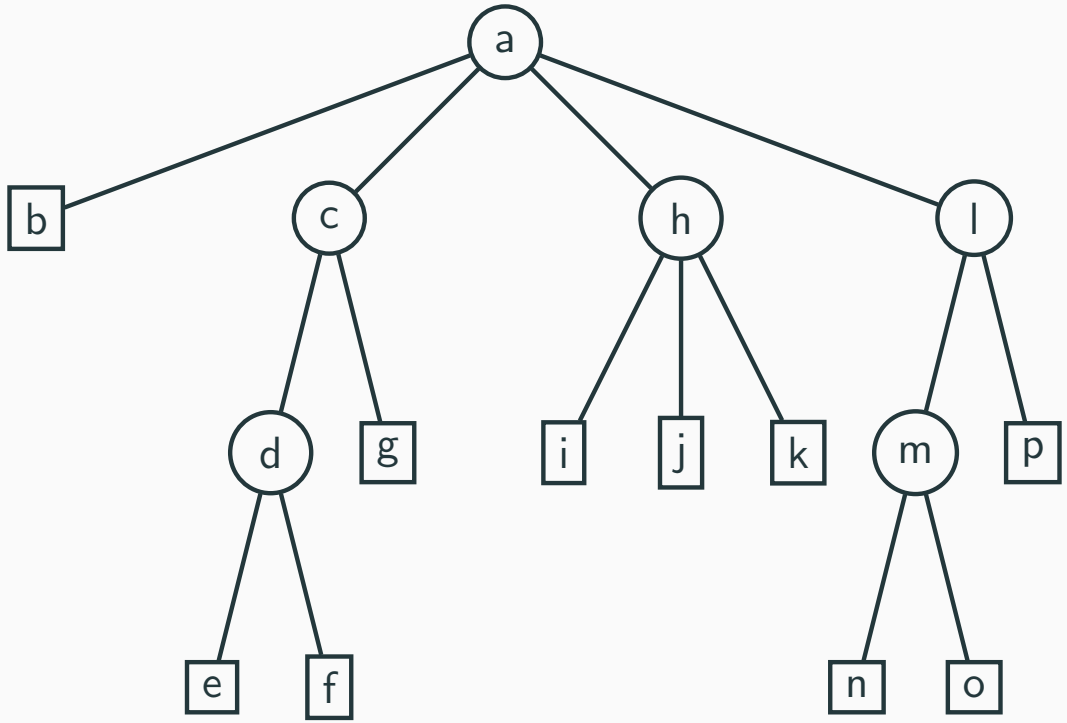
Construction – Global Trie



Construction – Global Trie



Succinct Trees



BP	((()(((()())())(()()())(((()())())()))
DFUDS	((((())(()(()))))(((())))(()(())))))
LOUDS	1111001101110110110000011000000

Experiments

Platform

- 32 nodes with 16 MPI processes
- 2 Octa-Core Intel Xeon E5-2670 processors
- 64 GB main memory

