Linear-Time Suffix-Sorting

Proseminar Datenkompression

WS 16/17 – Clemens Damke

Übersicht

Übersicht

Problemstellung

Übersicht

Problemstellung

Lösungsansätze

Übersicht

Problemstellung

Lösungsansätze

GSACA

Übersicht

Problemstellung

Lösungsansätze

GSACA

Performance

Übersicht

Problemstellung

Lösungsansätze

GSACA

Performance

Rückblick

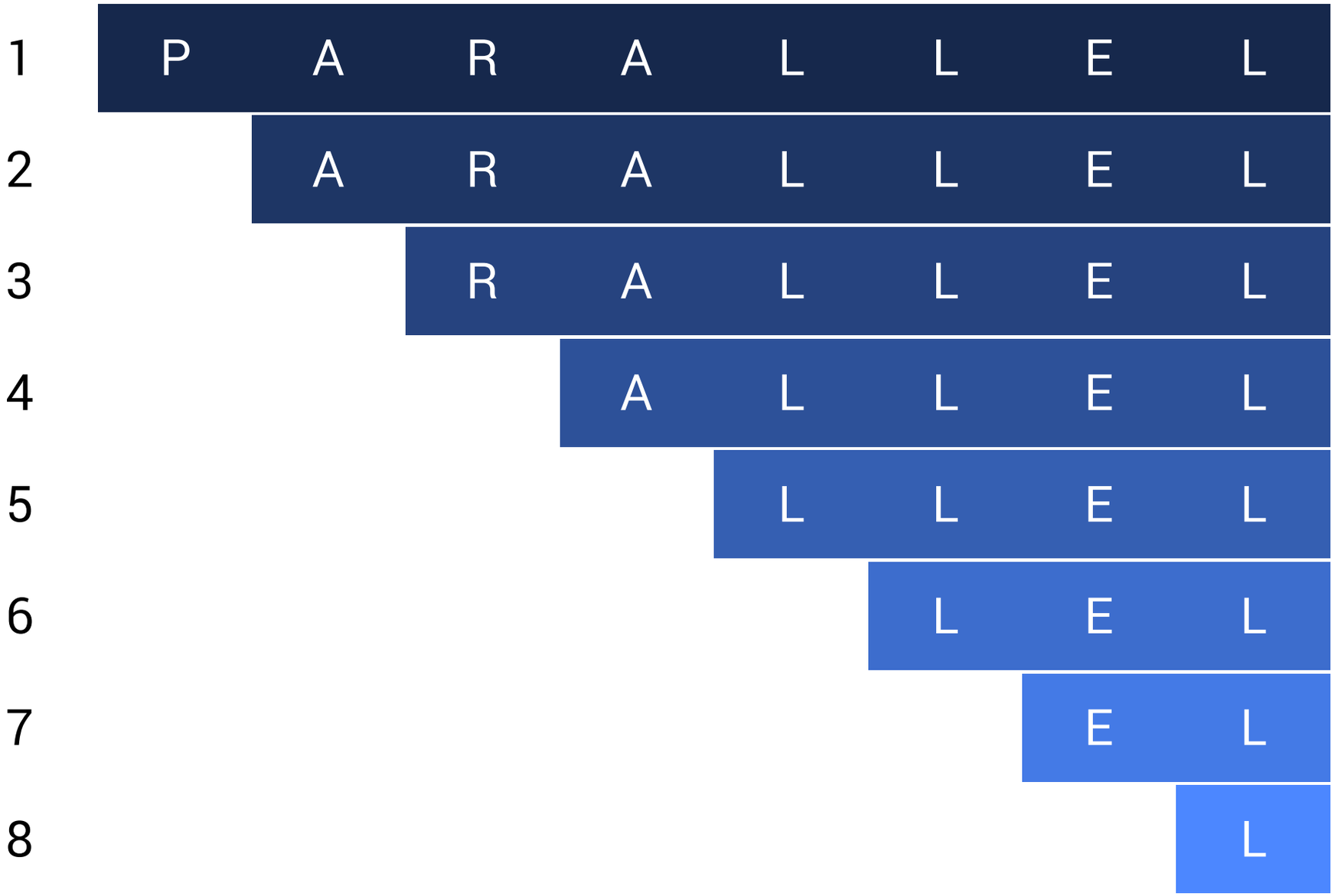
Problemstellung

Konstruktion eines **Suffix Arrays** mit einem **rekursionsfreien Linearzeit-Algorithmus**.

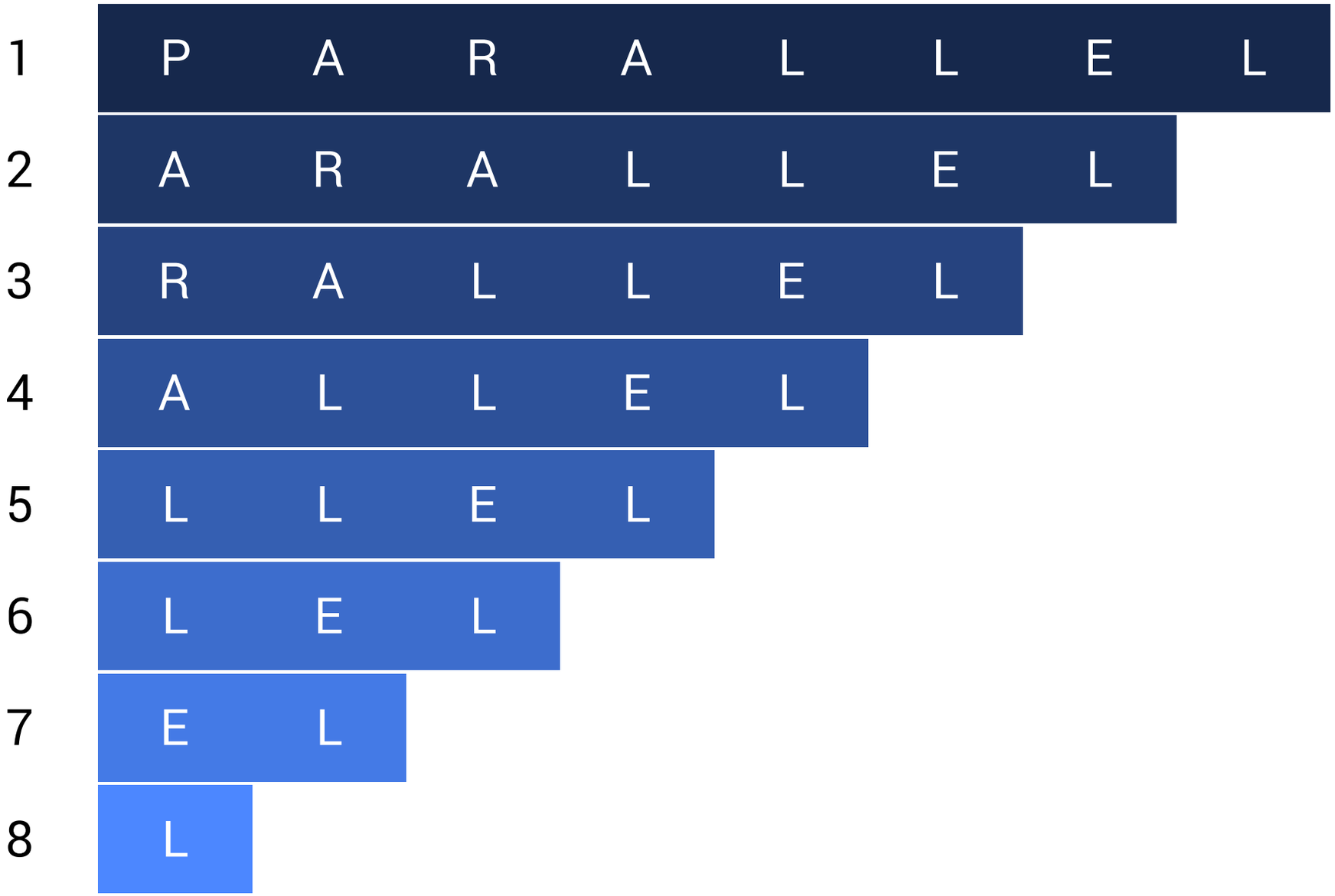
Konstruktion eines **Suffix Arrays** mit einem **rekursionsfreien Linearzeit-Algorithmus**.

Was ist ein Suffix Array?

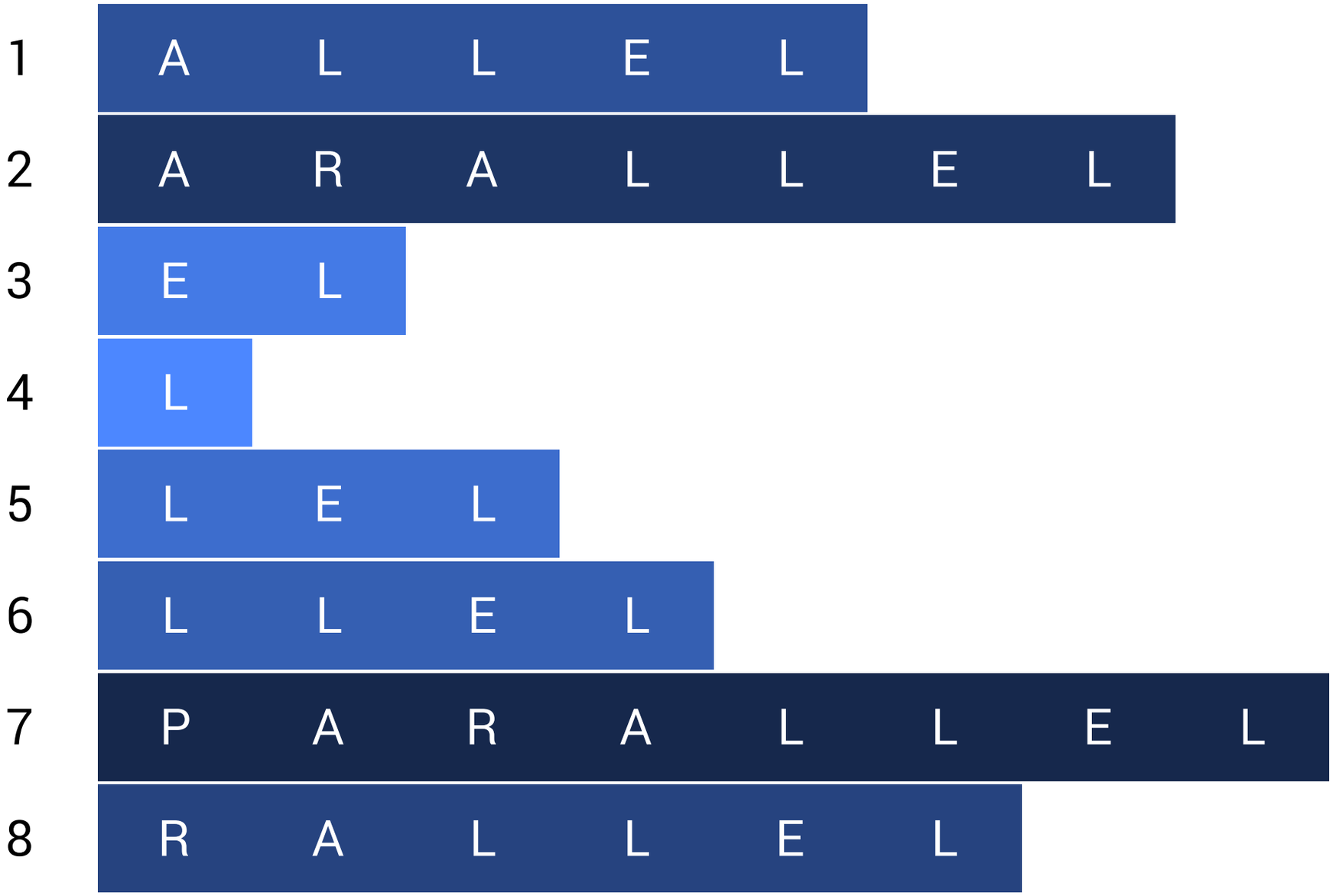
Was ist ein Suffix Array?



Was ist ein Suffix Array?



Was ist ein Suffix Array?



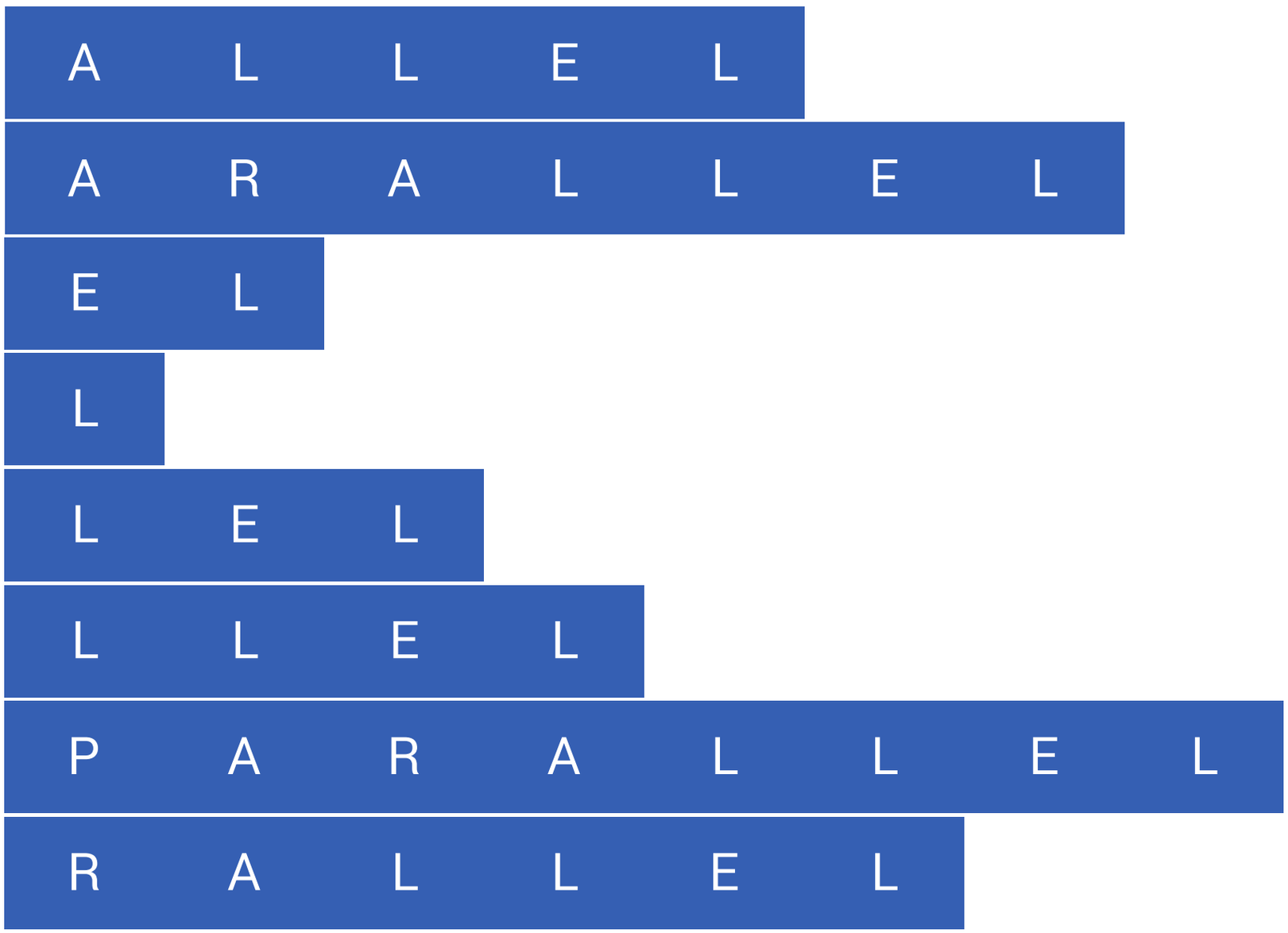
Was ist ein Suffix Array?

1	A L L E L	4
2	A R A L L E L	2
3	E L	7
4	L	8
5	L E L	6
6	L L E L	5
7	P A R A L L E L	1
8	R A L L E L	3

Einsatzgebiete

Substringsuche

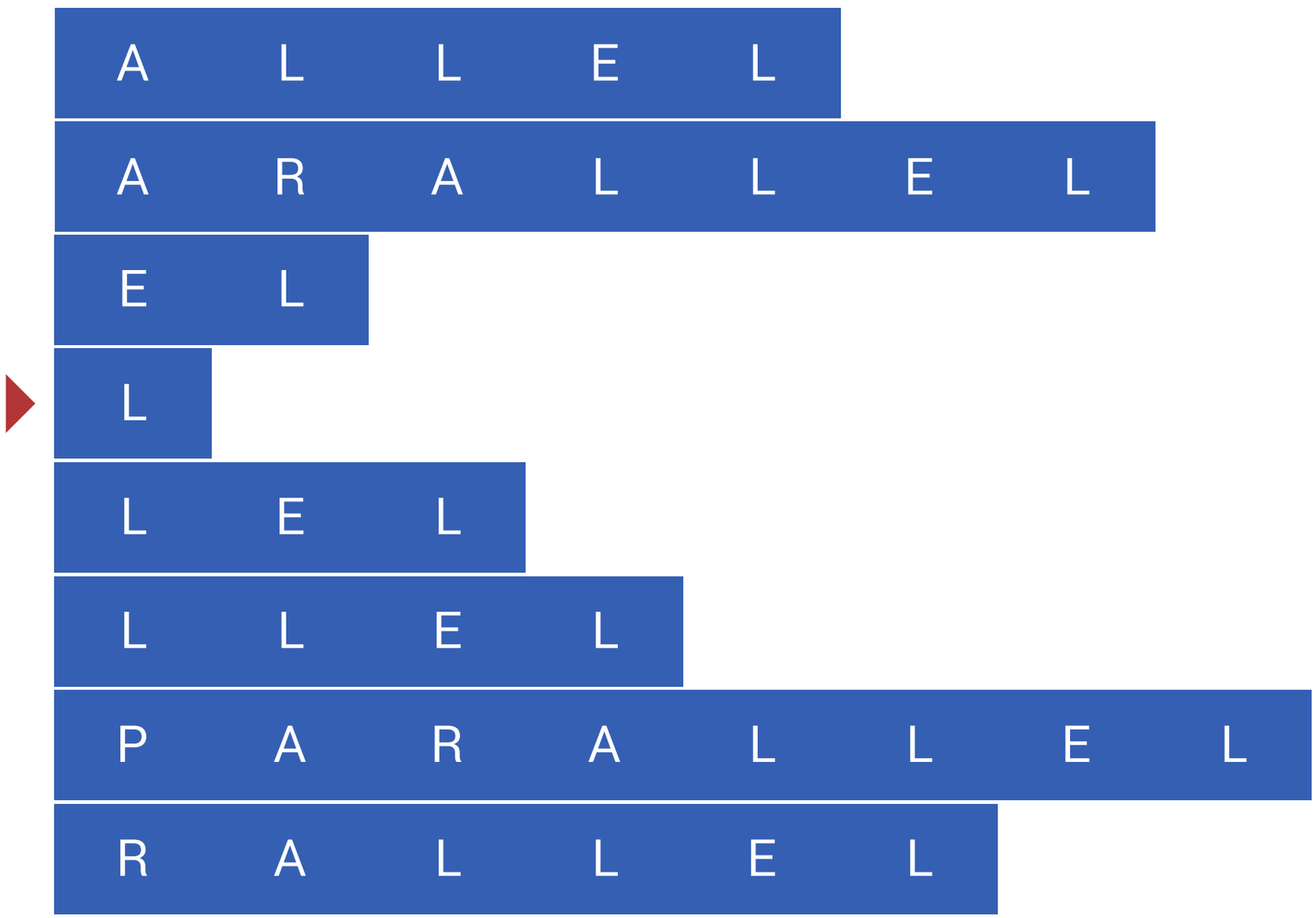
Ist *alle* in *parallel* enthalten?



Einsatzgebiete

Substringsuche

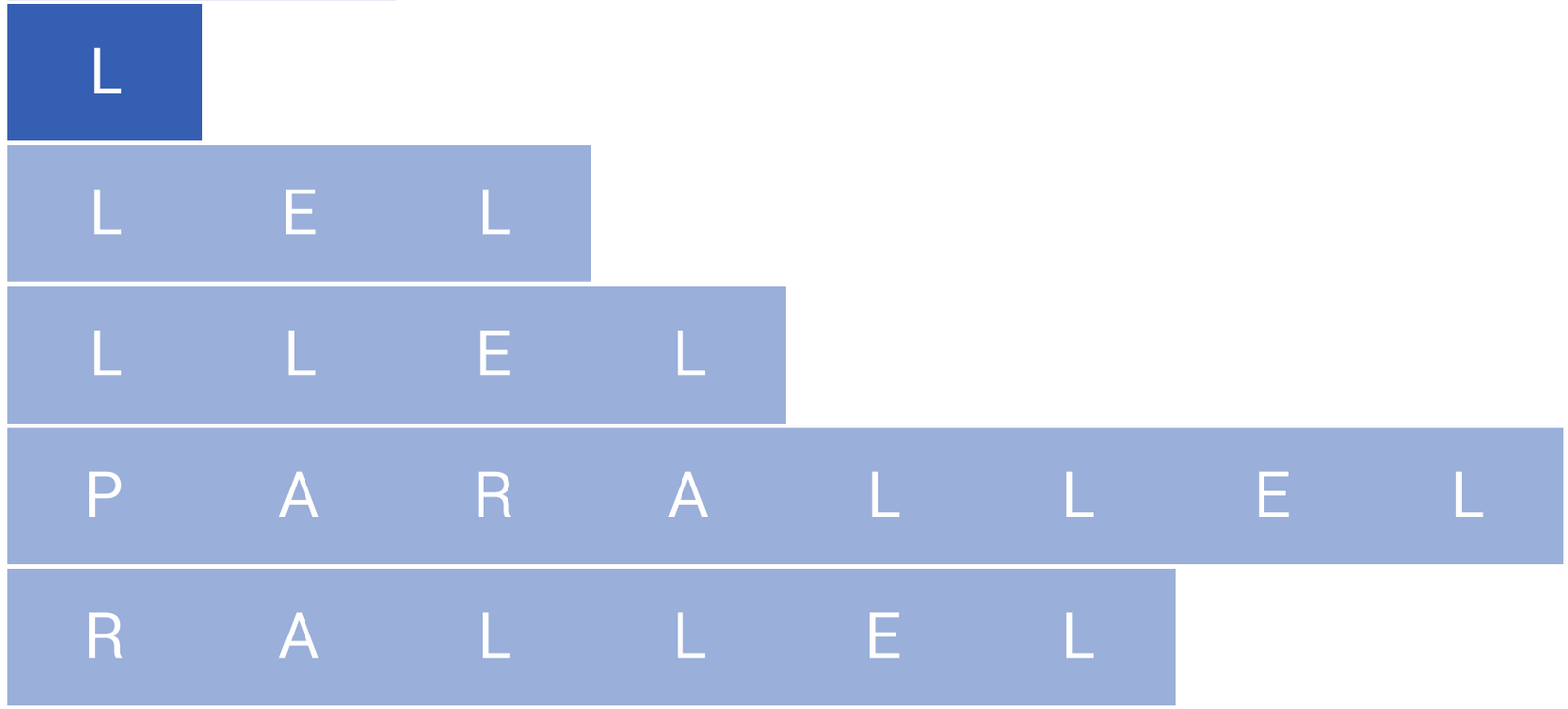
Ist *alle* in *parallel* enthalten?



Einsatzgebiete

Substringsuche

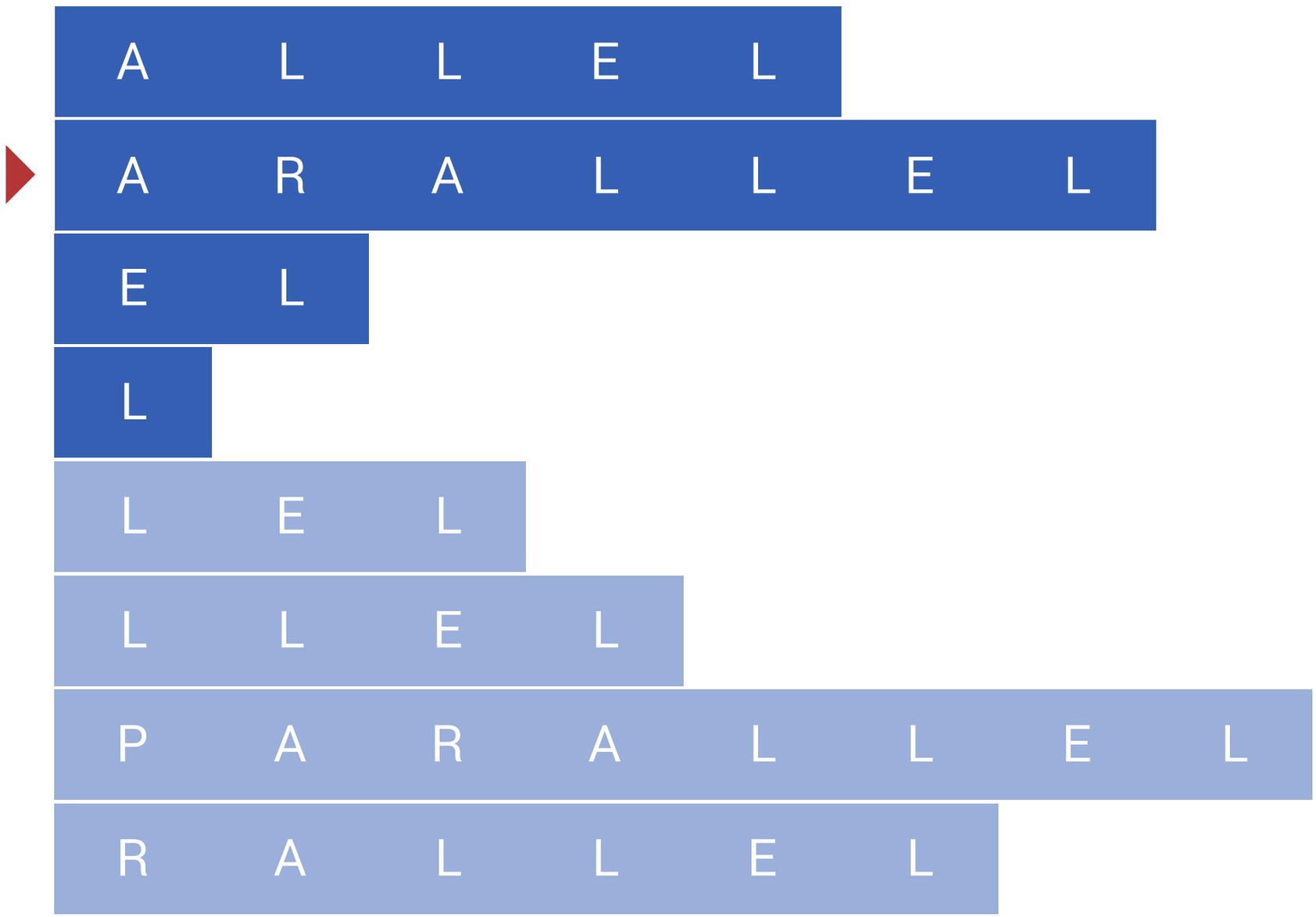
Ist *alle* in *parallel* enthalten?



Einsatzgebiete

Substringsuche

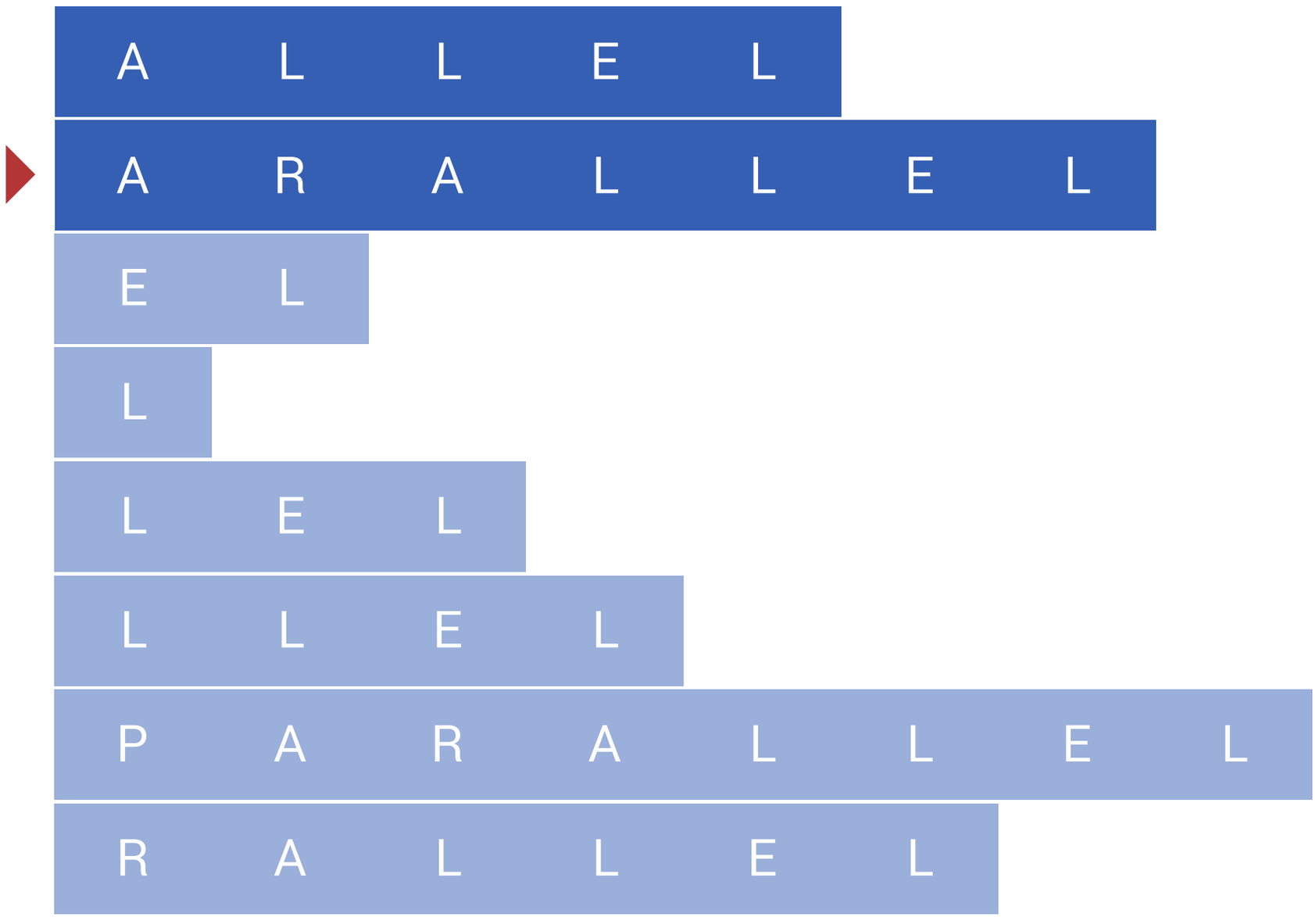
Ist *alle* in *parallel* enthalten?



Einsatzgebiete

Substringsuche

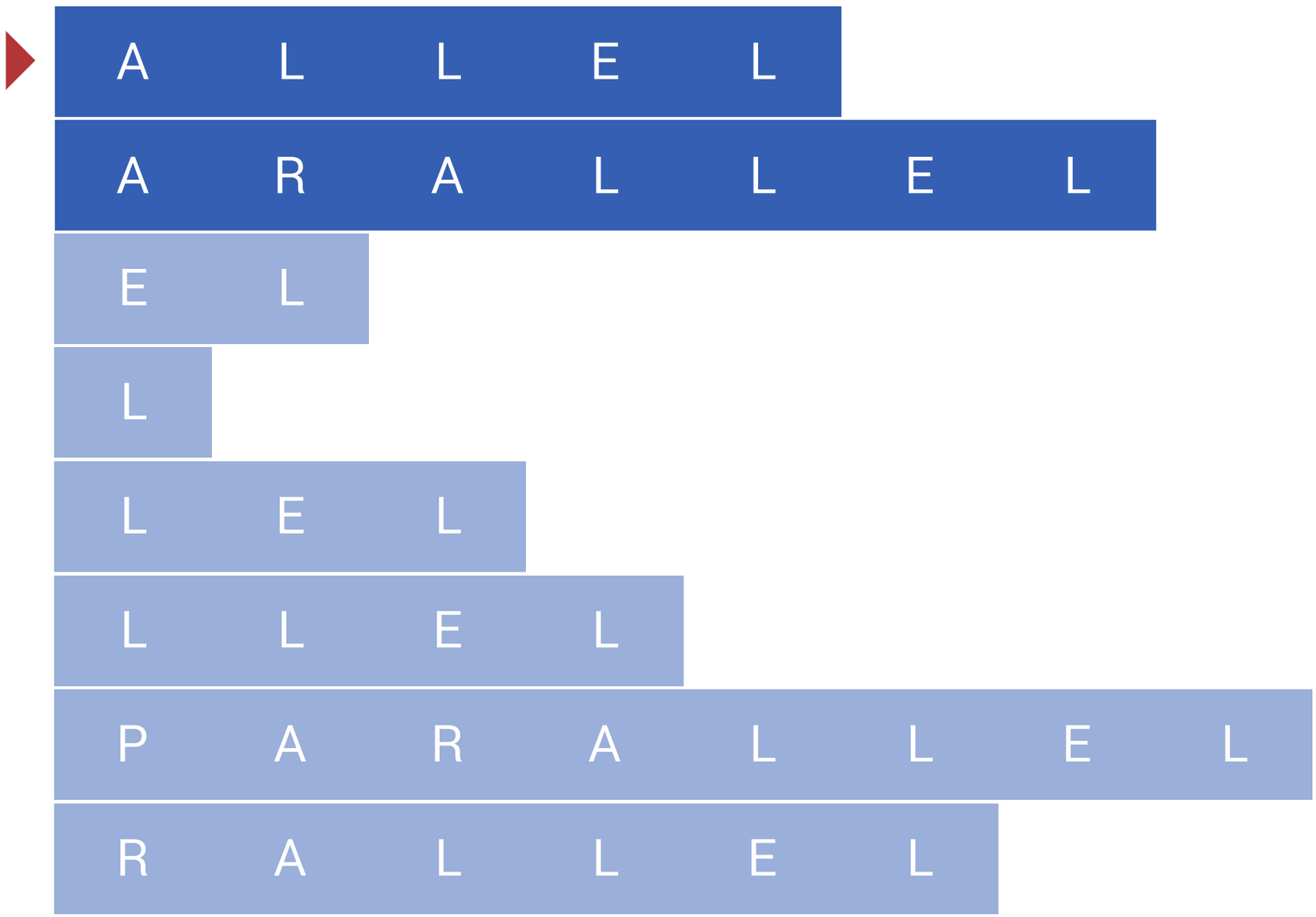
Ist *alle* in *parallel* enthalten?



Einsatzgebiete

Substringsuche

Ist *alle* in *parallel* enthalten?

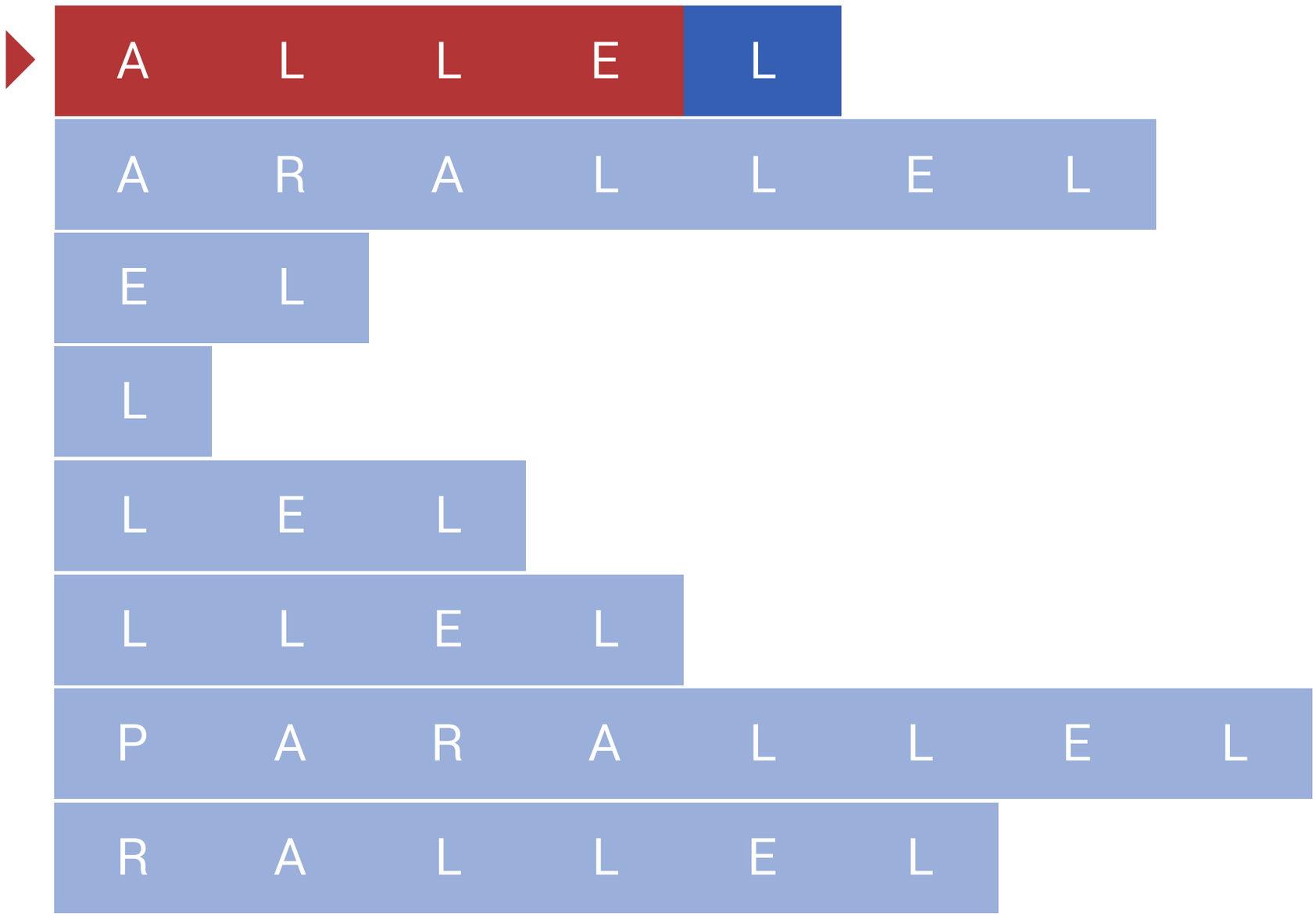


Einsatzgebiete

Substringsuche

Ist *alle* in *parallel* enthalten?

Ja, an Stelle 4.



Einsatzgebiete

Verwendet in Implementationen

des LZ77-Kompressionsalgorithmus

Konstruktion eines **Suffix Arrays** mit einem **rekursionsfreien Linearzeit-Algorithmus**.

Konstruktion eines **Suffix Arrays** mit

einem **rekursionsfreien Linearzeit-Algorithmus**

Lösungsansätze

Naiver Ansatz

Verwendung eines allgemeinen Sortierverfahrens

(z. B. Quicksort)

$$O(n \log n) \cdot O(n) = O(n^2 \log n)$$

Naiver Ansatz

Verwendung eines allgemeinen Sortierverfahrens

(z. B. Quicksort)

$$O(n \log n) \cdot O(n) = O(n^2 \log n) \neq O(n)$$

Linearzeit Ansätze

	Skew	SA-IS
Art	rekursiv	rekursiv
Zeit	$O(n)$	$O(n)$
Speicher	$O(\log n) + \max 24n$	$O(\log n) + \max 2n$

Linearzeit Ansätze

	Skew	SA-IS	?
Art	rekursiv	rekursiv	iterativ
Zeit	$O(n)$	$O(n)$	$O(n)$
Speicher	$O(\log n) + \max 24n$	$O(\log n) + \max 2n$	$O(1) + ?$

?

iterativ

$O(n)$

$O(1) + ?$

GSACA

iterativ

$O(n)$

$O(1) + ?$

GSACA

Greedy **S**uffix **A**rray **C**onstruction **A**lgorithm

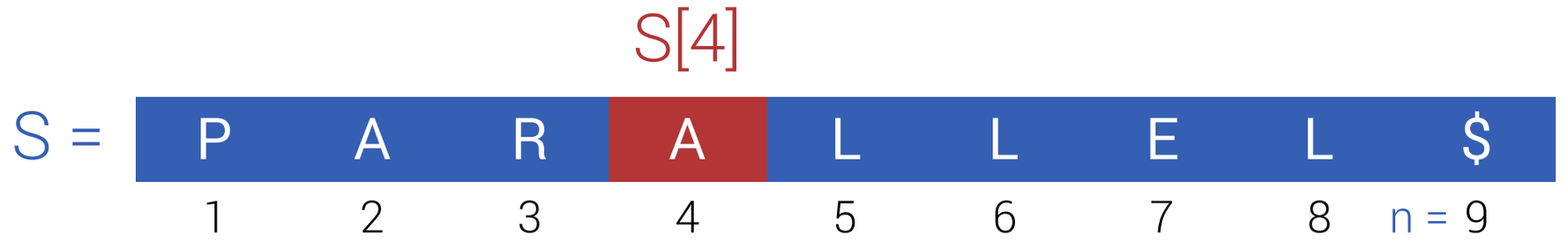
Definitionen

P	A	R	A	L	L	E	L	$
1	2	3	4	5	6	7	8	9

Definitionen

S := Eingabe, eine mit **$** terminierte Zeichenkette der Länge **n**

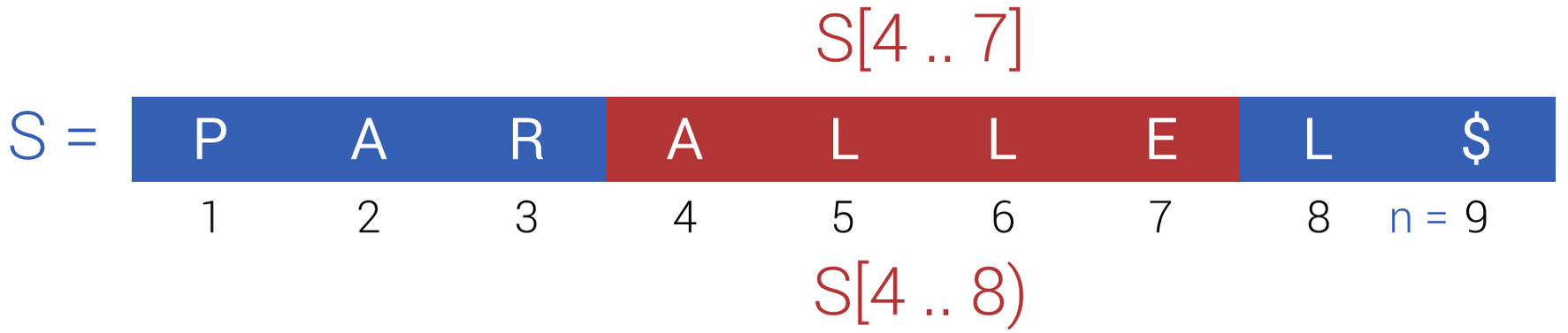
Definitionen



S := Eingabe, eine mit **$** terminierte Zeichenkette der Länge **n**

S[**i**] := **i**-tes Zeichen von **S**

Definitionen



S := Eingabe, eine mit **$** terminierte Zeichenkette der Länge **n**

S[i] := **i**-tes Zeichen von **S**

S[i .. j + 1) := **S[i .. j]** := **S[i] ... S[j]**

Definitionen

S := Eingabe, eine mit **$** terminierte Zeichenkette der Länge **n**

S[i] := **i**-tes Zeichen von **S**

S[i .. j + 1) := **S[i .. j]** := **S[i] ... S[j]**

S_i := **S[i .. n]**

Definitionen

S := Eingabe, eine mit **$** terminierte Zeichenkette der Länge **n**

S[i] := **i**-tes Zeichen von **S**

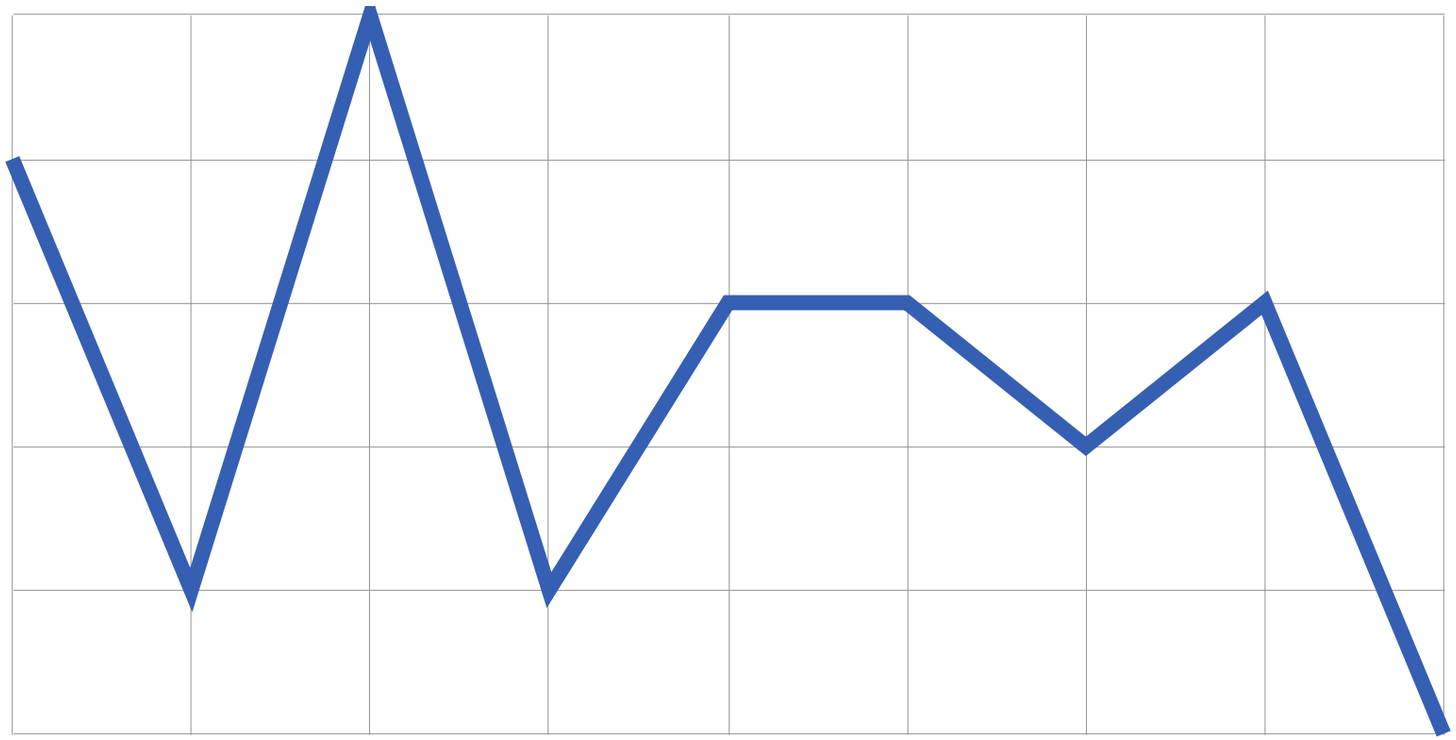
S[i .. j + 1) := **S[i .. j]** := **S[i]** ... **S[j]**

S_i := **S[i .. n]**

$\hat{\imath}$:= *min* { **j** ∈ **[i .. n]**: $S_j <_{lex} S_i$ }

Definitionen

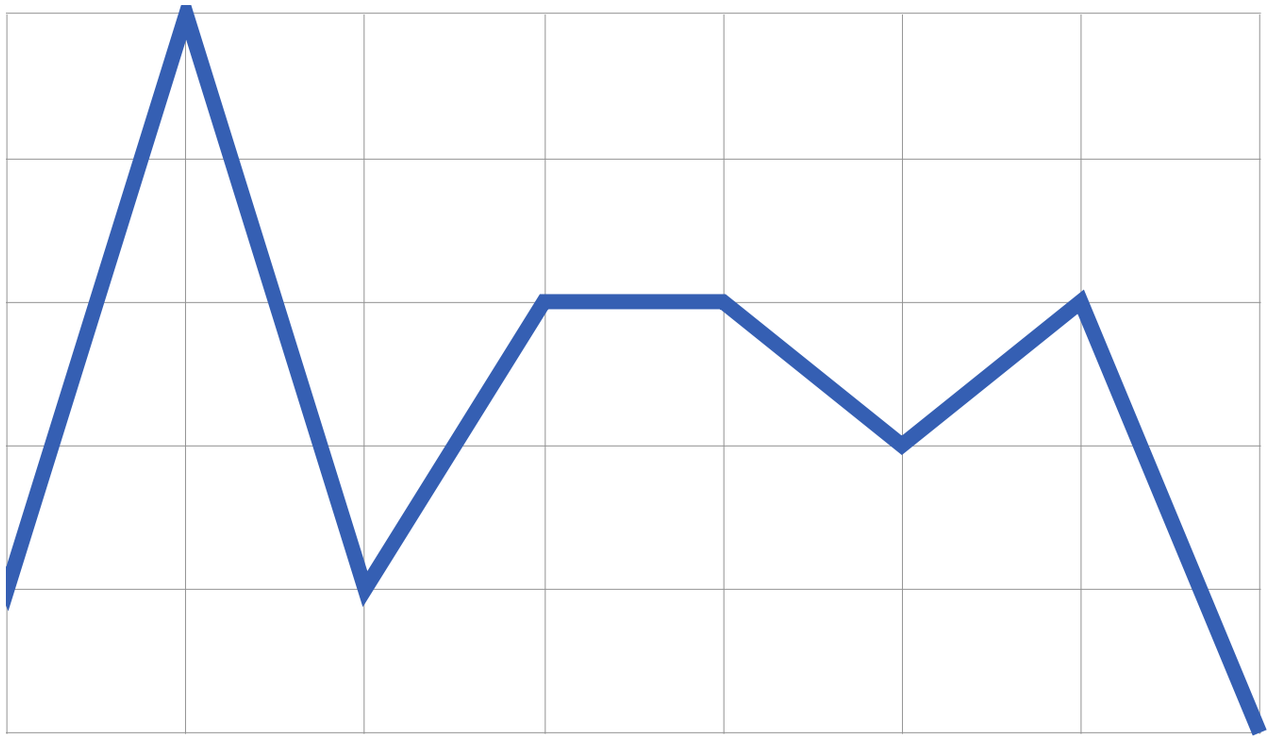
$\hat{\imath} := \mathit{min}\ \{\ j \in [i\ ..\ n]: S_j <_{lex} S_i\ \}$