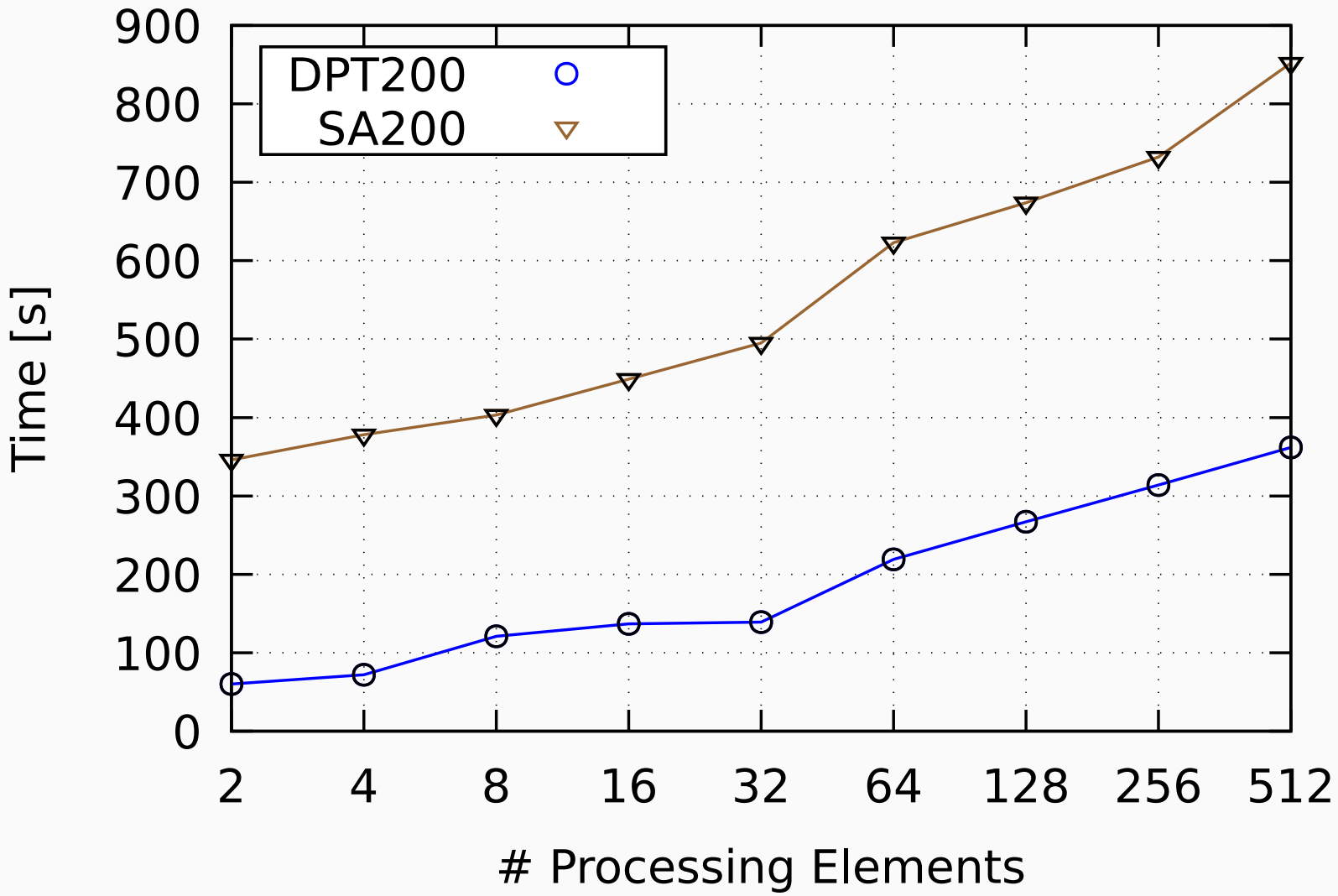
Data

- 100 GiB of the CommonCrawl Corpus
- 20 M AOL Queries

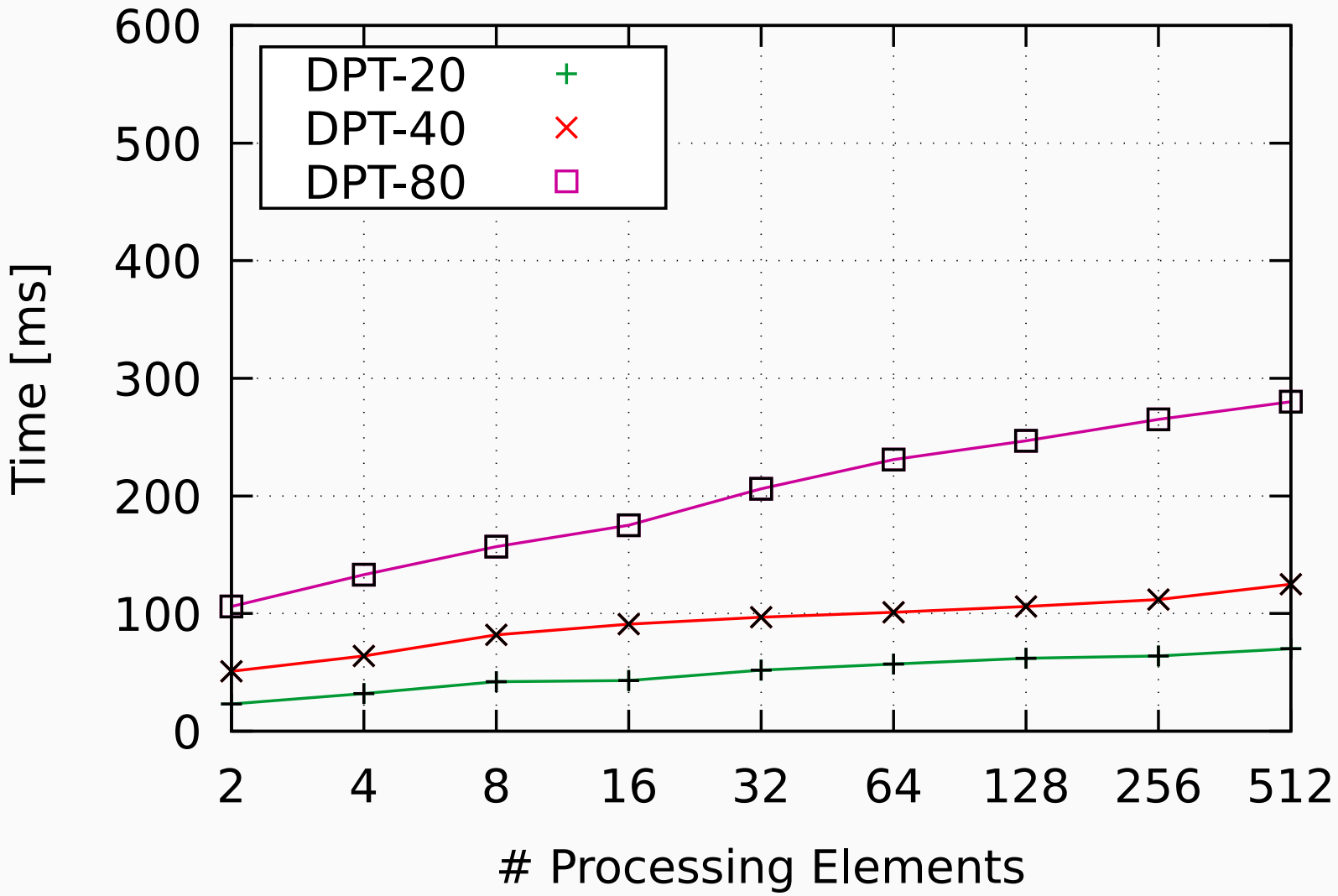
Competitor

- Distributed Suffix Array (Multiplexed) [Arroyuelo et al., 2014]