P	A	R	A	L	L	E	L	$
1	2	3	4	5	6	7	8	9

Definitionen

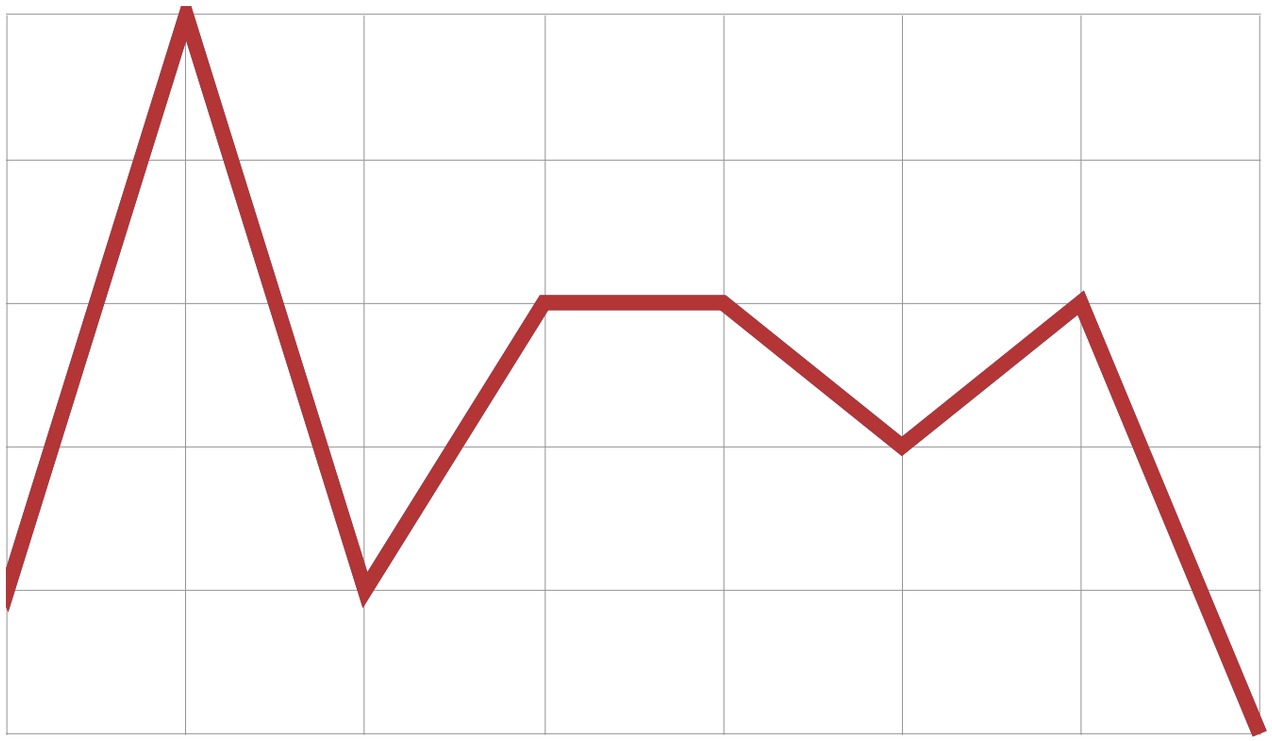
$\hat{\imath} := min\ \{\ j \in [i\ ..\ n]: S_j <_{lex} S_i\ \}$



S_i =	A	R	A	L	L	E	L	$
i =	2	3	4	5	6	7	8	9

Definitionen

$\hat{\imath} := min\ \{\ j \in [i\ ..\ n]: S_j <_{lex} S_i\ \}$

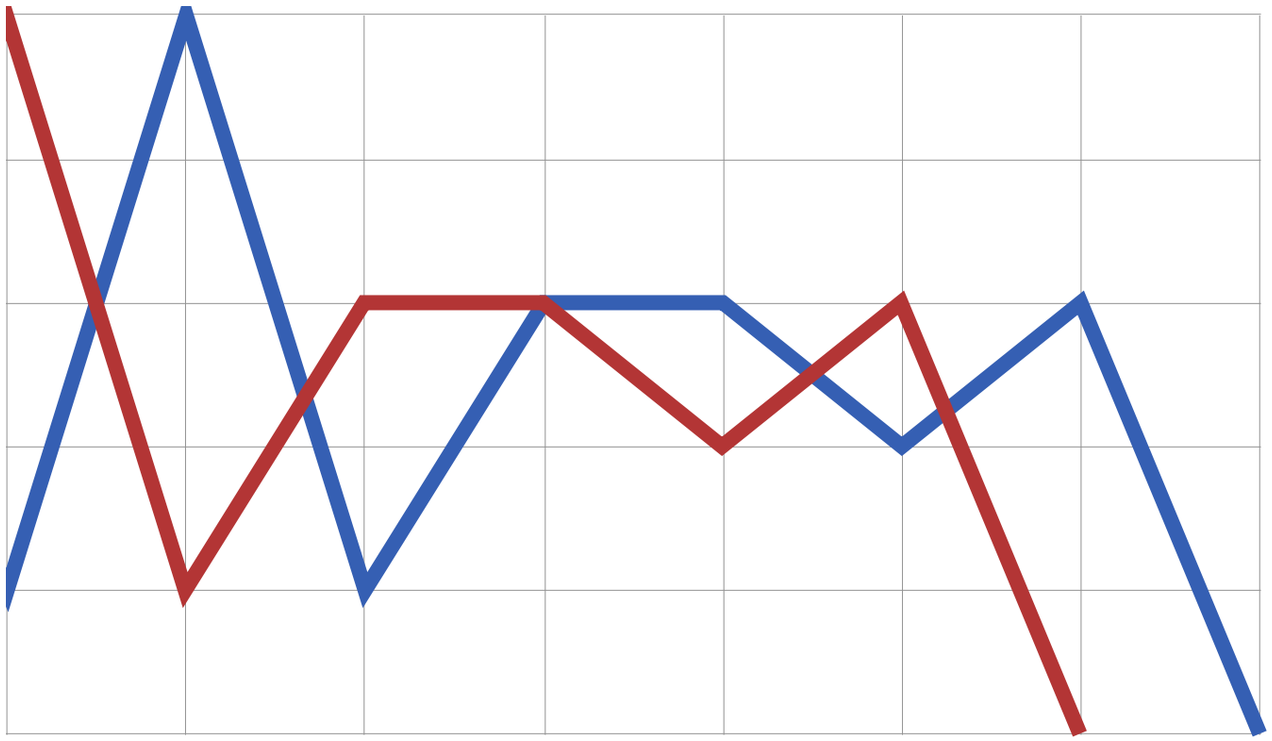


S_i =	A	R	A	L	L	E	L	$
i =	2	3	4	5	6	7	8	9

S_j =	A	R	A	L	L	E	L	$
j =	2	3	4	5	6	7	8	9

Definitionen

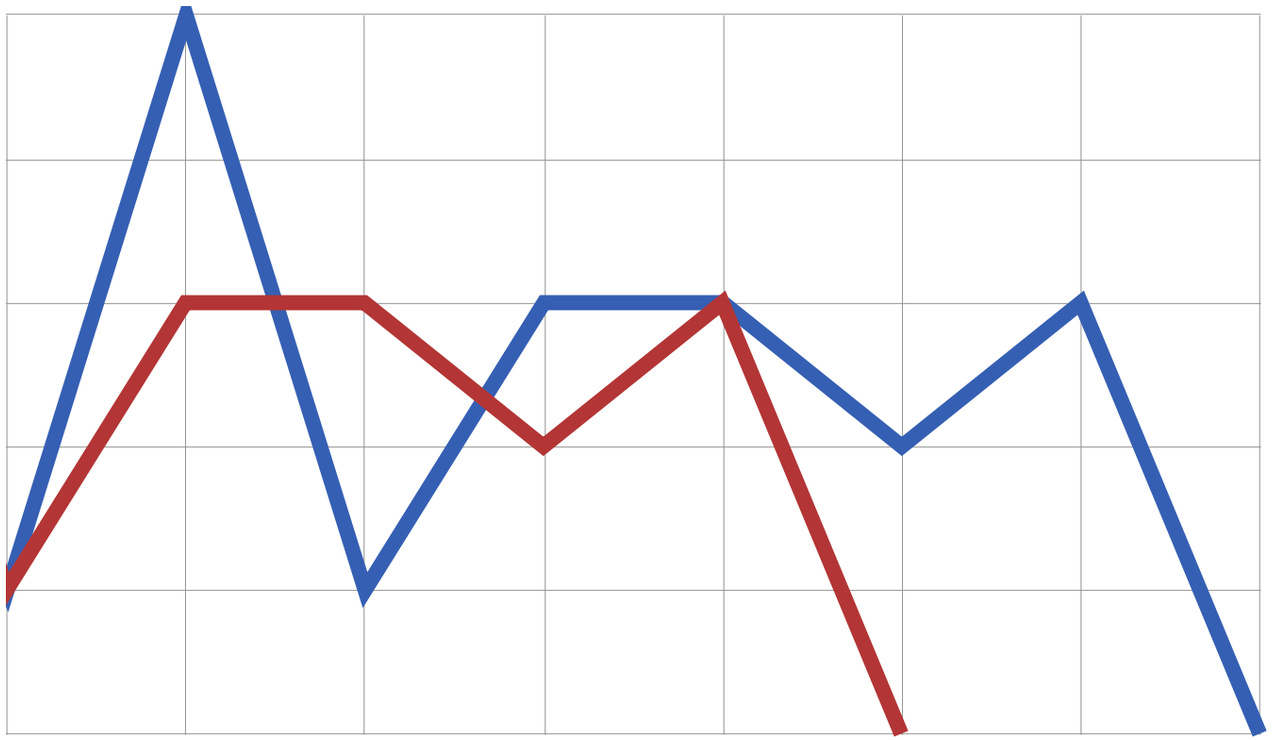
$\hat{\imath} := min\ \{\ j \in [i\ ..\ n]: S_j <_{lex} S_i\ \}$



S_i =	A	R	A	L	L	E	L	$
i =	2	3	4	5	6	7	8	9
S_j =	R	A	L	L	E	L	$	
j =	3	4	5	6	7	8	9	

Definitionen

$\hat{\imath} := min\ \{\ j \in [i\ ..\ n]: S_j <_{lex} S_i\ \}$

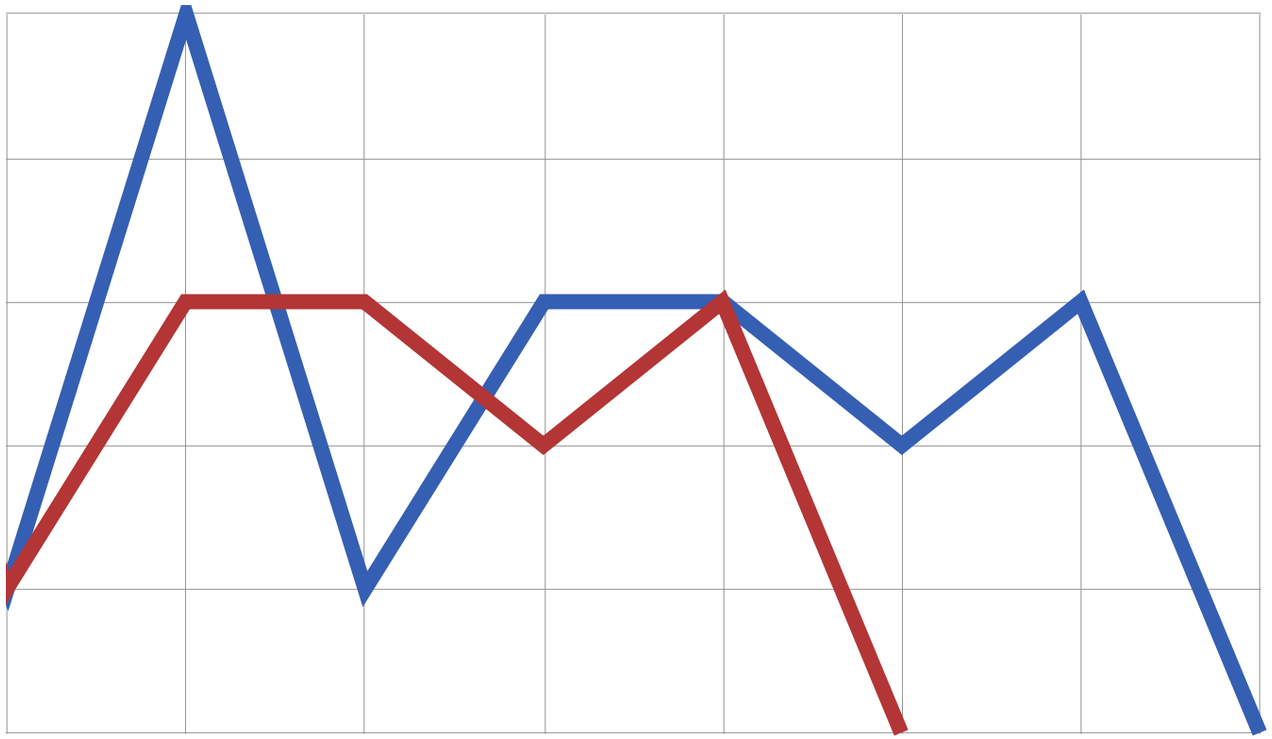


S_i =	A	R	A	L	L	E	L	$
i =	2	3	4	5	6	7	8	9

S_j =	A	L	L	E	L	$
j =	4	5	6	7	8	9

Definitionen

$$\hat{\imath} := \mathit{min}\ \{\ j \in [i\ ..\ n]\colon S_j <_{lex} S_i\ \}$$



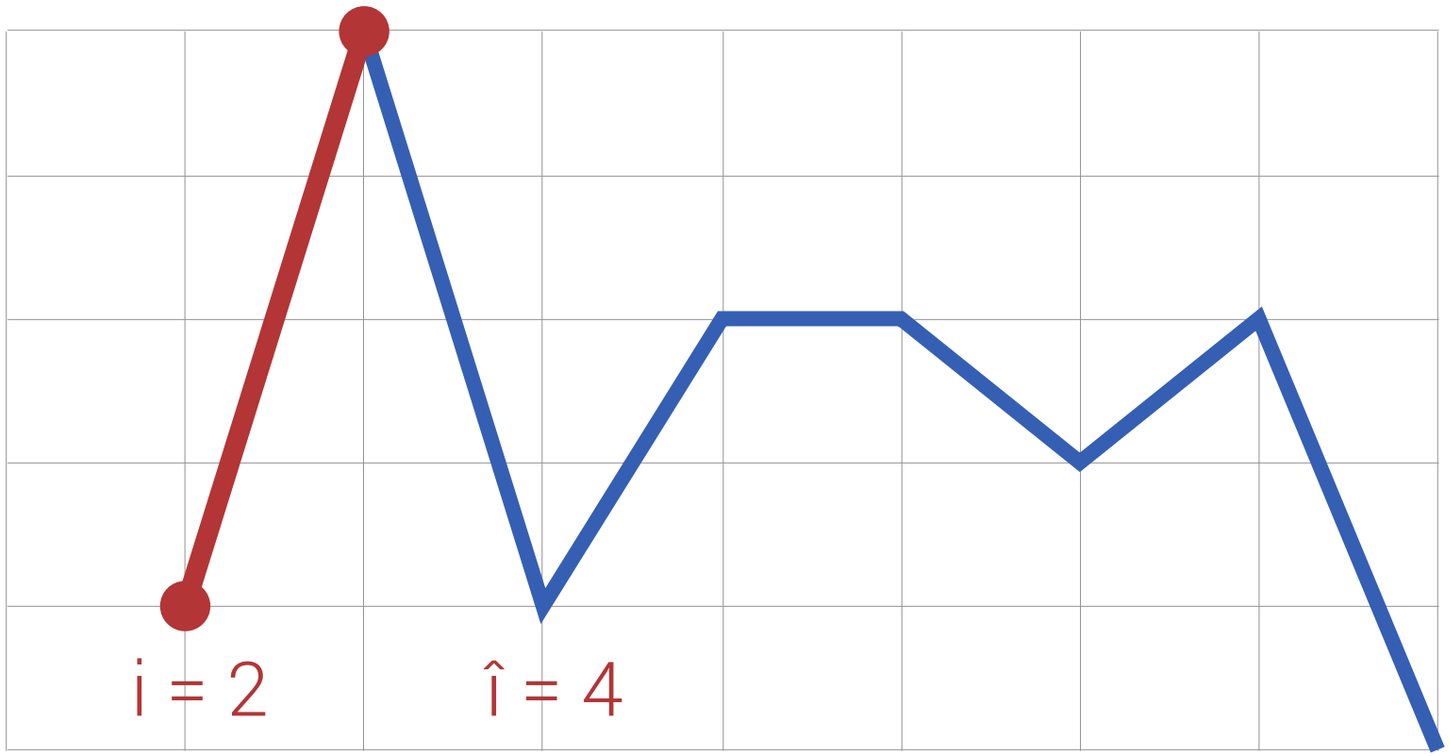
S_i =	A	R	A	L	L	E	L	$
i =	2	3	4	5	6	7	8	9

S_j =	A	L	L	E	L	$
j =	4	5	6	7	8	9

$\hat{\imath} = 4$

Definitionen

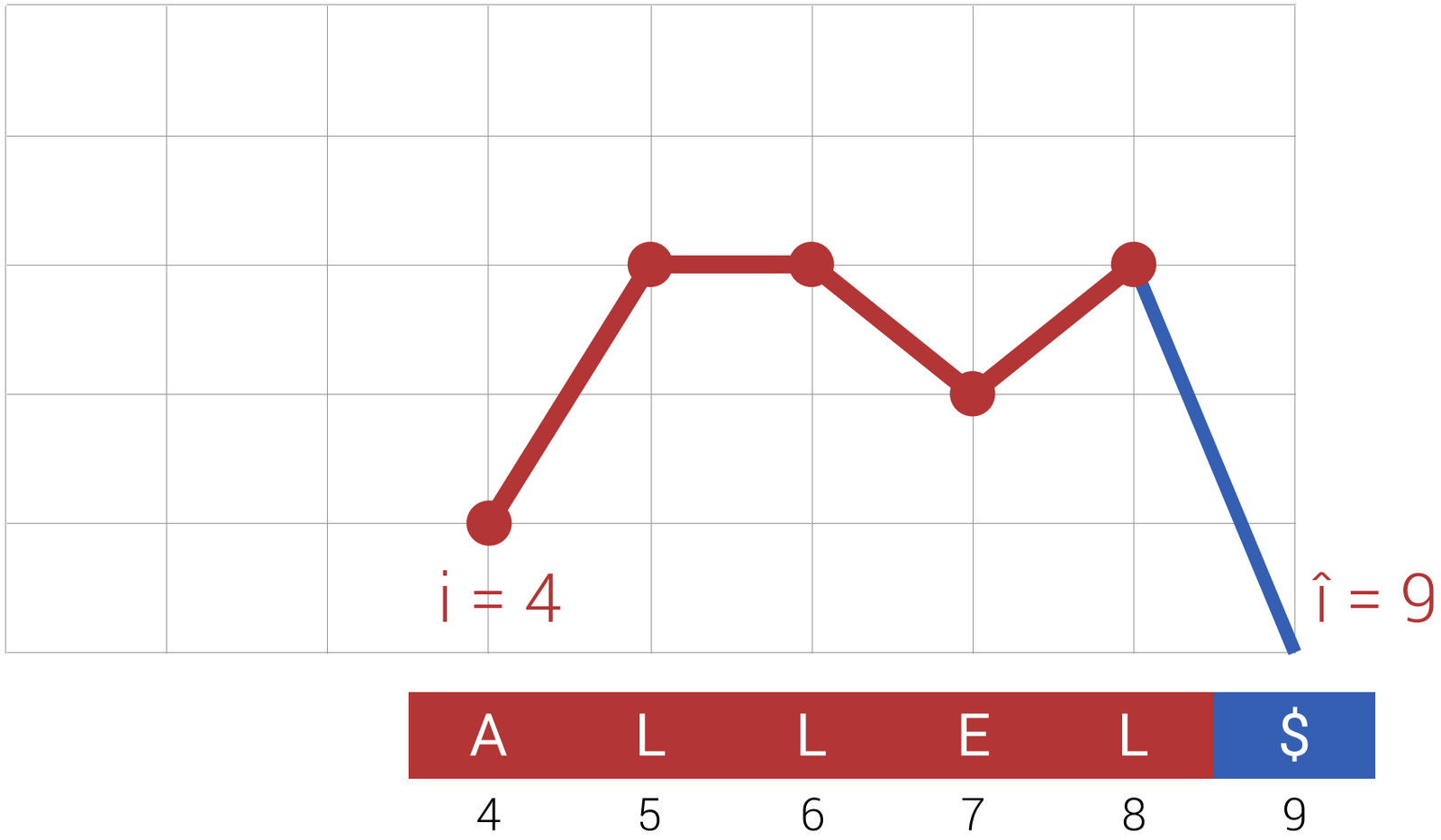
Gruppenkontext von S_i := S[i .. î)



A	R	A	L	L	E	L	$
2	3	4	5	6	7	8	9

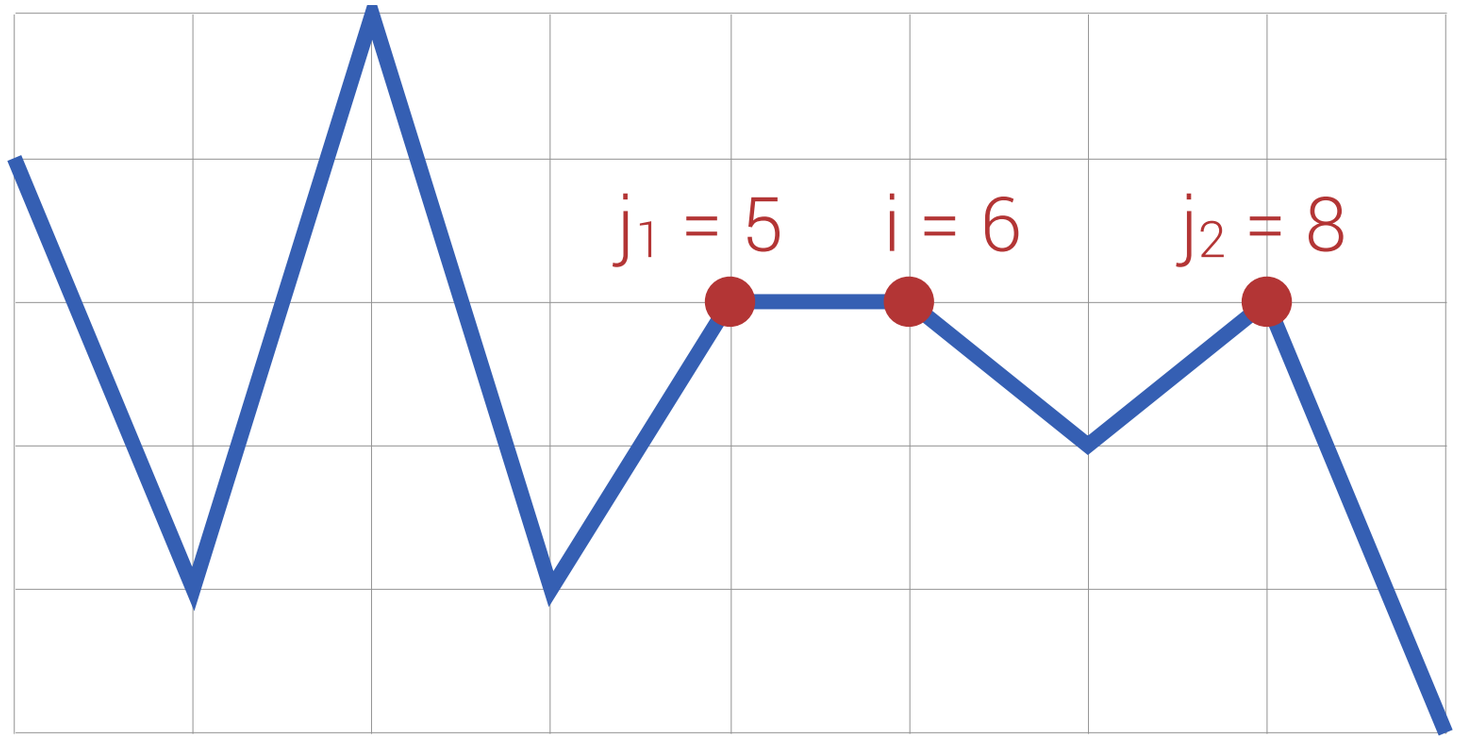
Definitionen

Gruppenkontext von S_i := $S[i \,..\, \hat{\imath})$



Definitionen

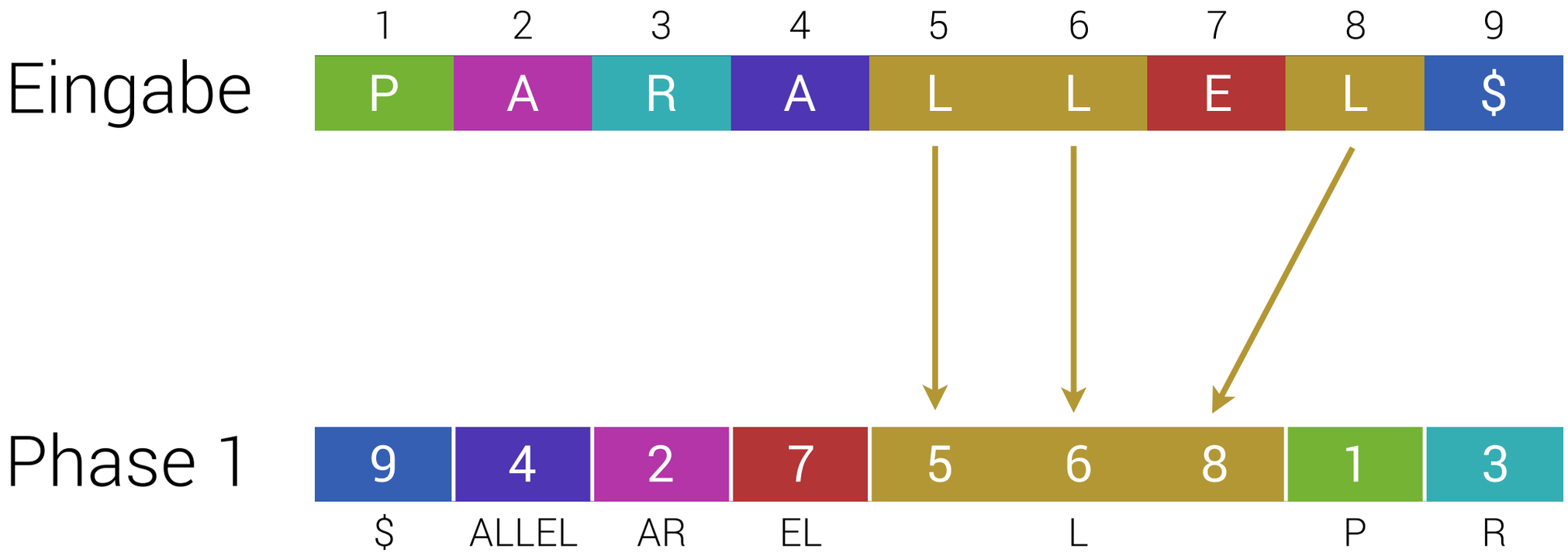
Gruppe von S_i := { S_j : Gr.kontext S_j = Gr.kontext S_i }



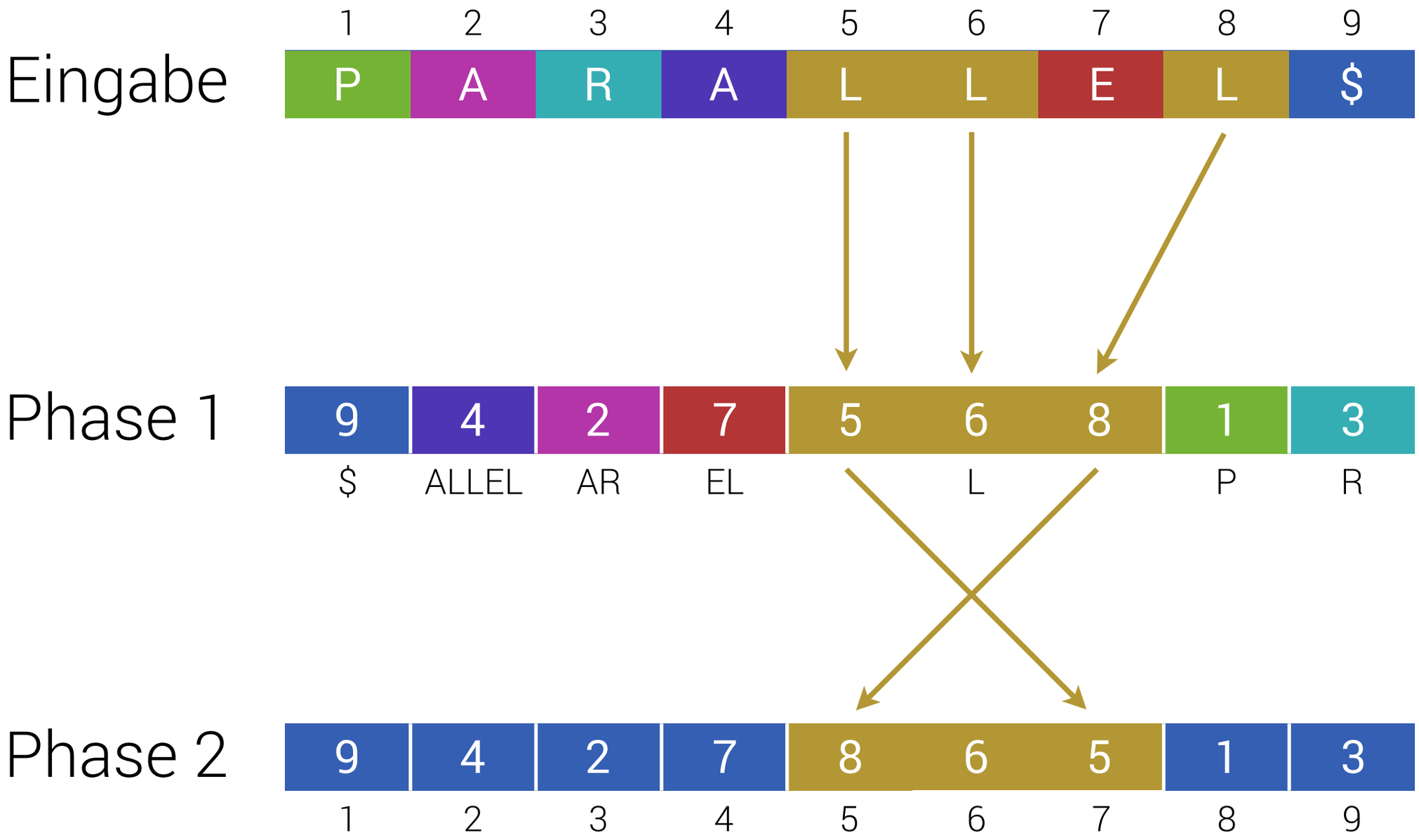
P	A	R	A	L	L	E	L	$
1	2	3	4	5	6	7	8	9

Grundprinzip

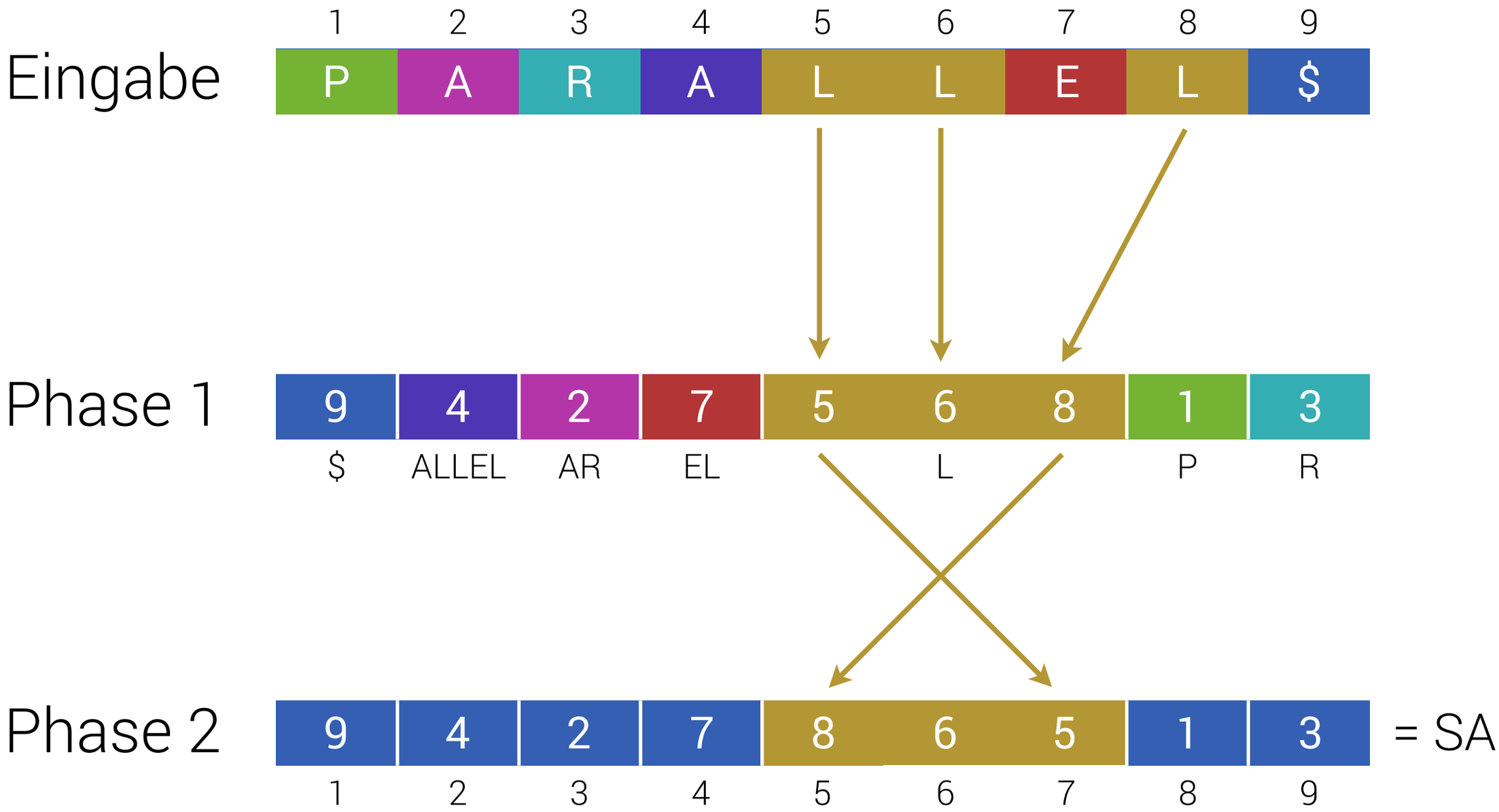
Grundprinzip



Grundprinzip



Grundprinzip



Grundprinzip

Eingabe

Phase 1 Sortierte Folge von Gruppen berechnen

Phase 2 Suffixe innerhalb der Gruppen sortieren

Grundprinzip

Eingabe

Phase 1 Sortierte Folge von Gruppen berechnen $O(n)$

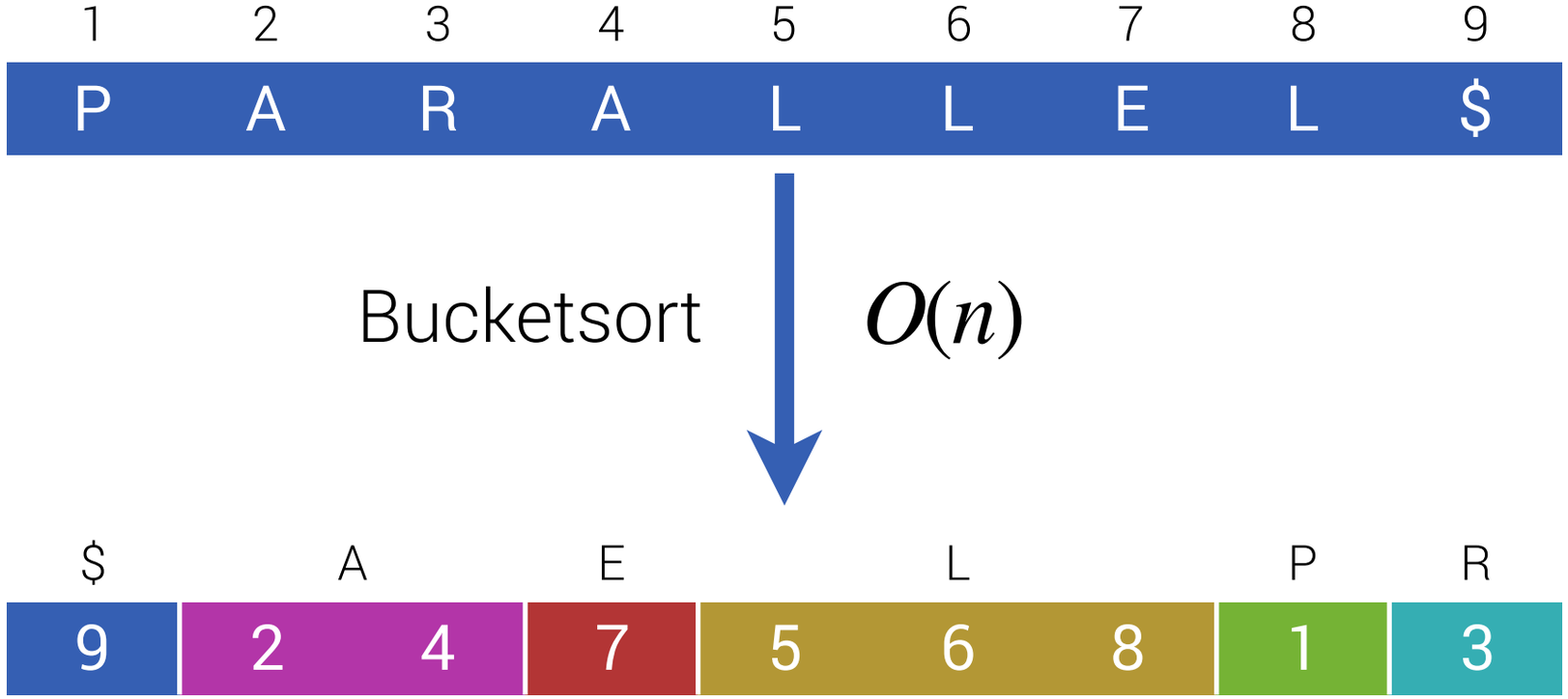
Phase 2 Suffixe innerhalb der Gruppen sortieren $O(n)$

Phase 1

Sortierte Folge von Gruppen berechnen

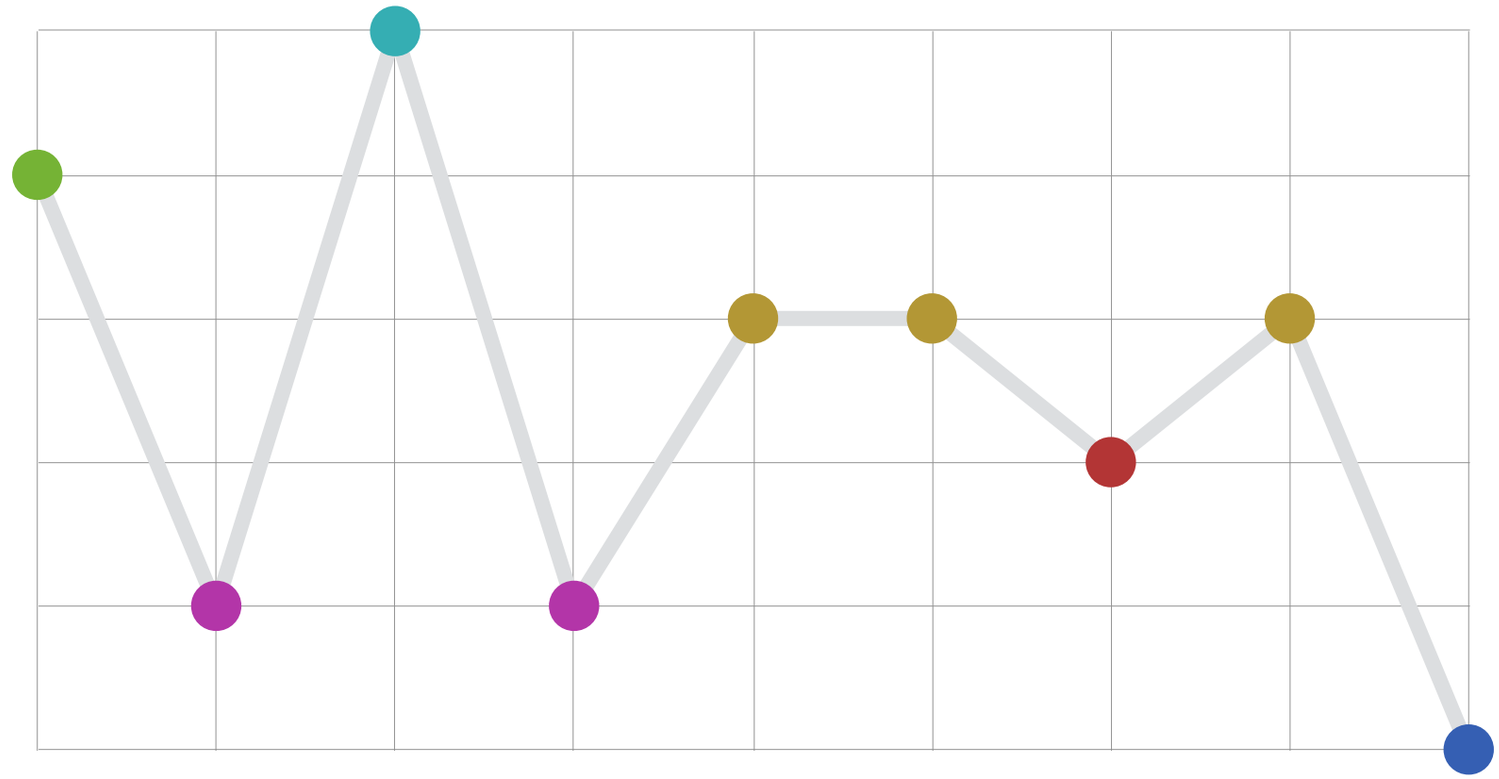
Phase 1

Sortierte Folge von Gruppen berechnen



Phase 1

$	A		E	L			P	R
9	2	4	7	5	6	8	1	3

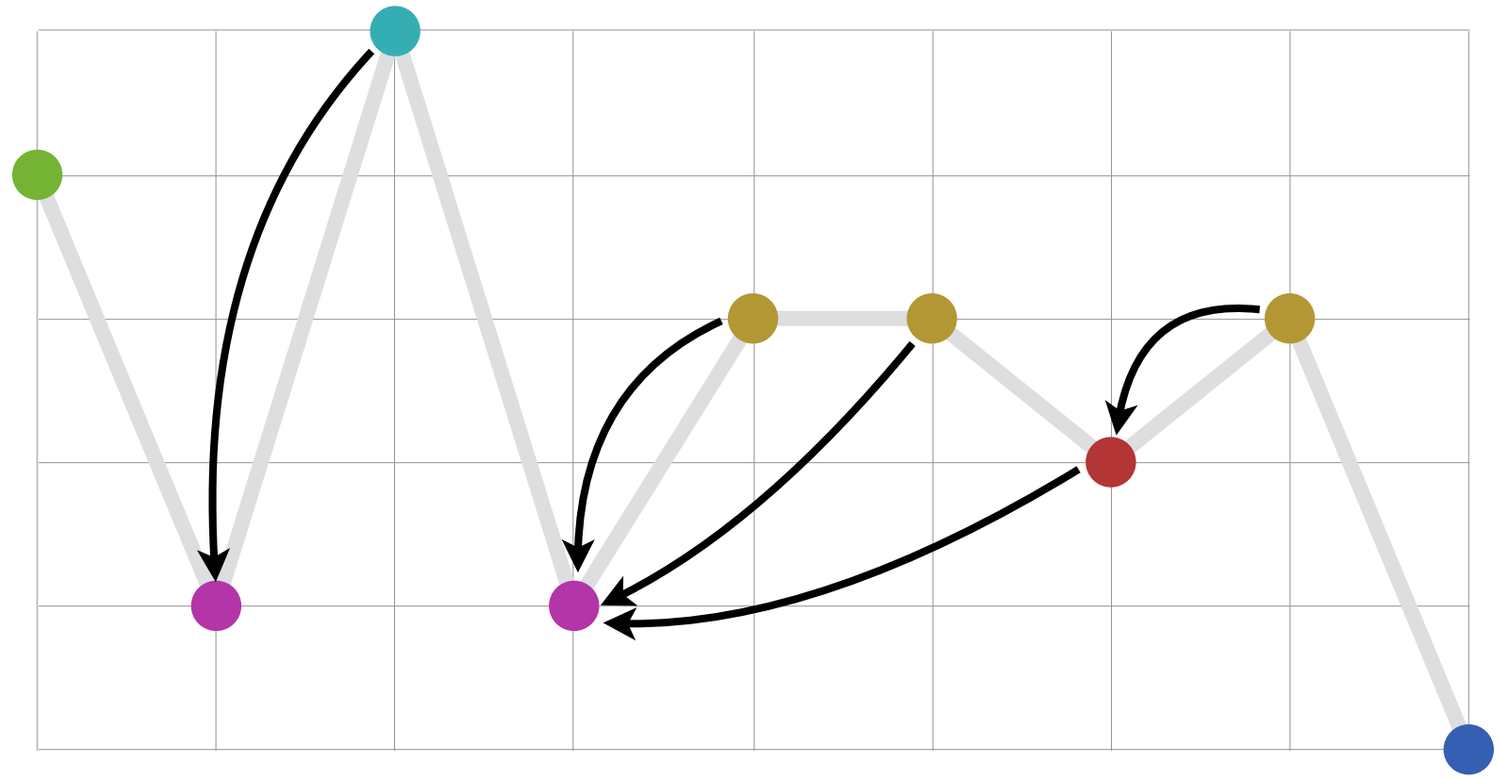


P	A	R	A	L	L	E	L	$
1	2	3	4	5	6	7	8	9

Phase 1

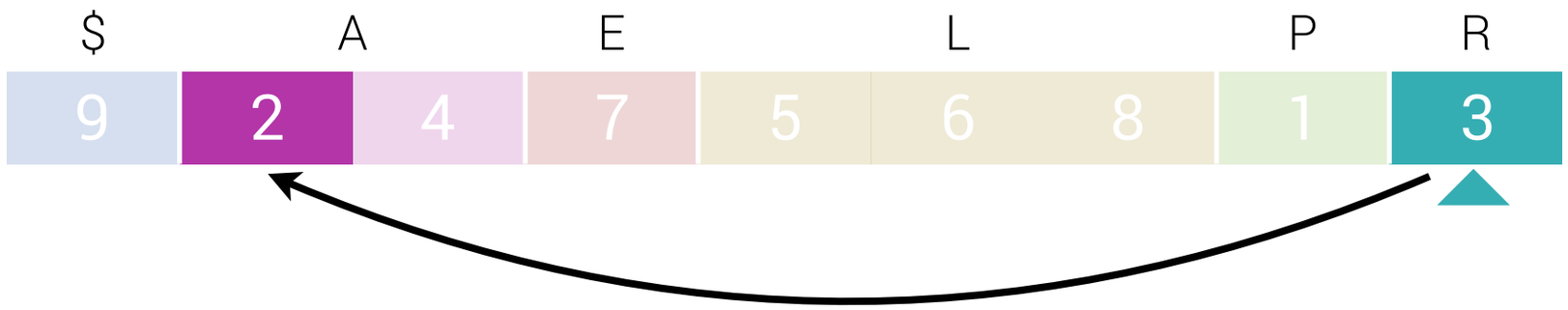
$	A	A	E	L	L	L	P	R
9	2	4	7	5	6	8	1	3

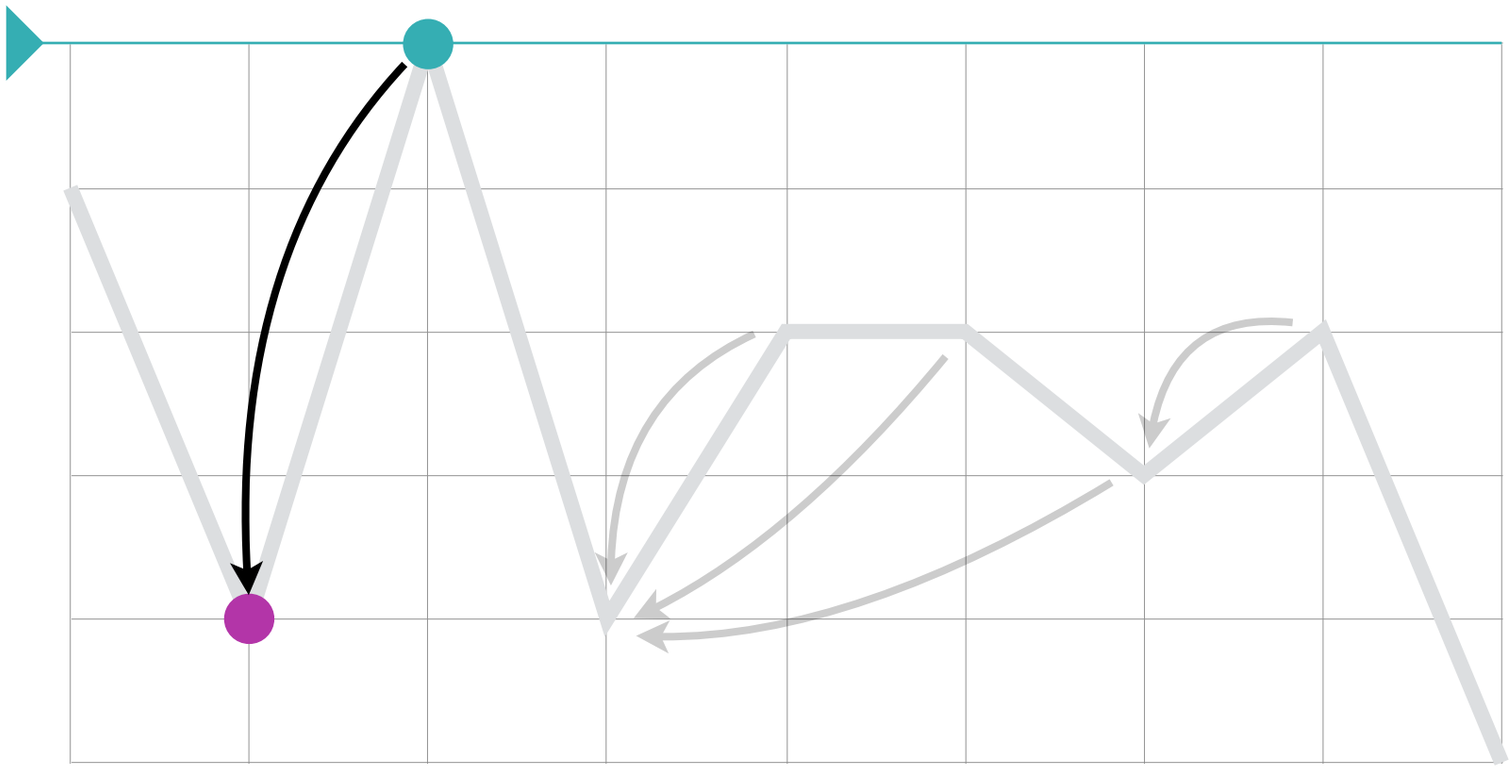
prev(**i**) := *max* { **j** ∈ [1 .. **i**]: Gr.kontext S_j <$_{lex}$ Gr.kontext S_i }