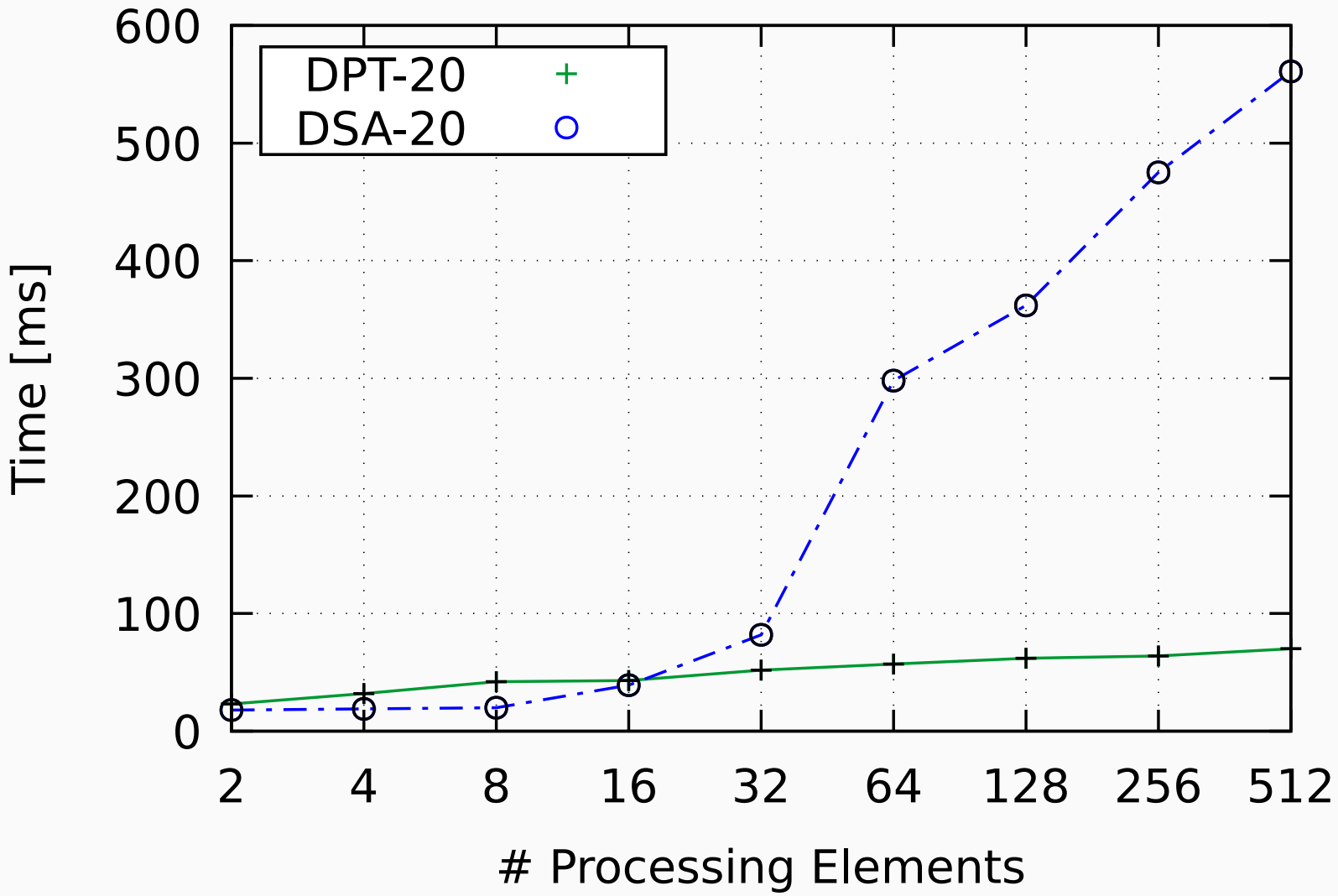
Experiments – Construction



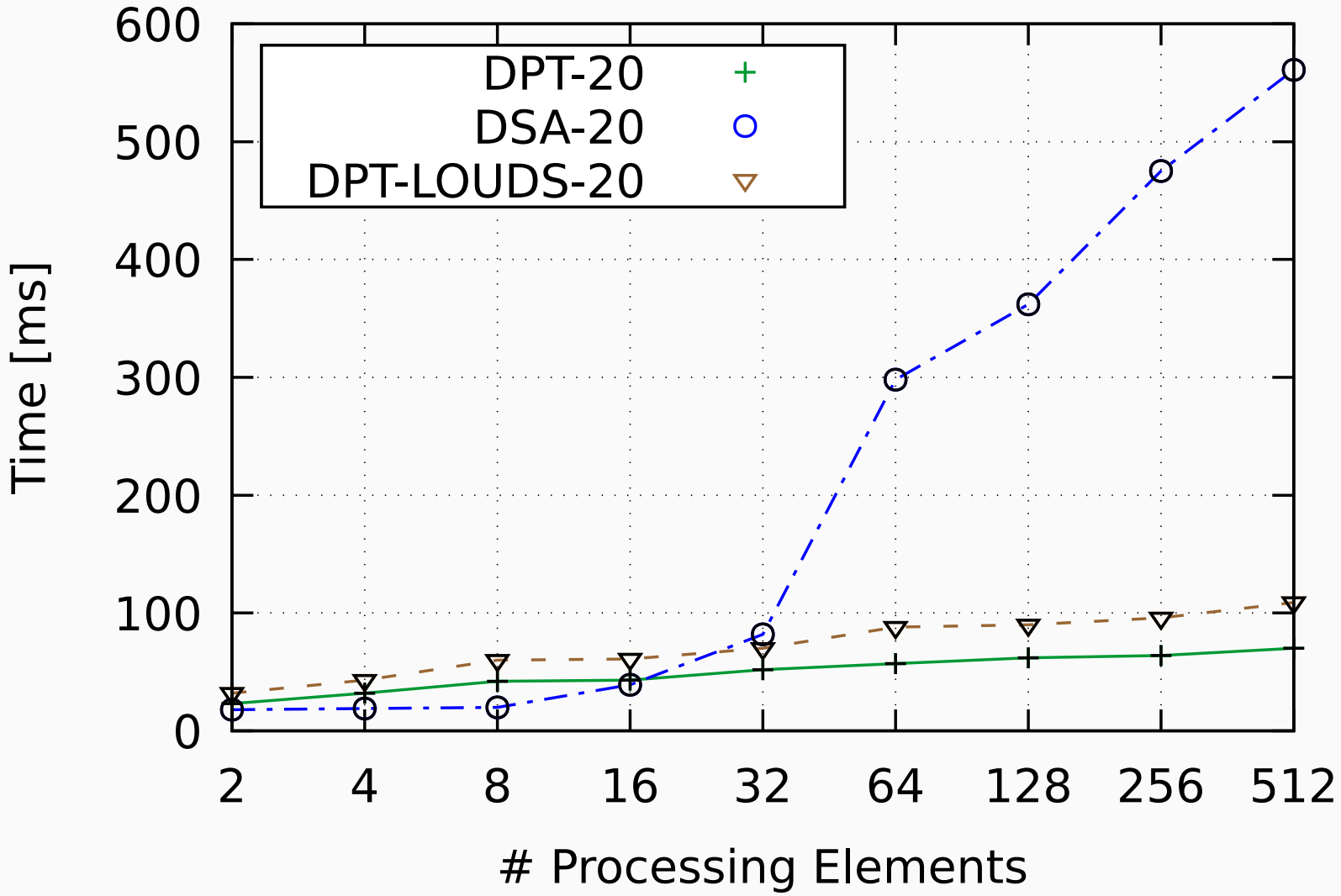
Experiments – Querying



Experiments – Querying



Experiments – Querying



Conclusion

Current Work

- Scaling distributed full-text index

Ideas for Future Work

- Better load balancing
- Asynchronous query processing
- Distributed SA and LCP construction

Conclusion

Current Work

- Scaling distributed full-text index

Ideas for Future Work

- Better load balancing
- Asynchronous query processing
- Distributed SA and LCP construction

Thank You