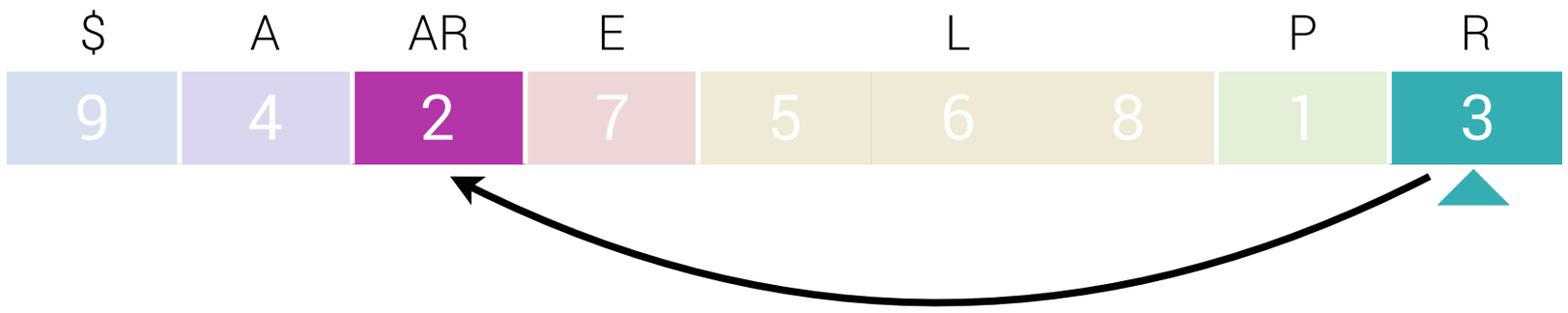
P	A	R	A	L	L	E	L	$
1	2	3	4	5	6	7	8	9

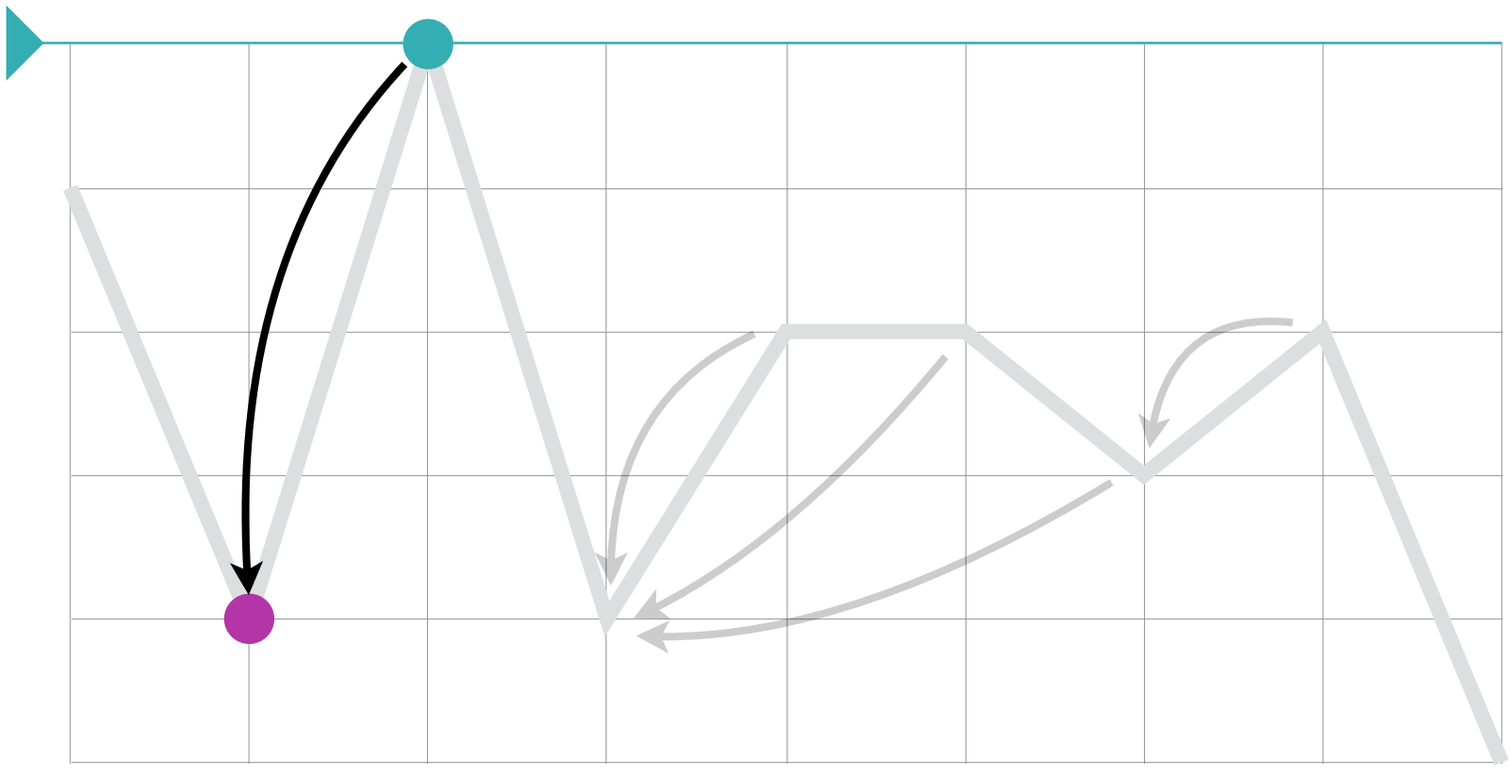
Phase 1





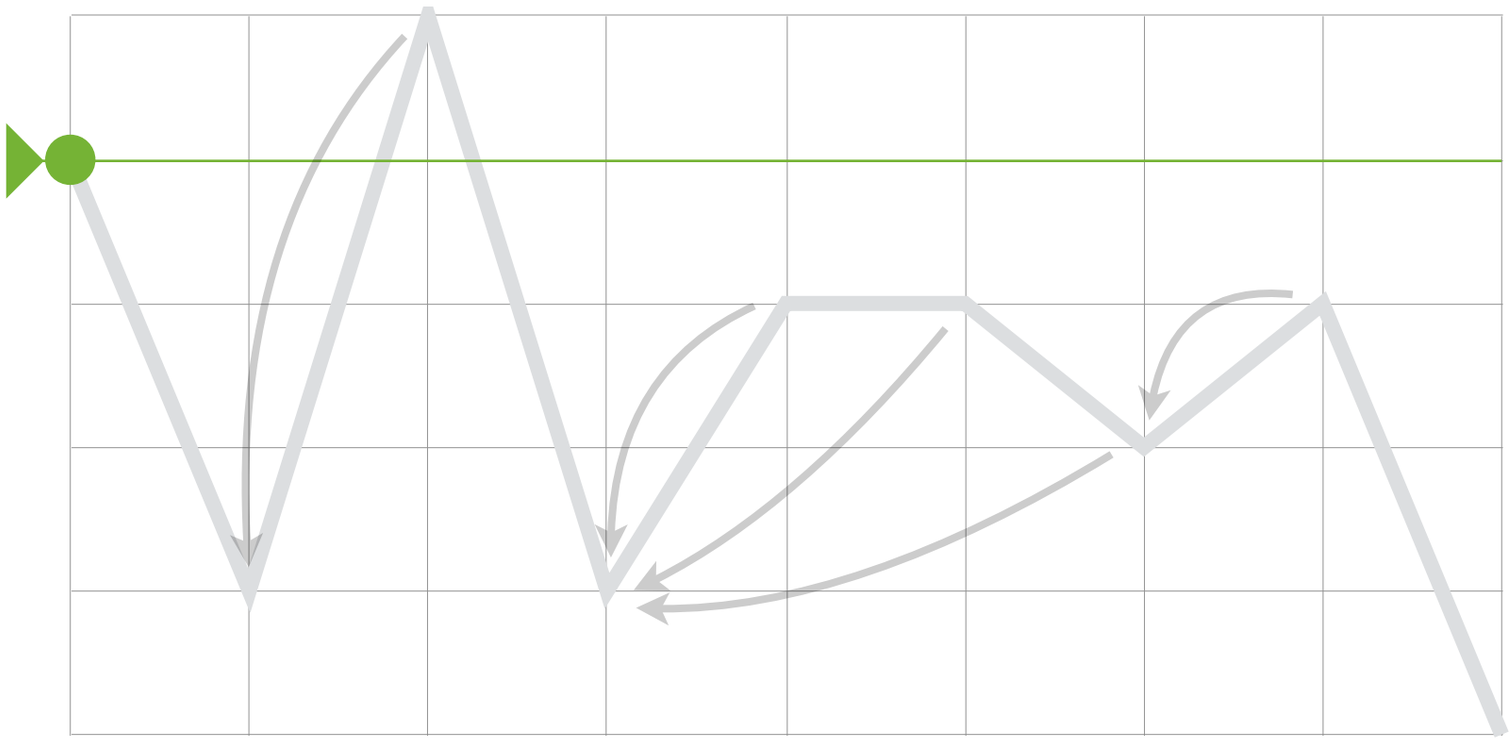
Phase 1





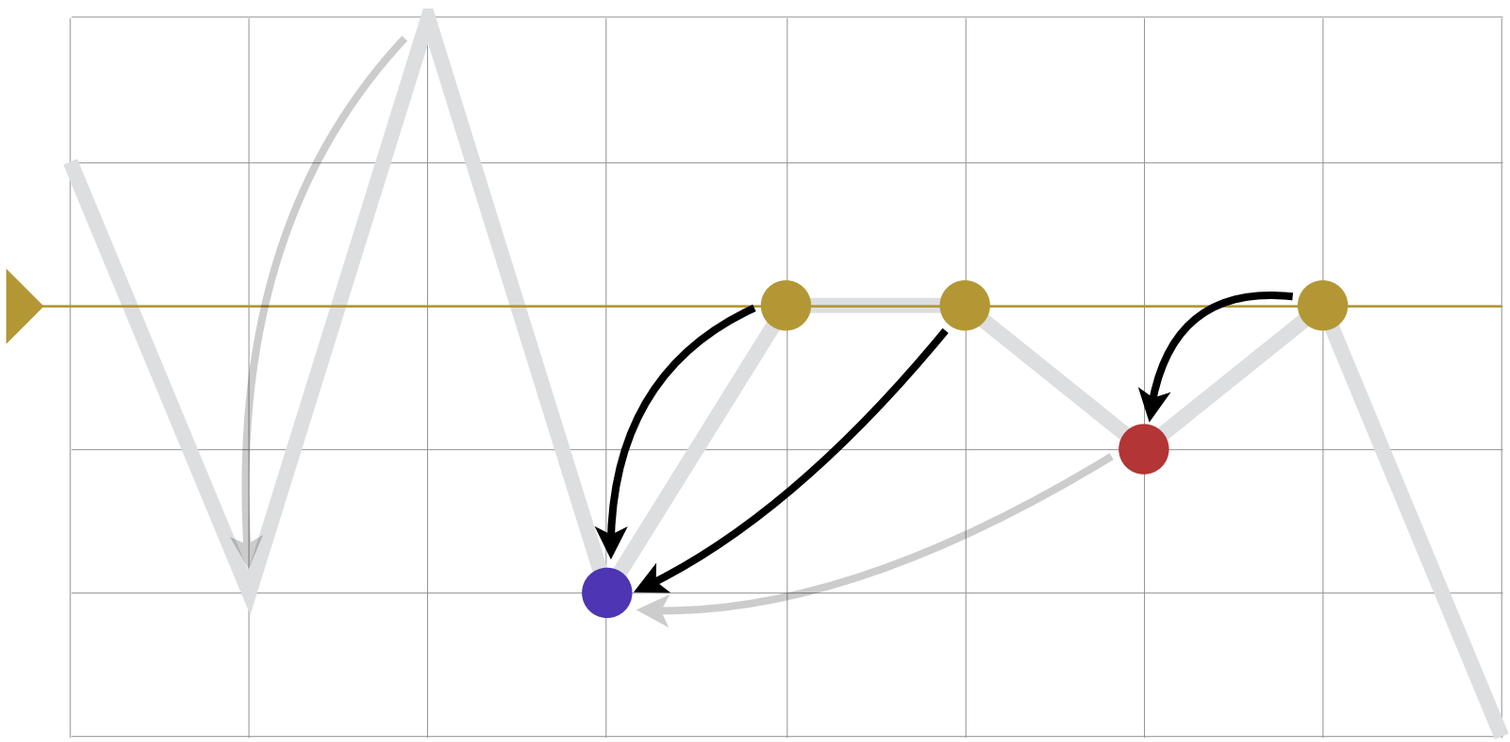
Phase 1

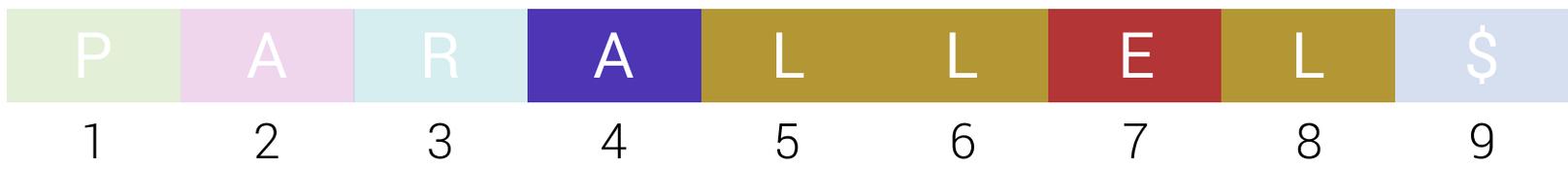
$	A	AR	E		L		P	R
9	4	2	7	5	6	8	1	3



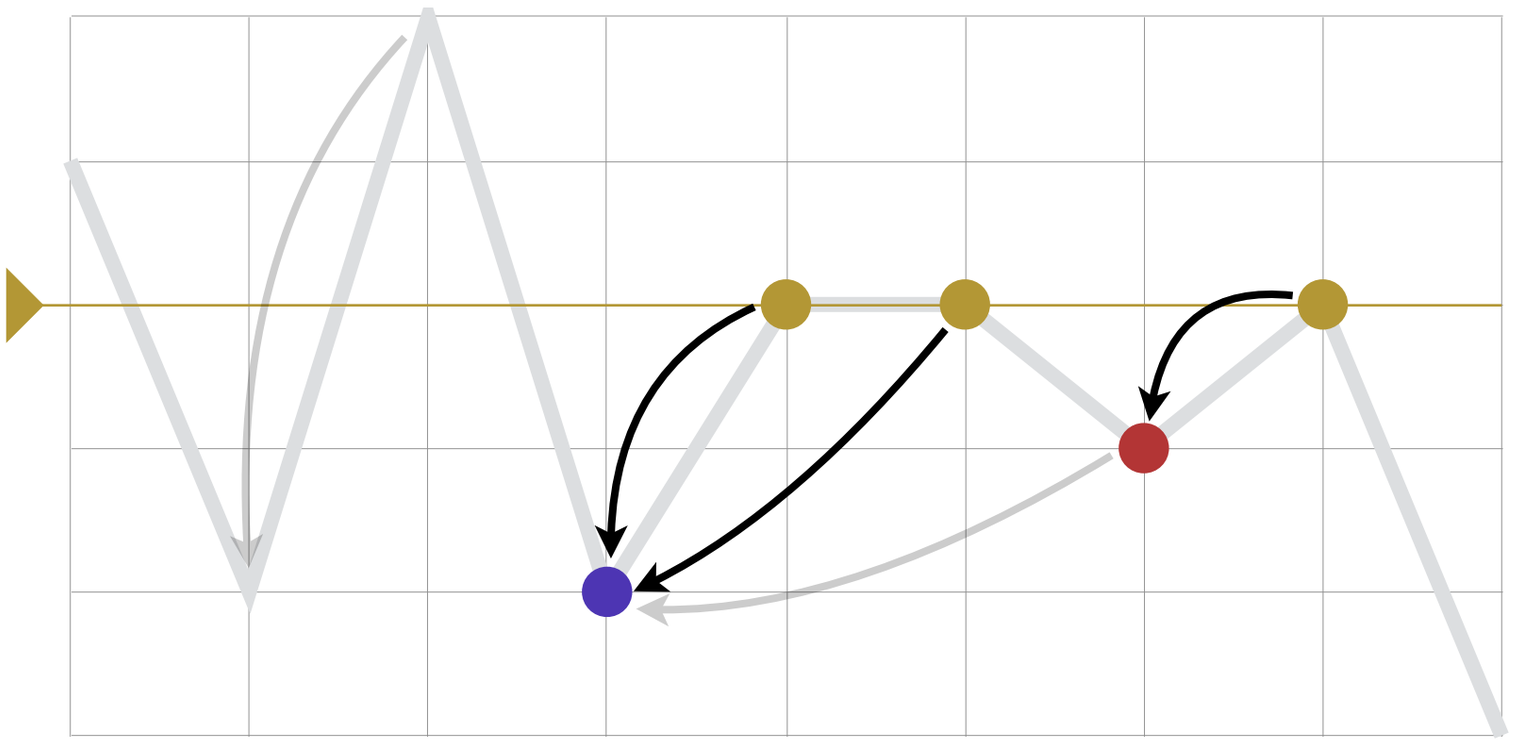
P	A	R	A	L	L	E	L	$
1	2	3	4	5	6	7	8	9

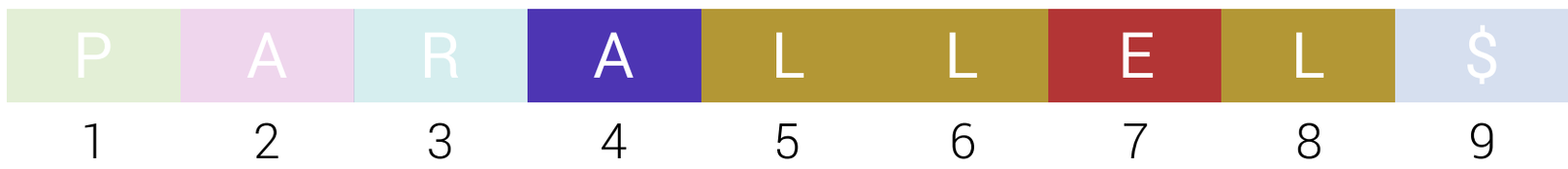
Phase 1



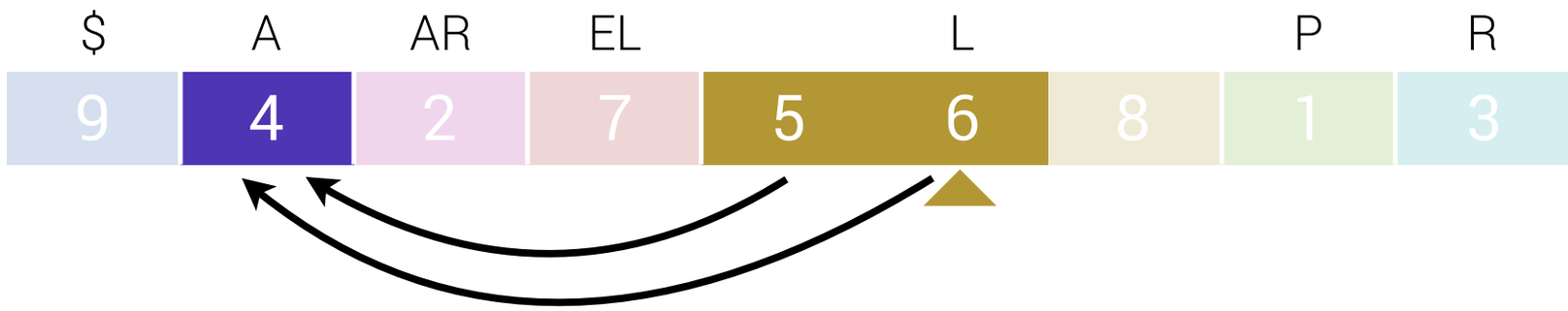


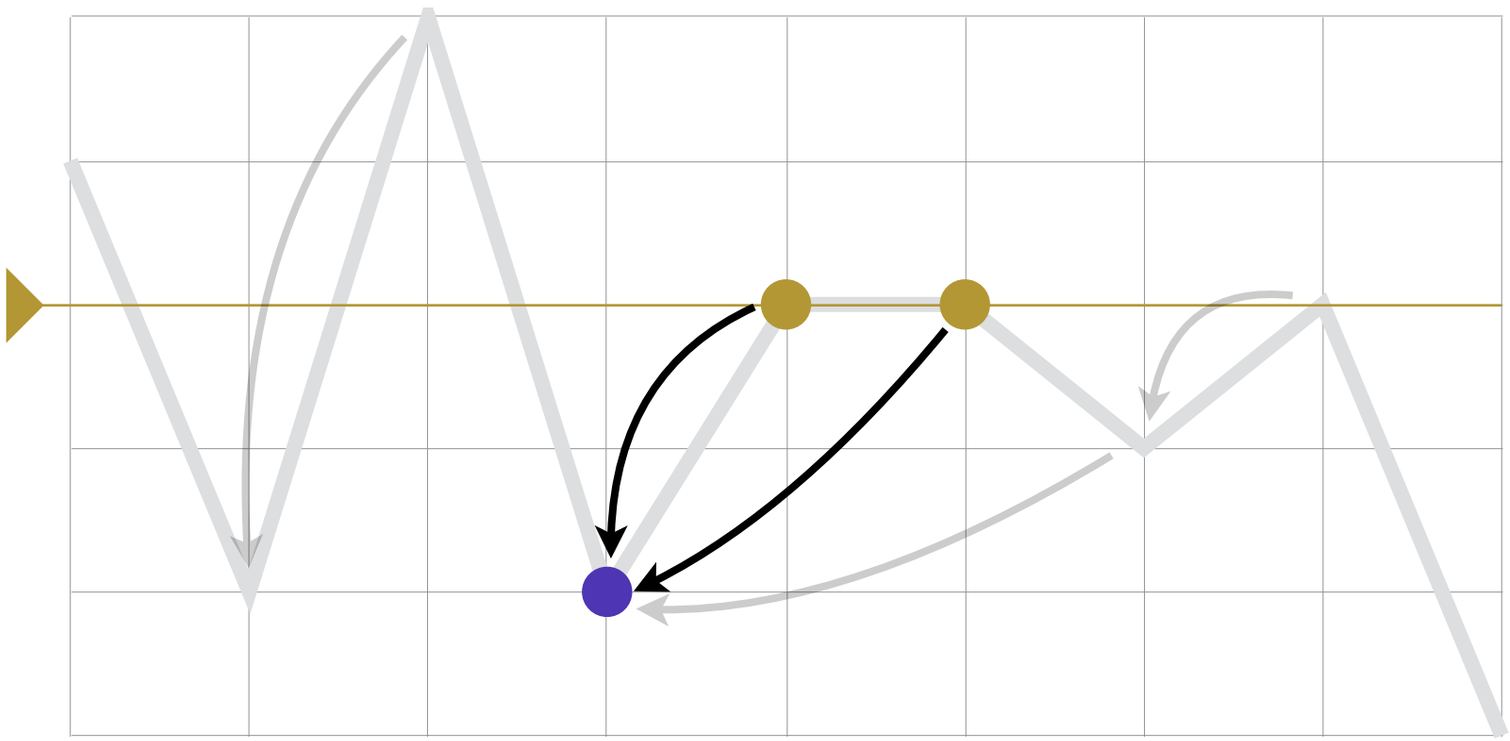
Phase 1

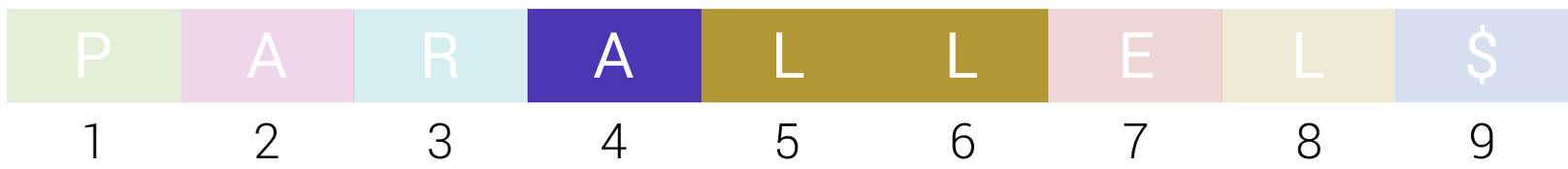




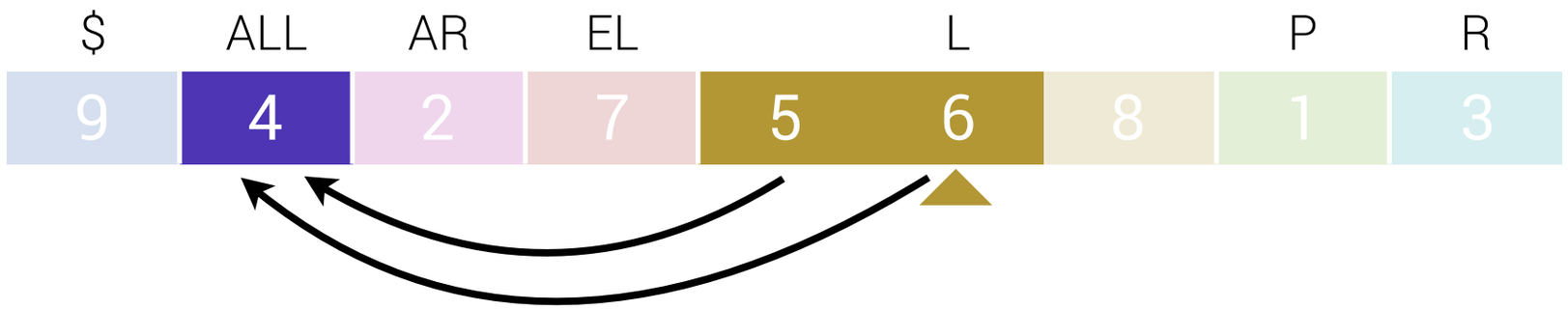
Phase 1

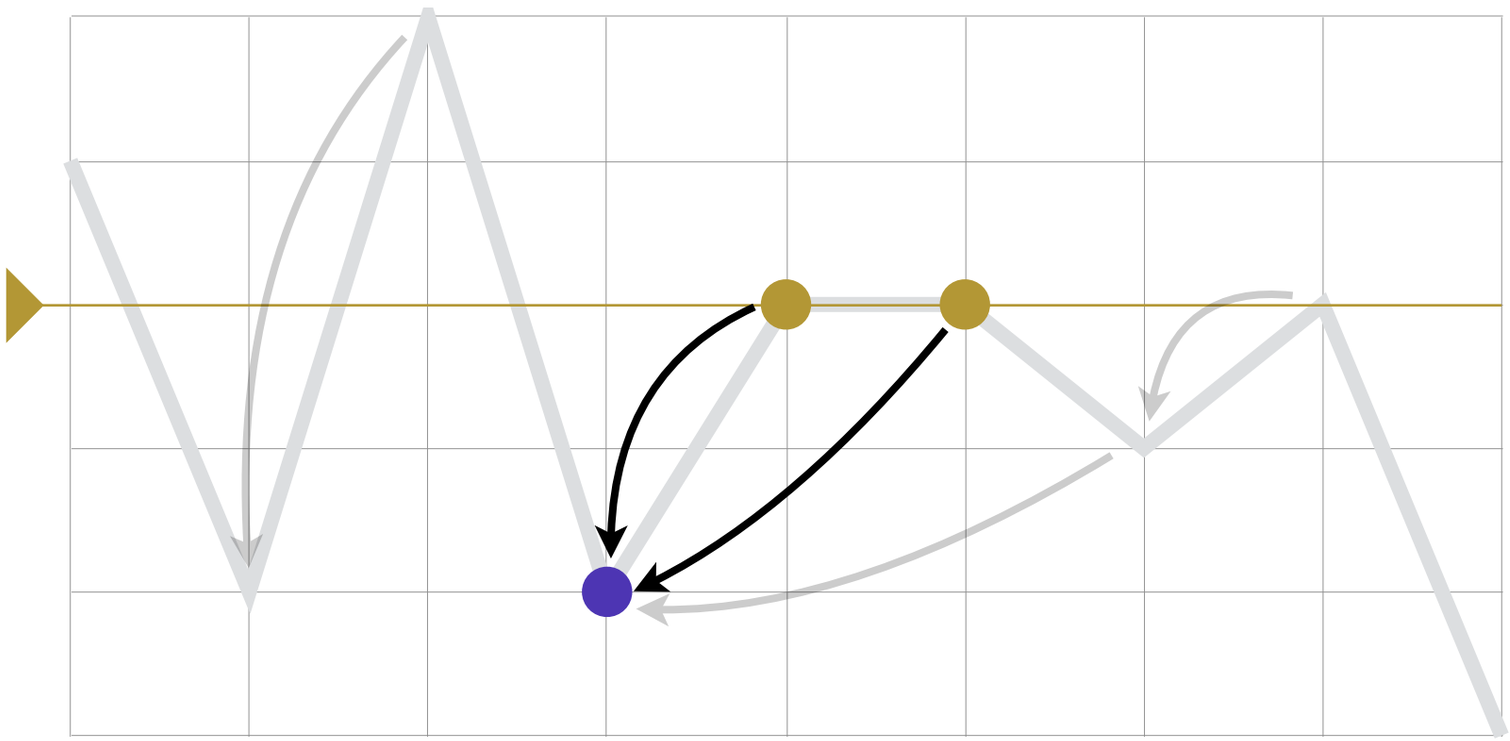




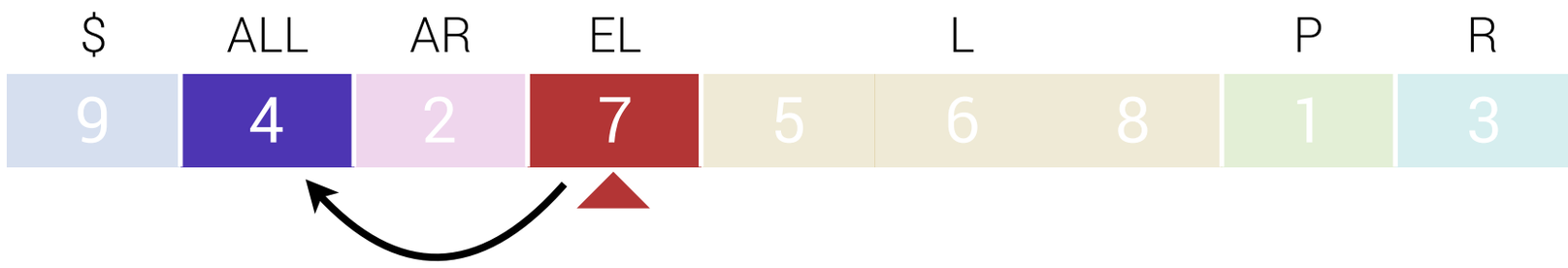


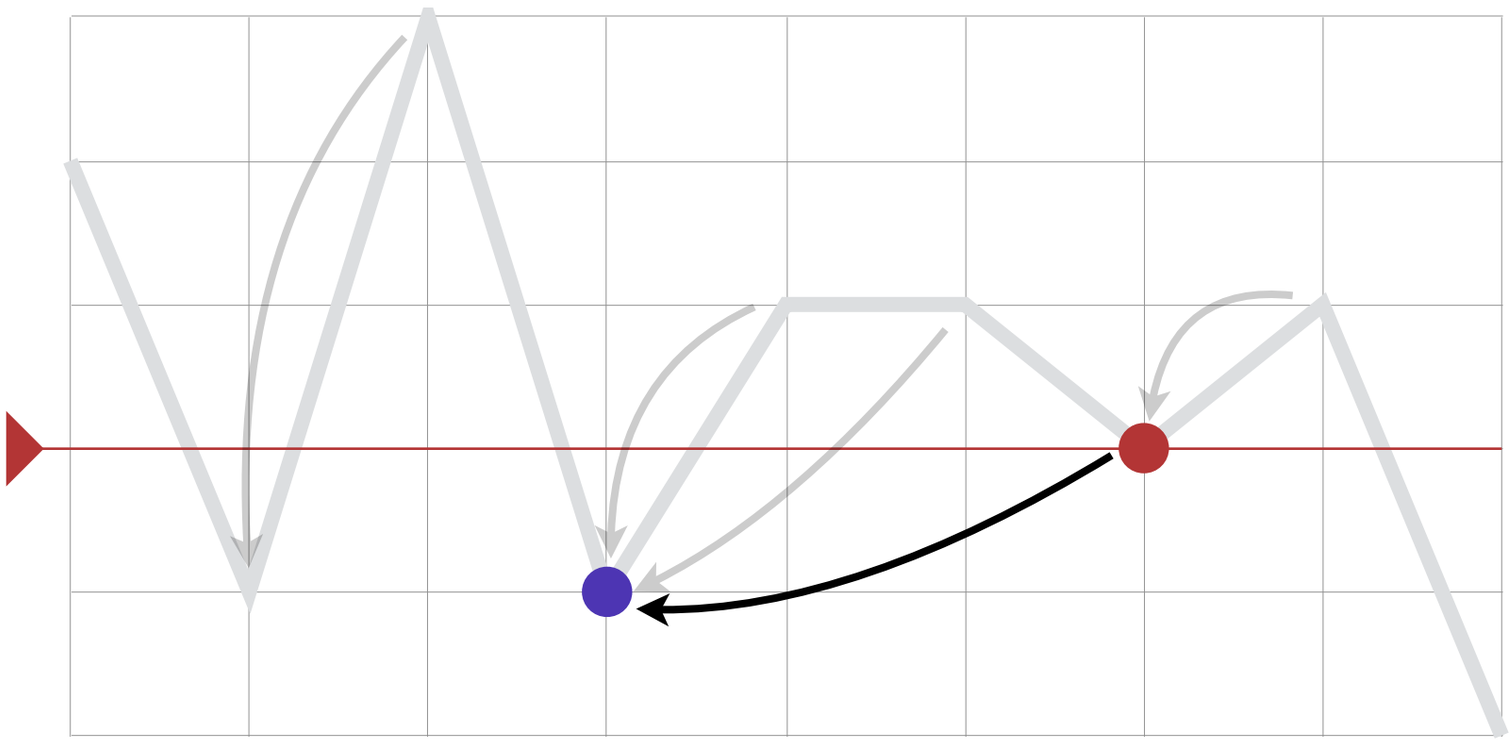
Phase 1



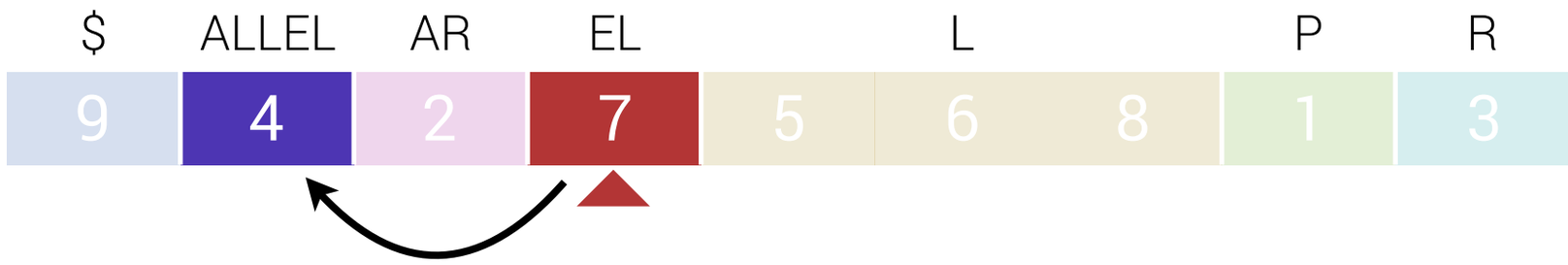


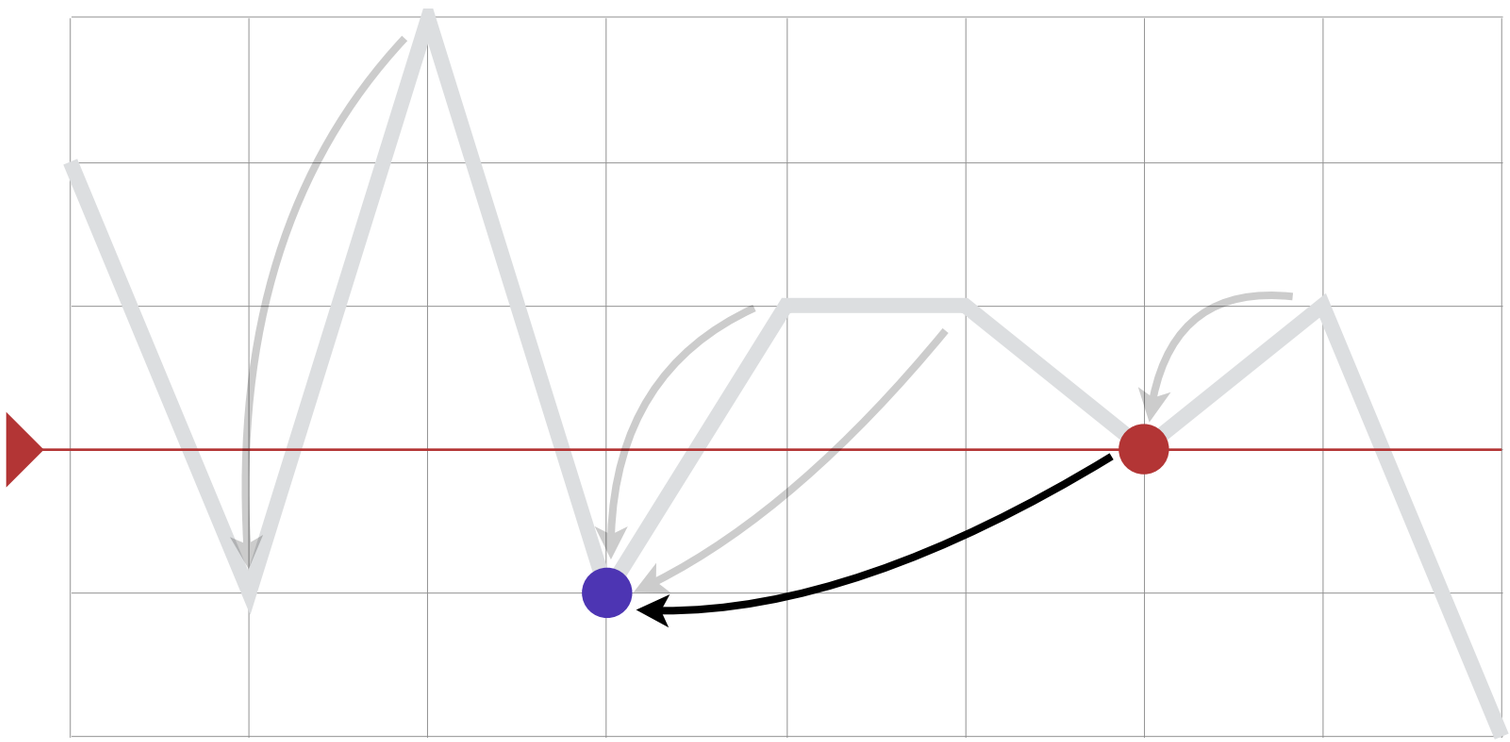
Phase 1





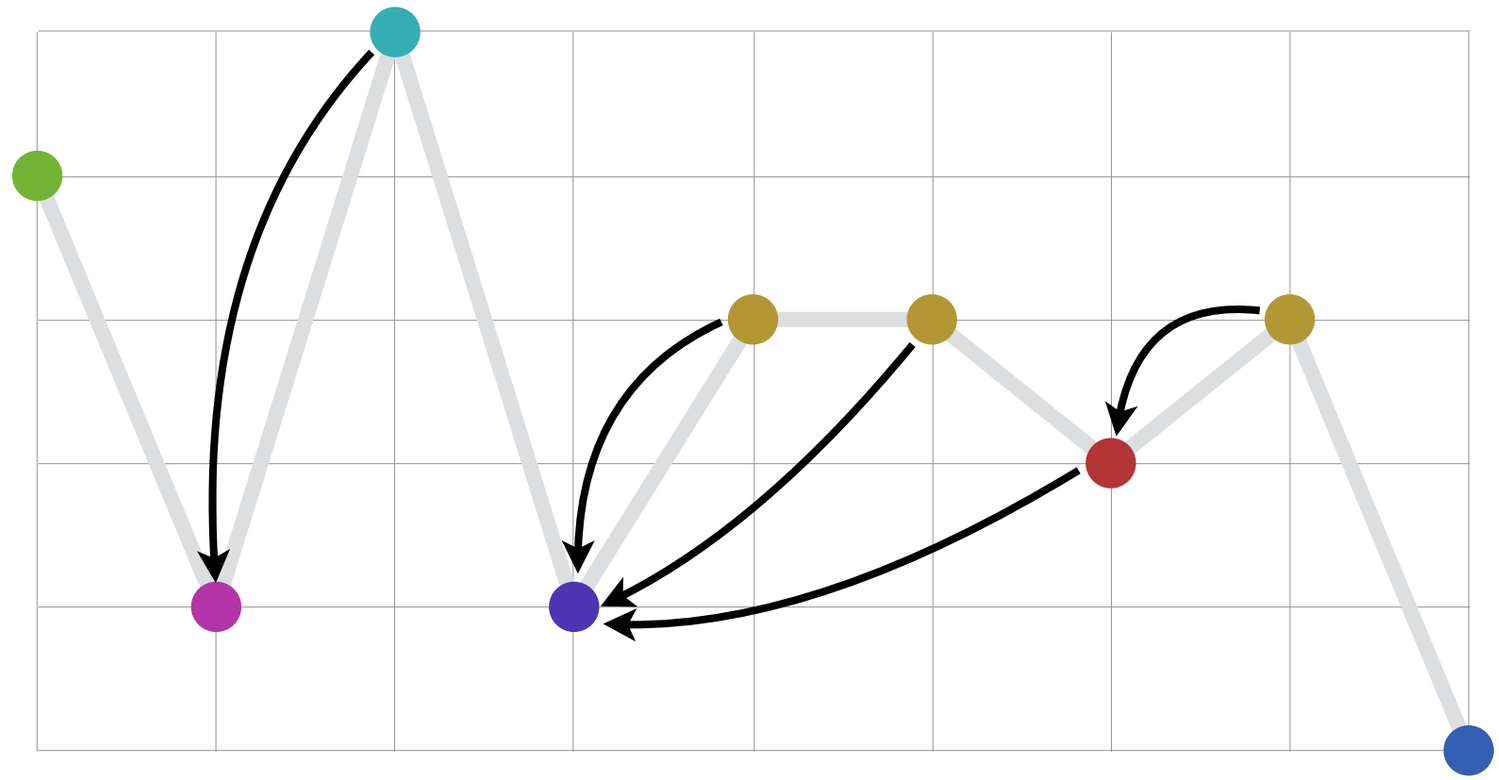
Phase 1





Phase 1

$	ALLEL	AR	EL		L		P	R
9	4	2	7	5	6	8	1	3

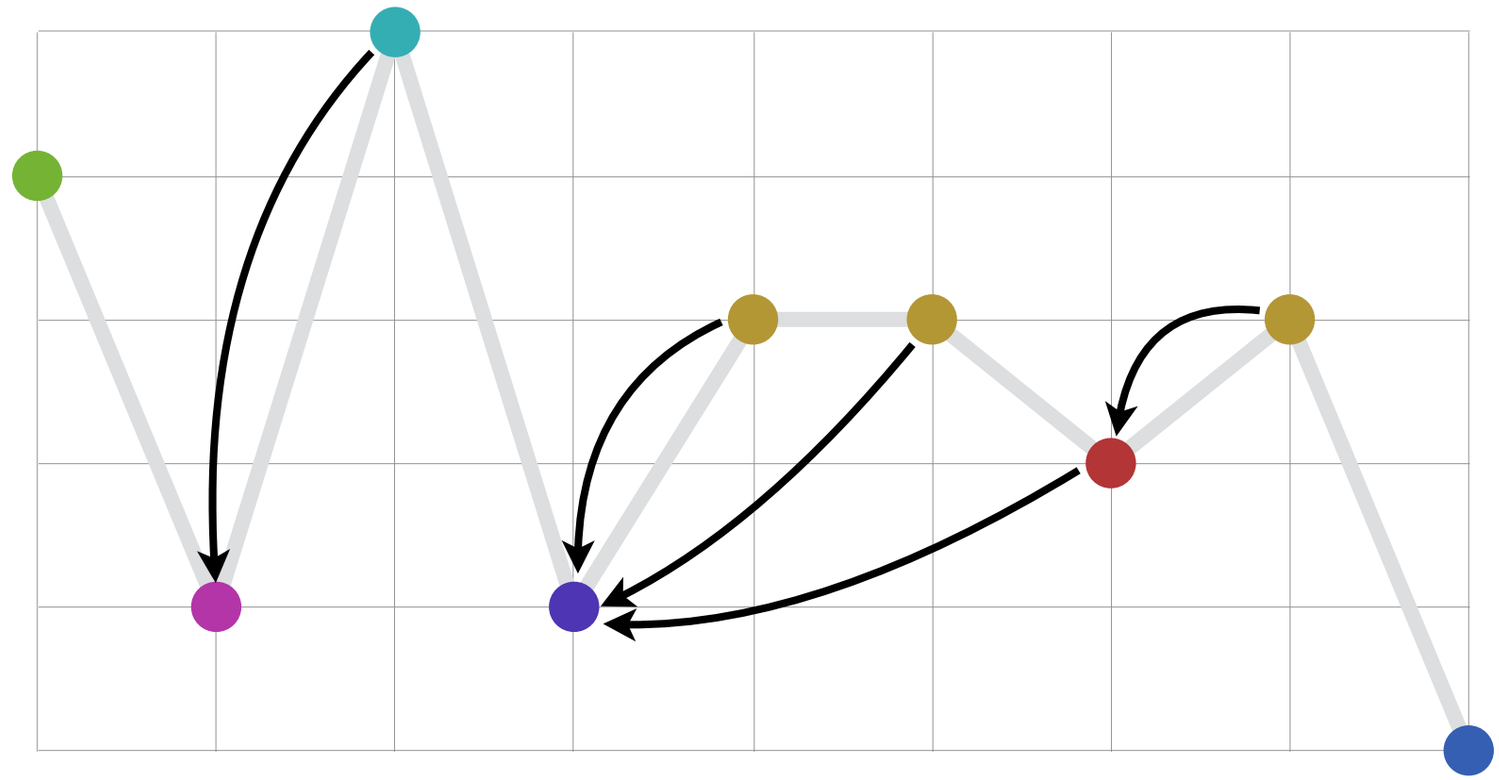


P	A	R	A	L	L	E	L	$
1	2	3	4	5	6	7	8	9

Phase 1

$	ALLEL	AR	EL		L		P	R
9	4	2	7	5	6	8	1	3

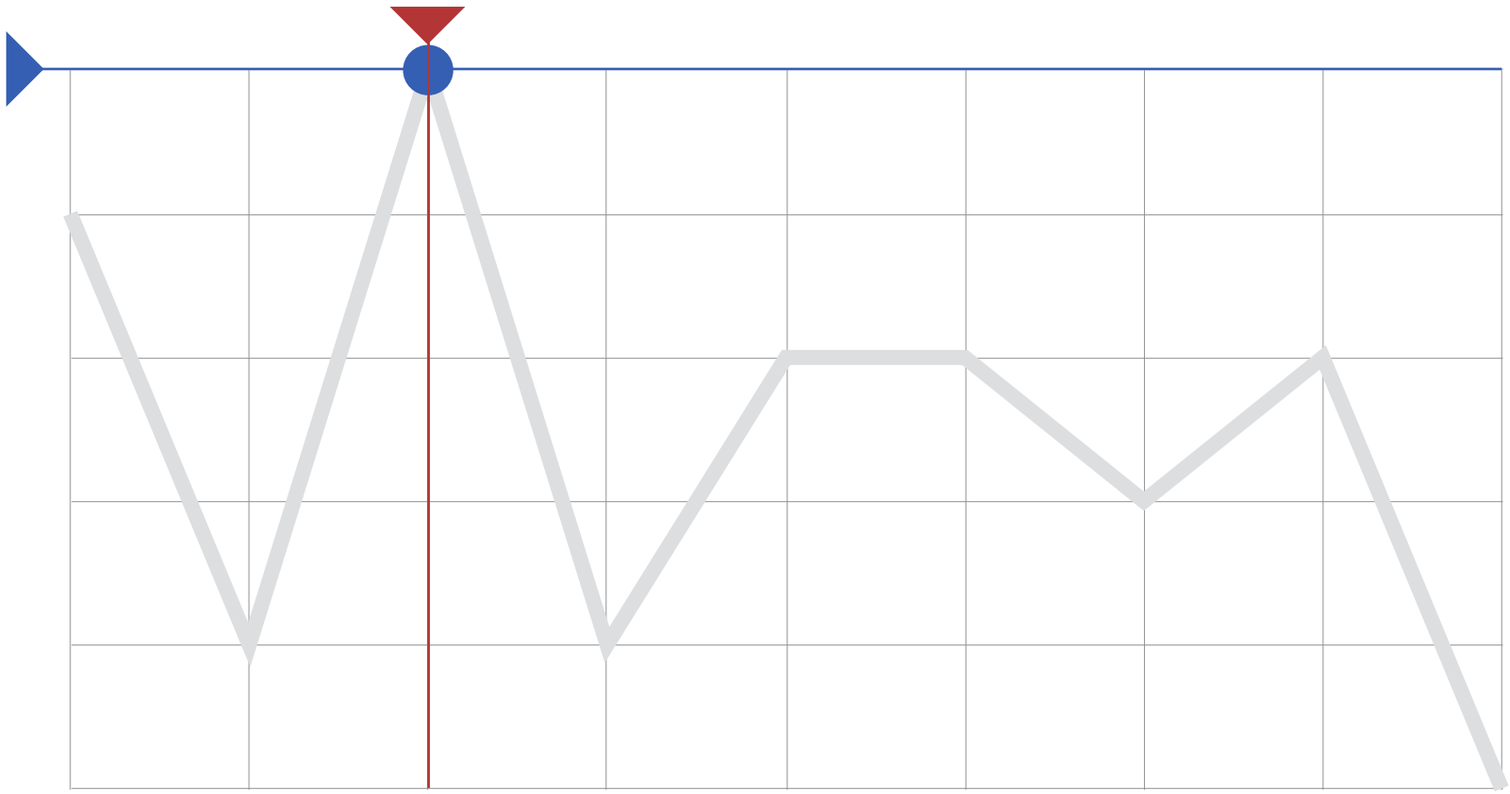
prev(**i**) := *max* { **j** ∈ [1 .. **i**]: Gr.kontext S_j <$_{lex}$ Gr.kontext S_i }



P	A	R	A	L	L	E	L	$
1	2	3	4	5	6	7	8	9

Phase 1

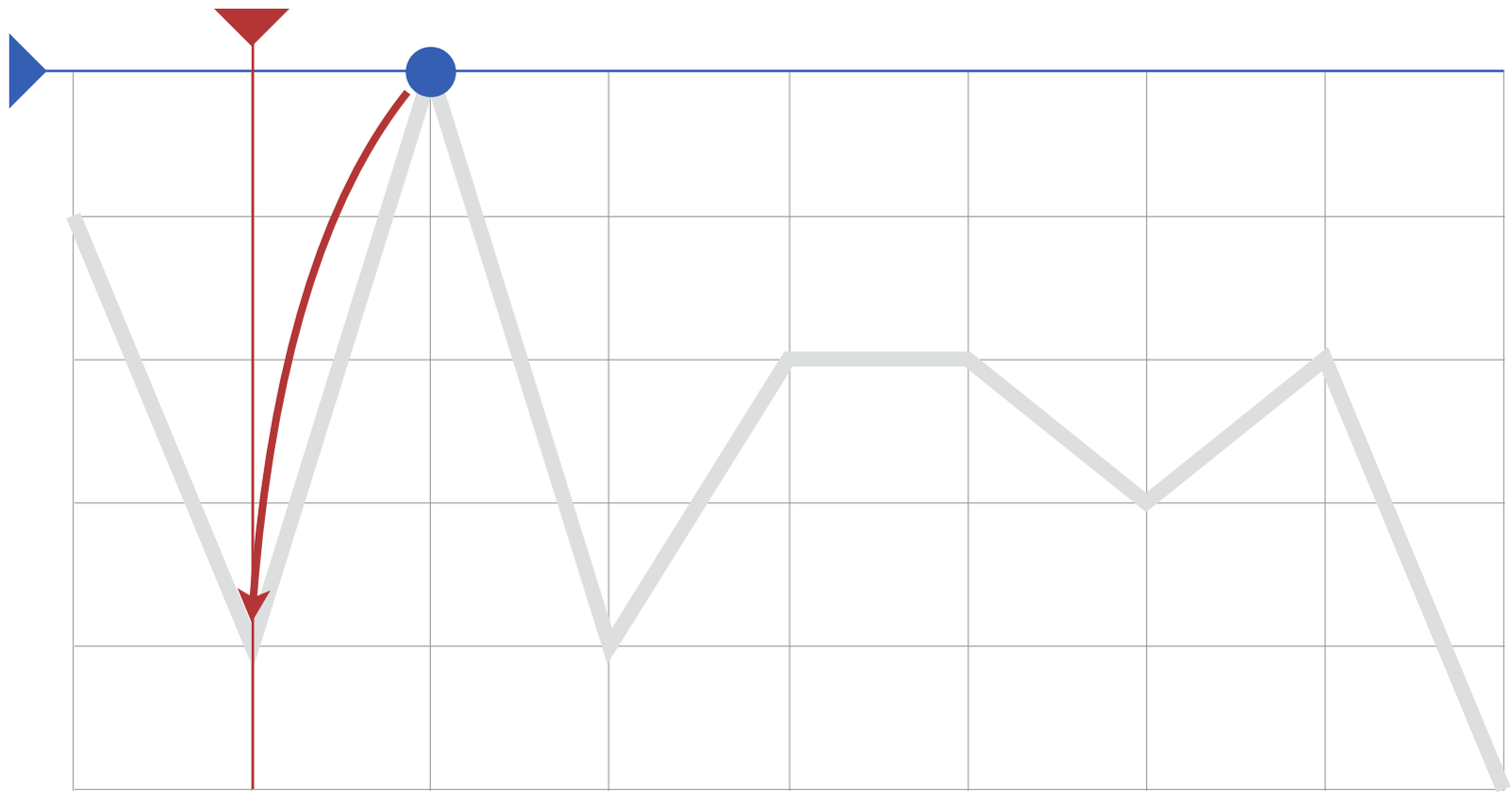
prev(i) := *max* { j ∈ [1 .. i]: Gr.kontext S_j <$_{lex}$ Gr.kontext S_i }



P	A	R	A	L	L	E	L	$
1	2	3	4	5	6	7	8	9

Phase 1

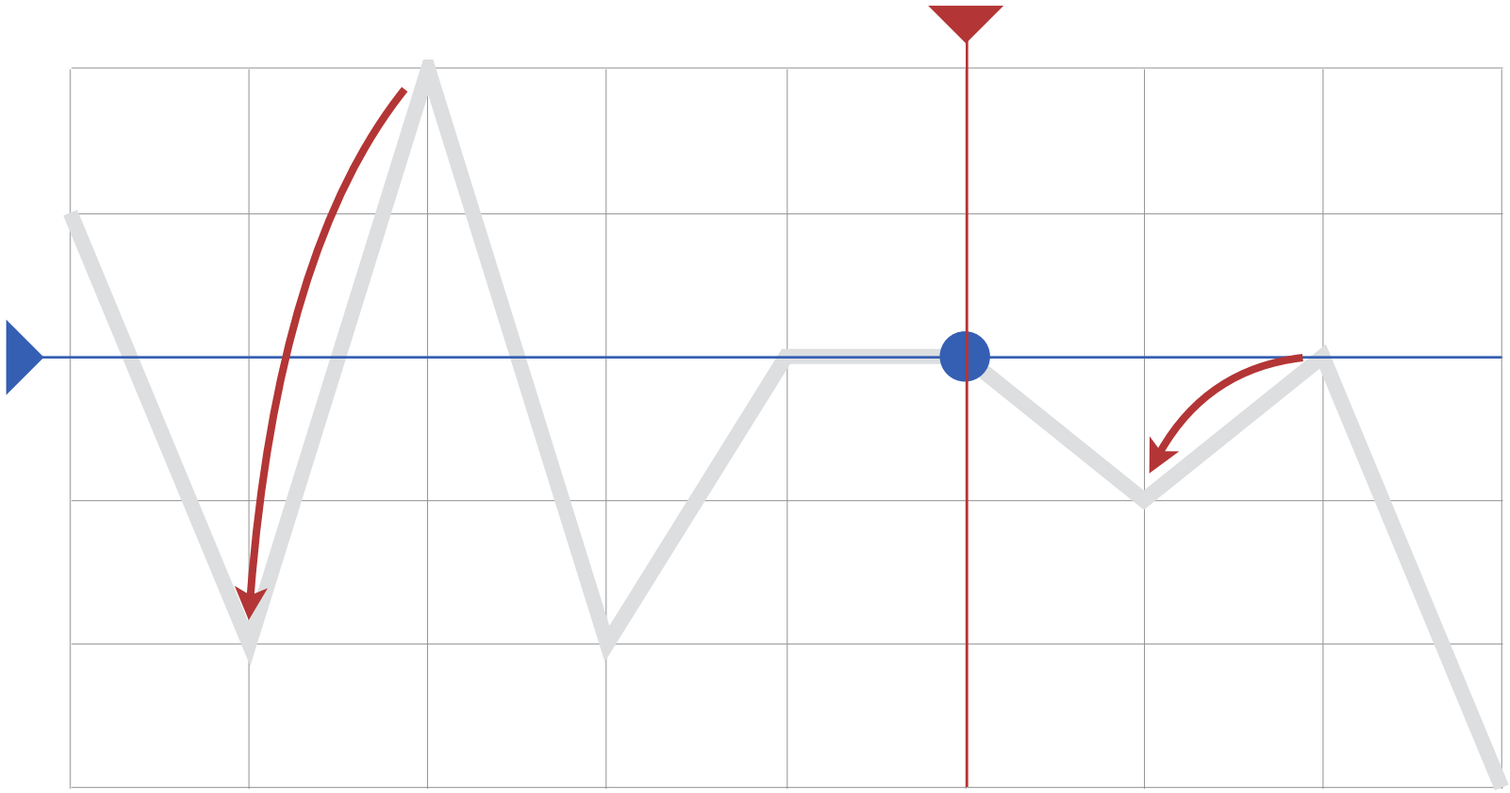
prev(i) := *max* { j ∈ [1 .. i]: Gr.kontext S_j <$_{lex}$ Gr.kontext S_i }



P	A	R	A	L	L	E	L	$
1	2	3	4	5	6	7	8	9

Phase 1

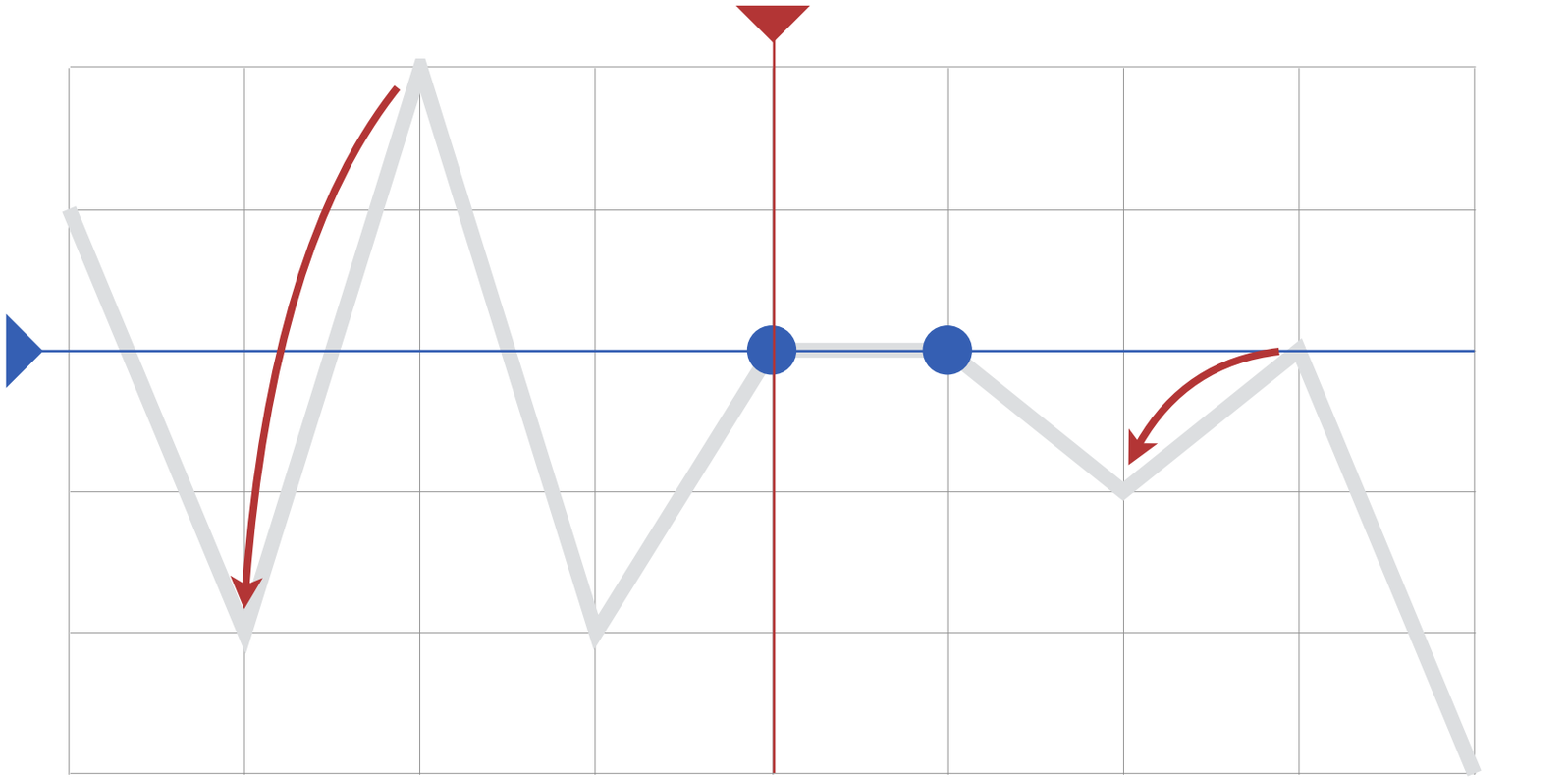
prev(i) := *max* { j ∈ [1 .. i]: Gr.kontext S_j <$_{lex}$ Gr.kontext S_i }



P	A	R	A	L	L	E	L	$
1	2	3	4	5	6	7	8	9

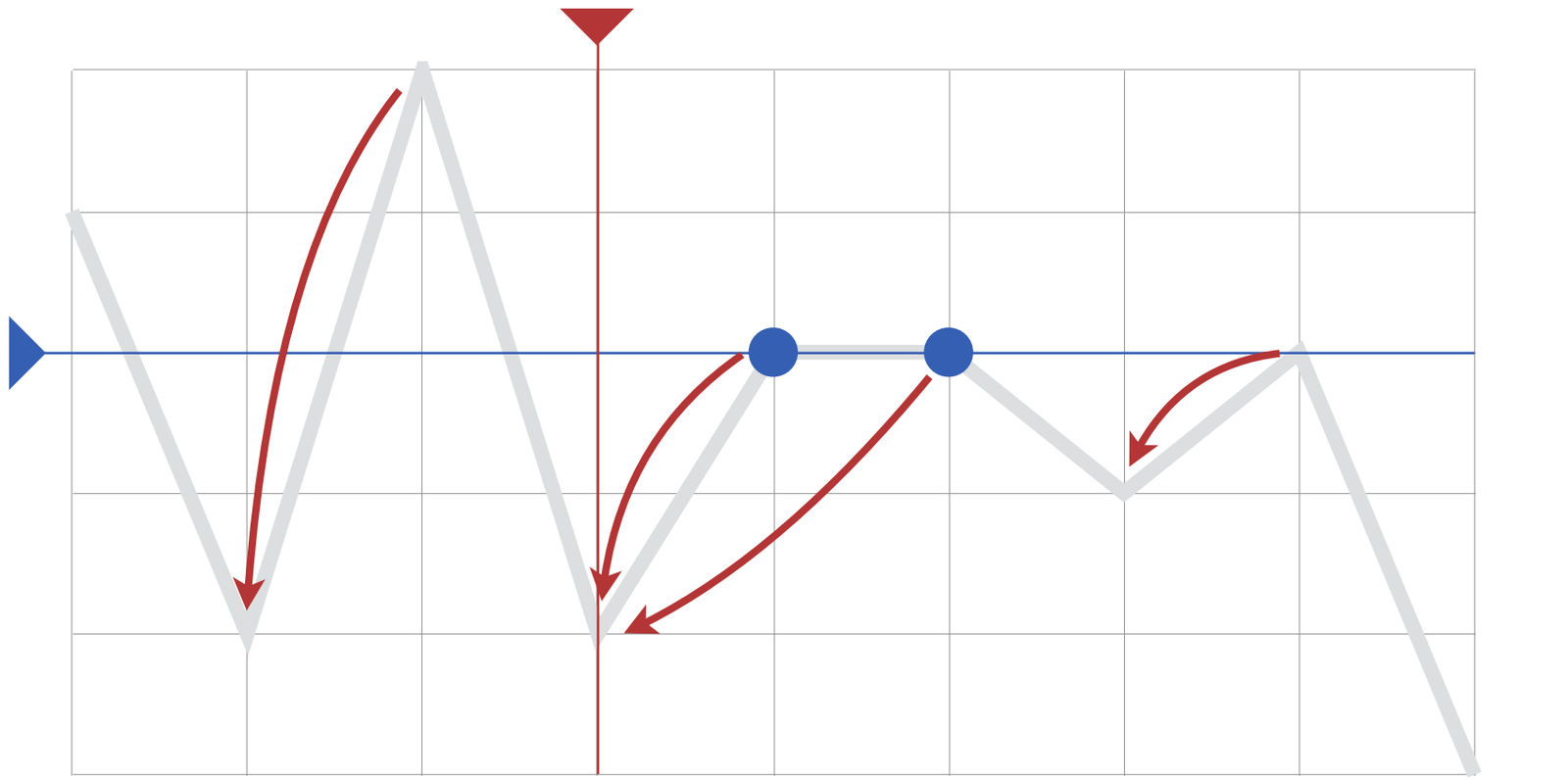
Phase 1

prev(i) := *max* { j ∈ [1 .. i]: Gr.kontext S_j <$_{lex}$ Gr.kontext S_i }



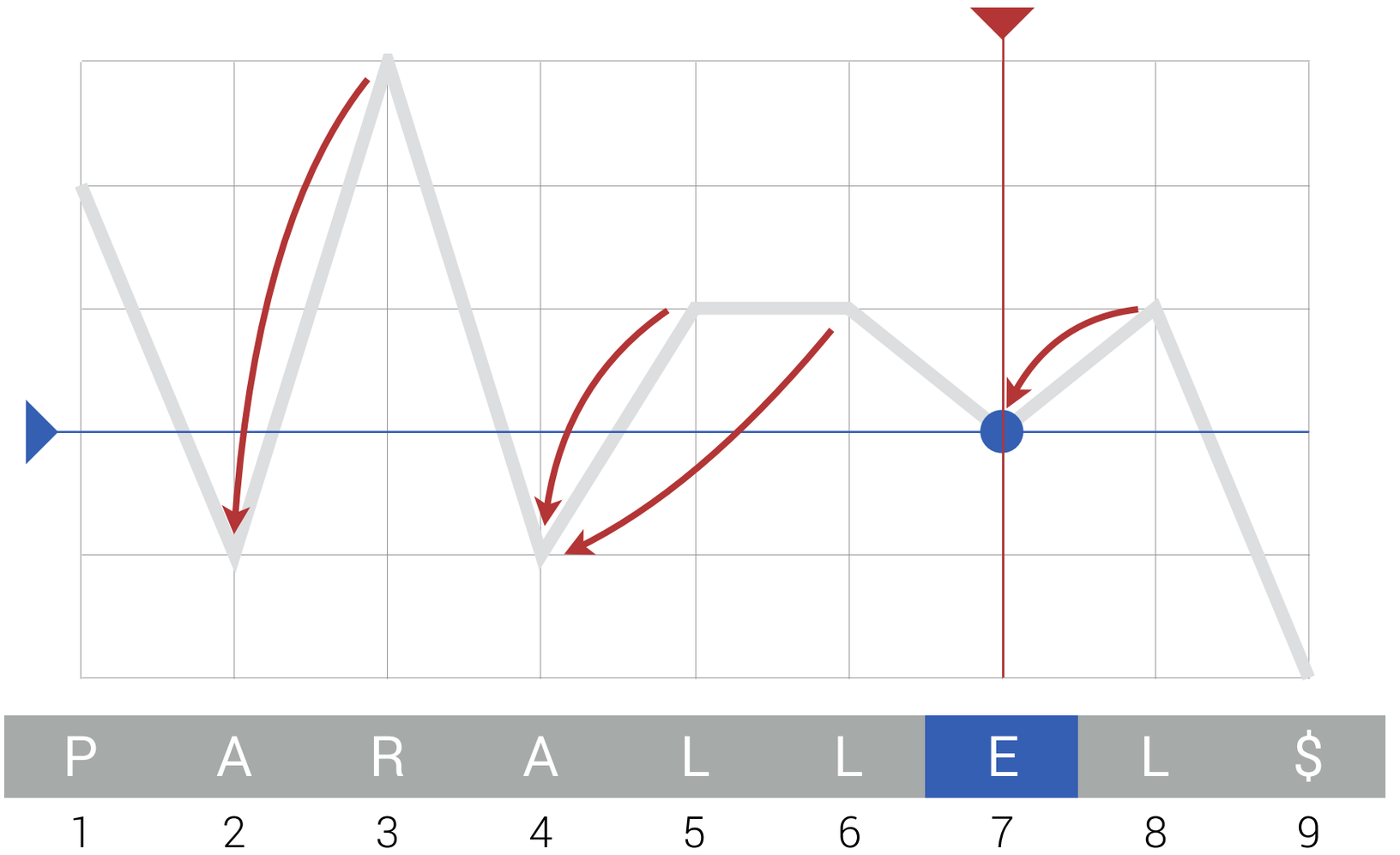
Phase 1

prev(i) := *max* { j ∈ [1 .. i]: Gr.kontext S_j <$_{lex}$ Gr.kontext S_i }



Phase 1

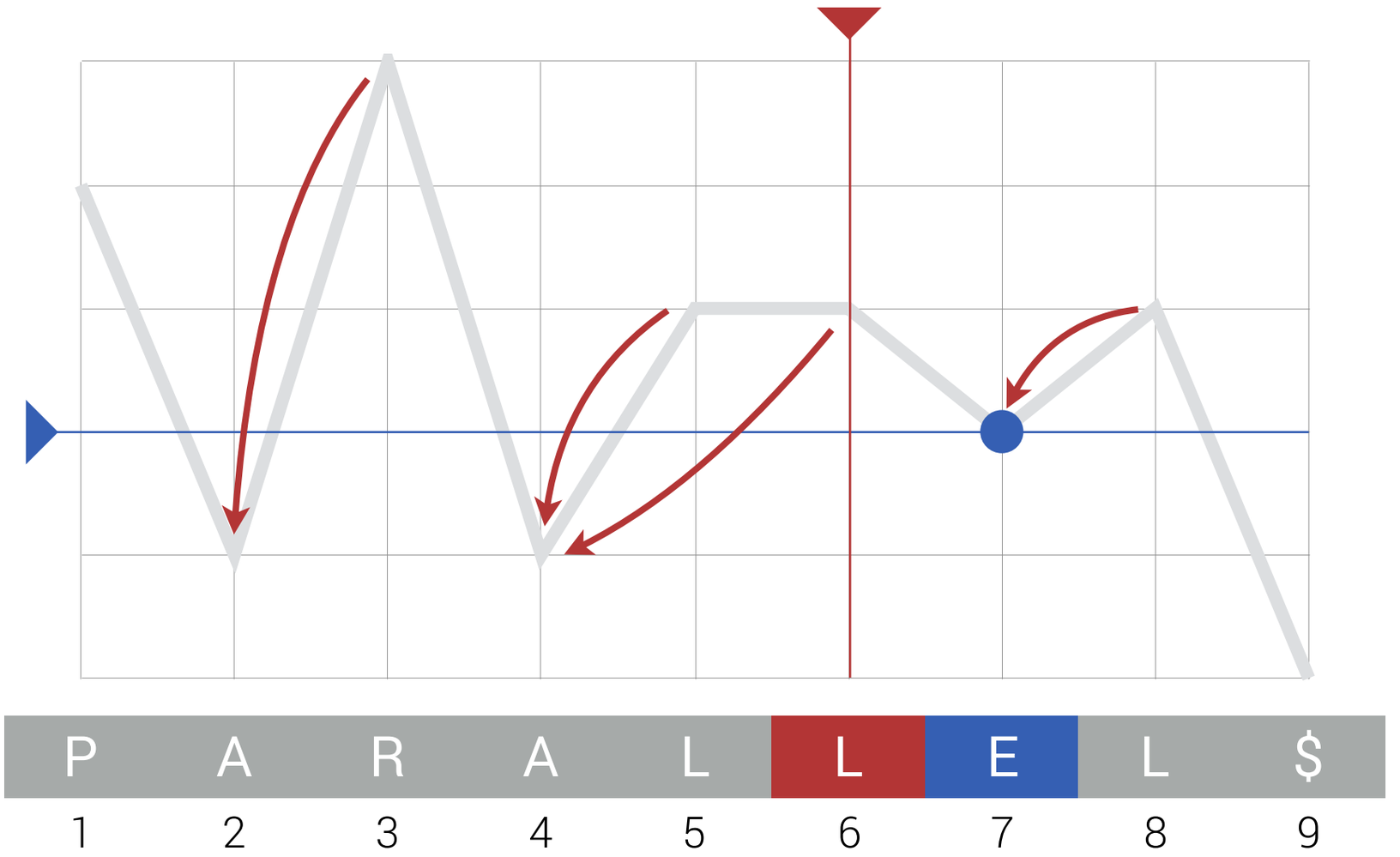
prev(i) := *max* { j ∈ [1 .. i]: Gr.kontext S_j <$_{lex}$ Gr.kontext S_i }



P	A	R	A	L	L	E	L	$
1	2	3	4	5	6	7	8	9

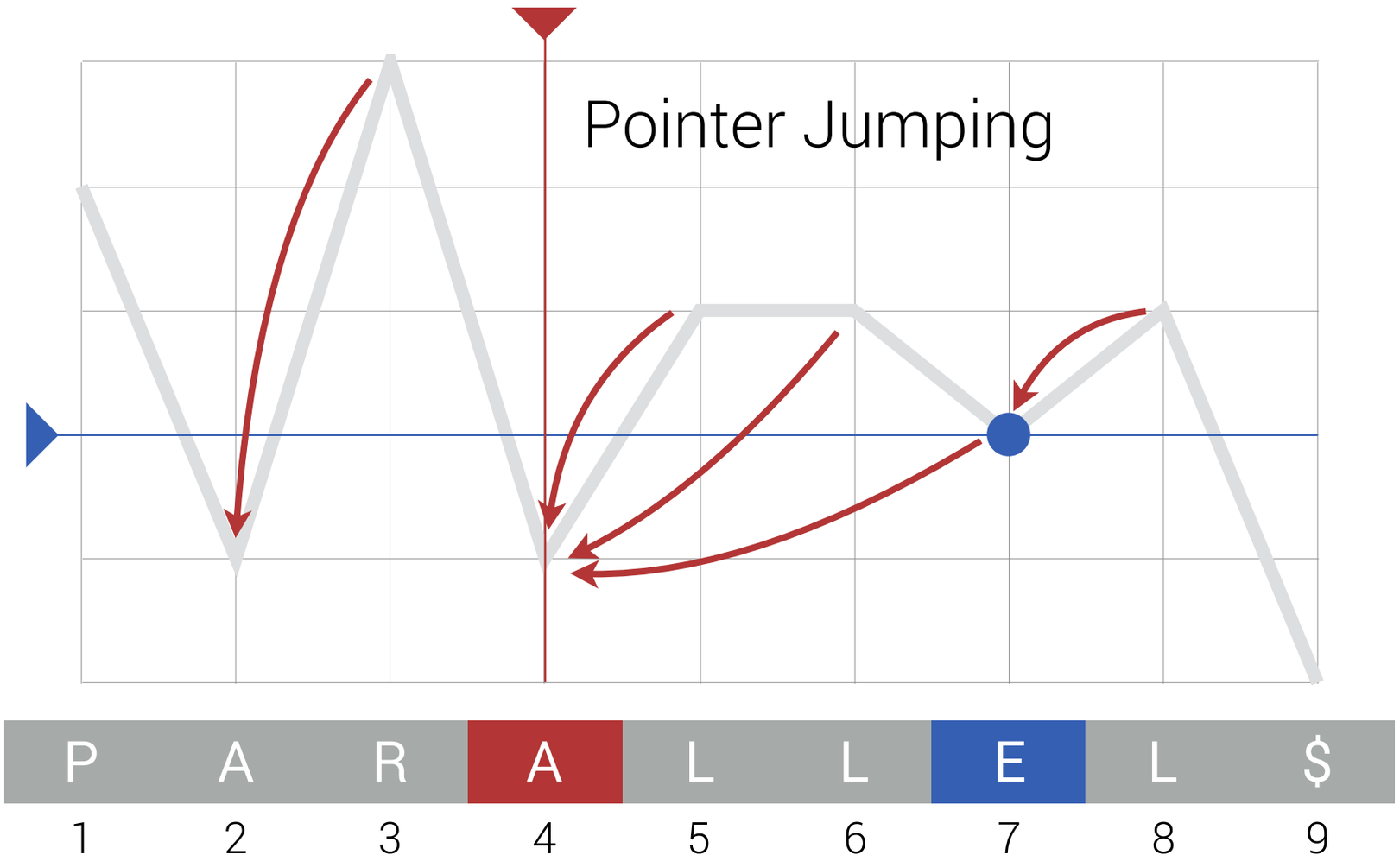
Phase 1

prev(i) := *max* { j ∈ [1 .. i]: Gr.kontext S_j <$_{lex}$ Gr.kontext S_i }



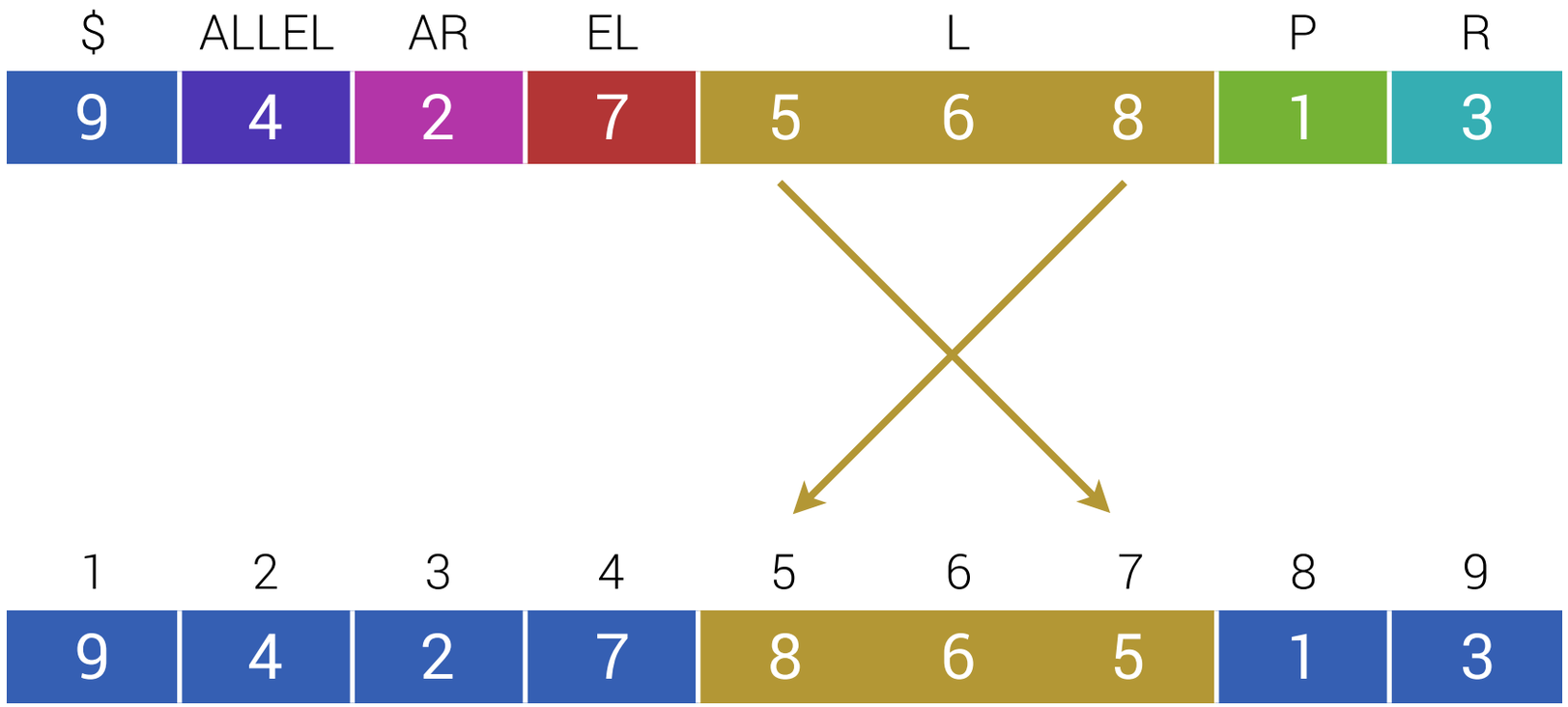
Phase 1

prev(i) := *max* { j ∈ [1 .. i]: Gr.kontext S_j <$_{lex}$ Gr.kontext S_i }



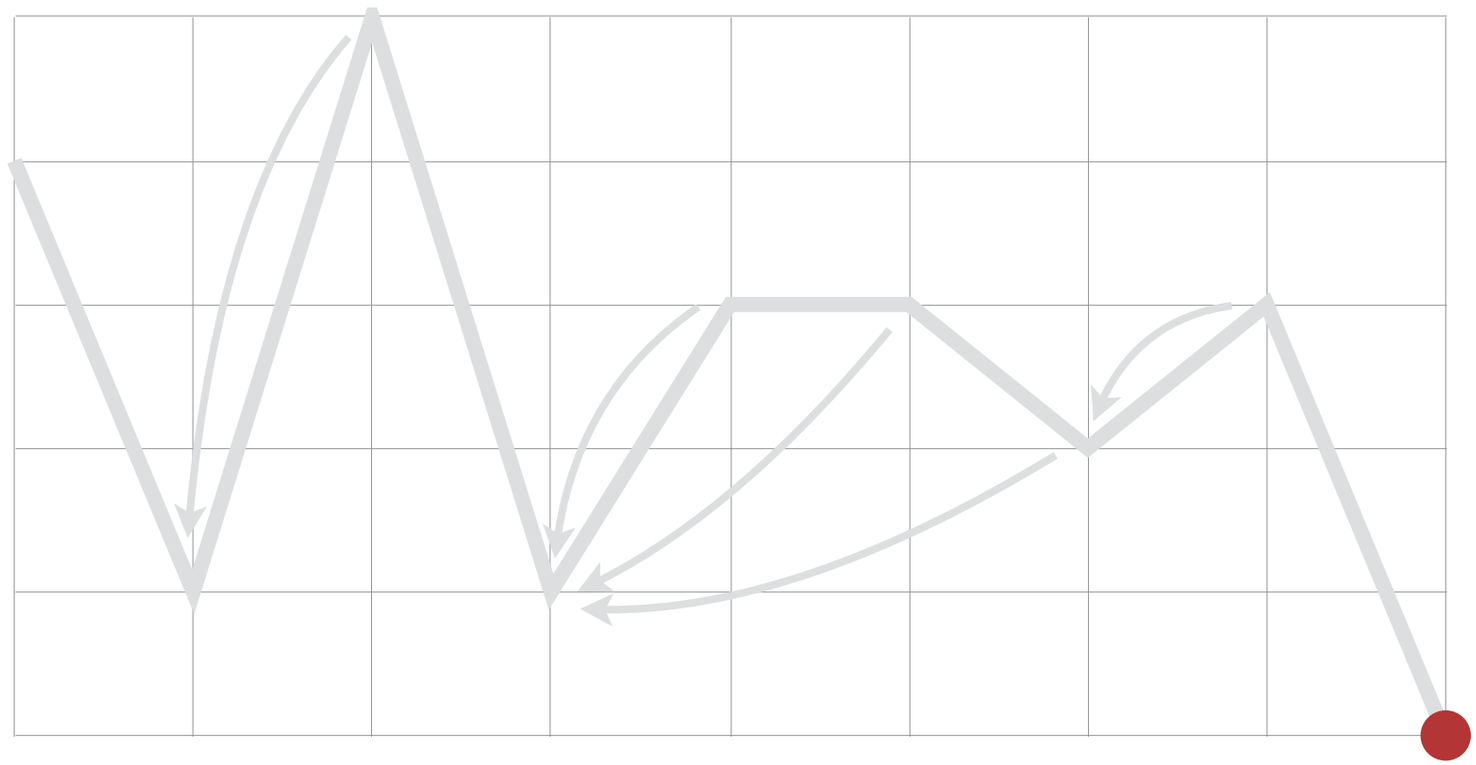
Phase 2

Suffixe innerhalb der Gruppen sortieren



Phase 2

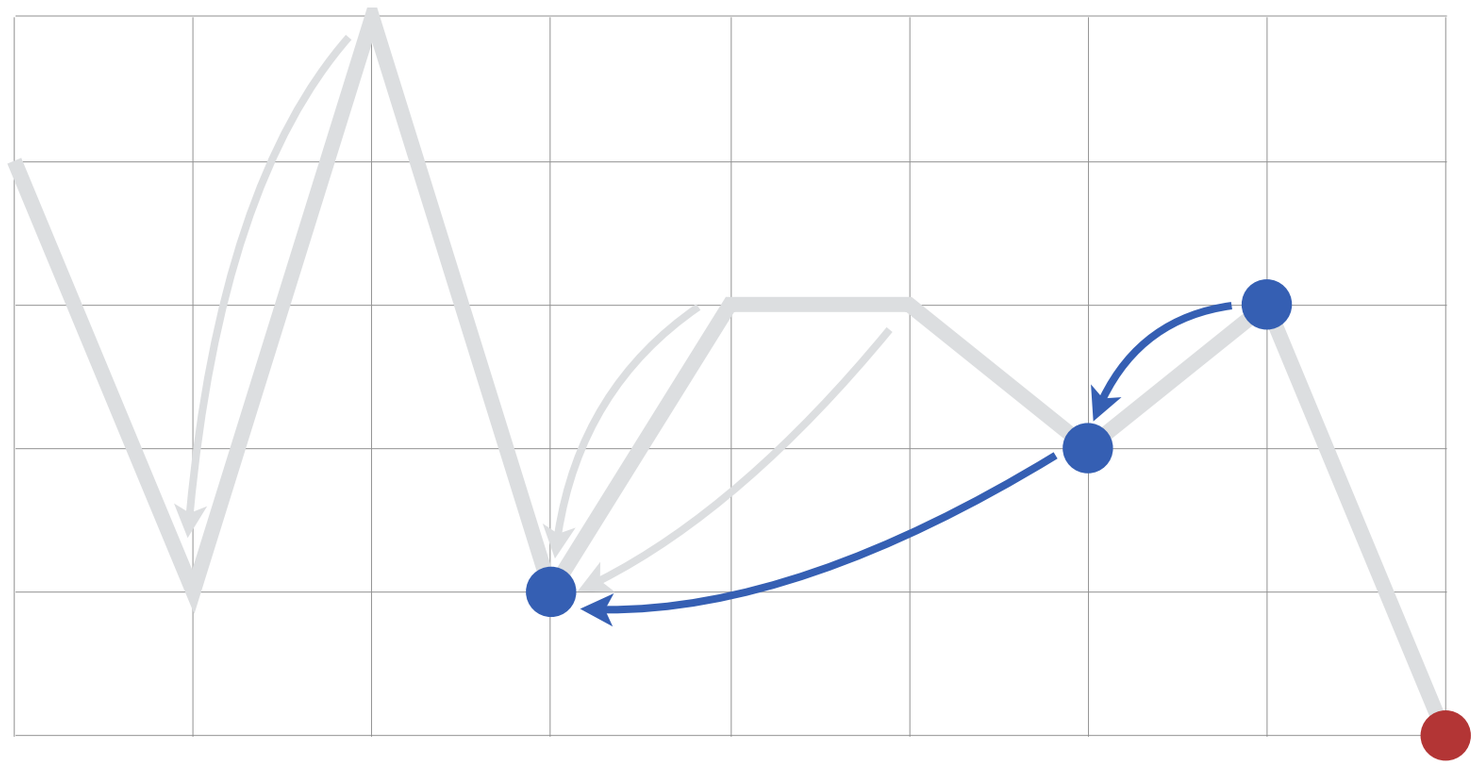
$	ALLEL	AR	EL		L			P	R
9	4	2	7	5	6	8		1	3



P	A	R	A	L	L	E	L	$
1	2	3	4	5	6	7	8	9

Phase 2

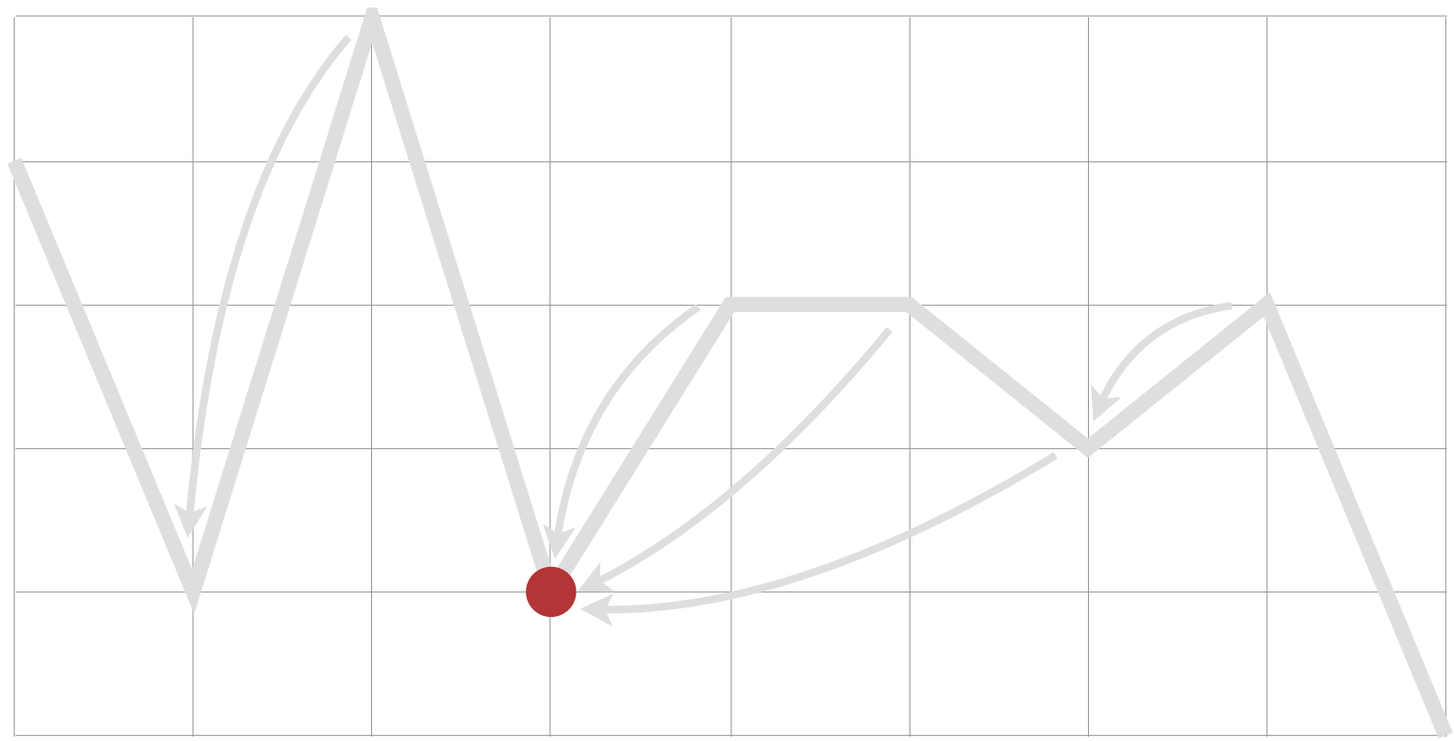
$	ALLEL$	AR	EL$	L$	L		P	R
9	4	2	7	8	5	6	1	3



P	A	R	A	L	L	E	L	$
1	2	3	4	5	6	7	8	9

Phase 2

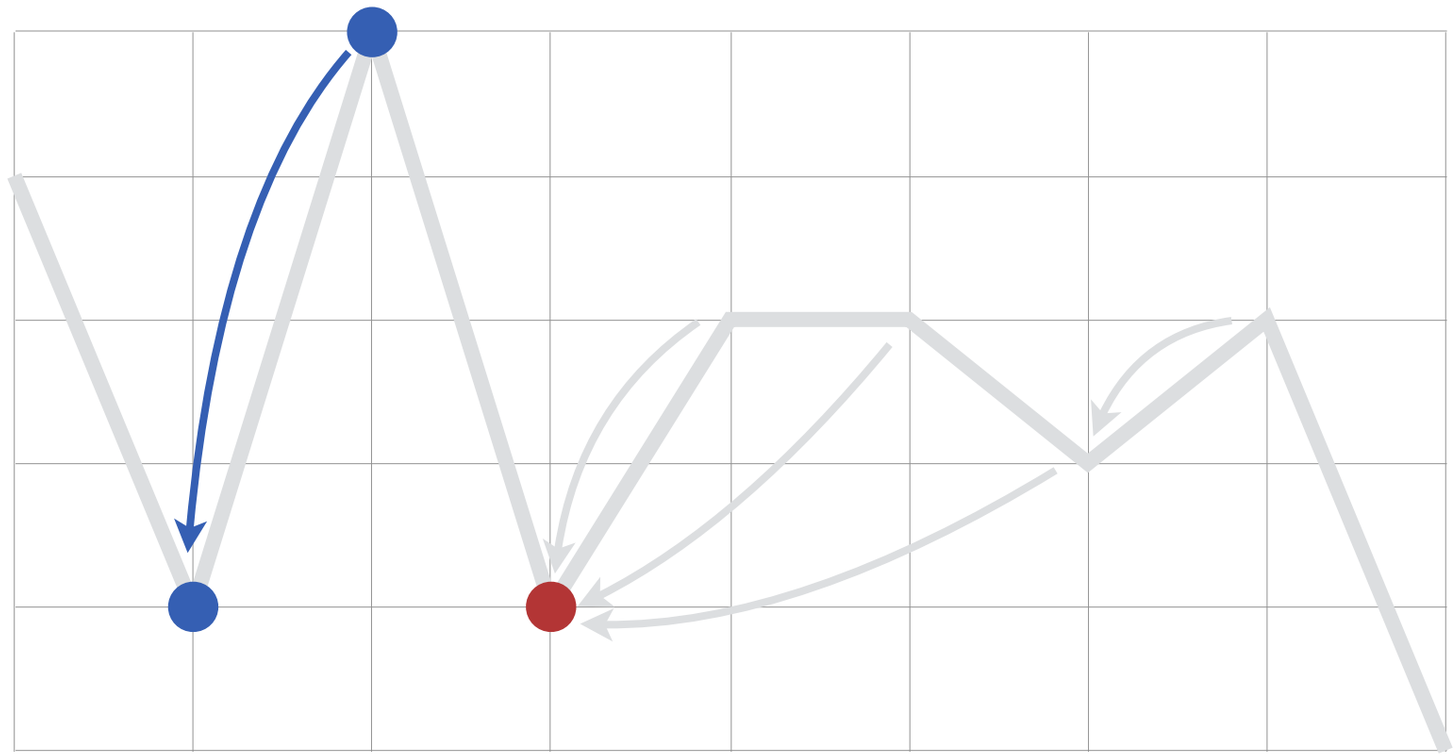
$	ALLEL$	AR	EL$	L$	L		P	R
9	4	2	7	8	5	6	1	3



P	A	R	A	L	L	E	L	$
1	2	3	4	5	6	7	8	9

Phase 2

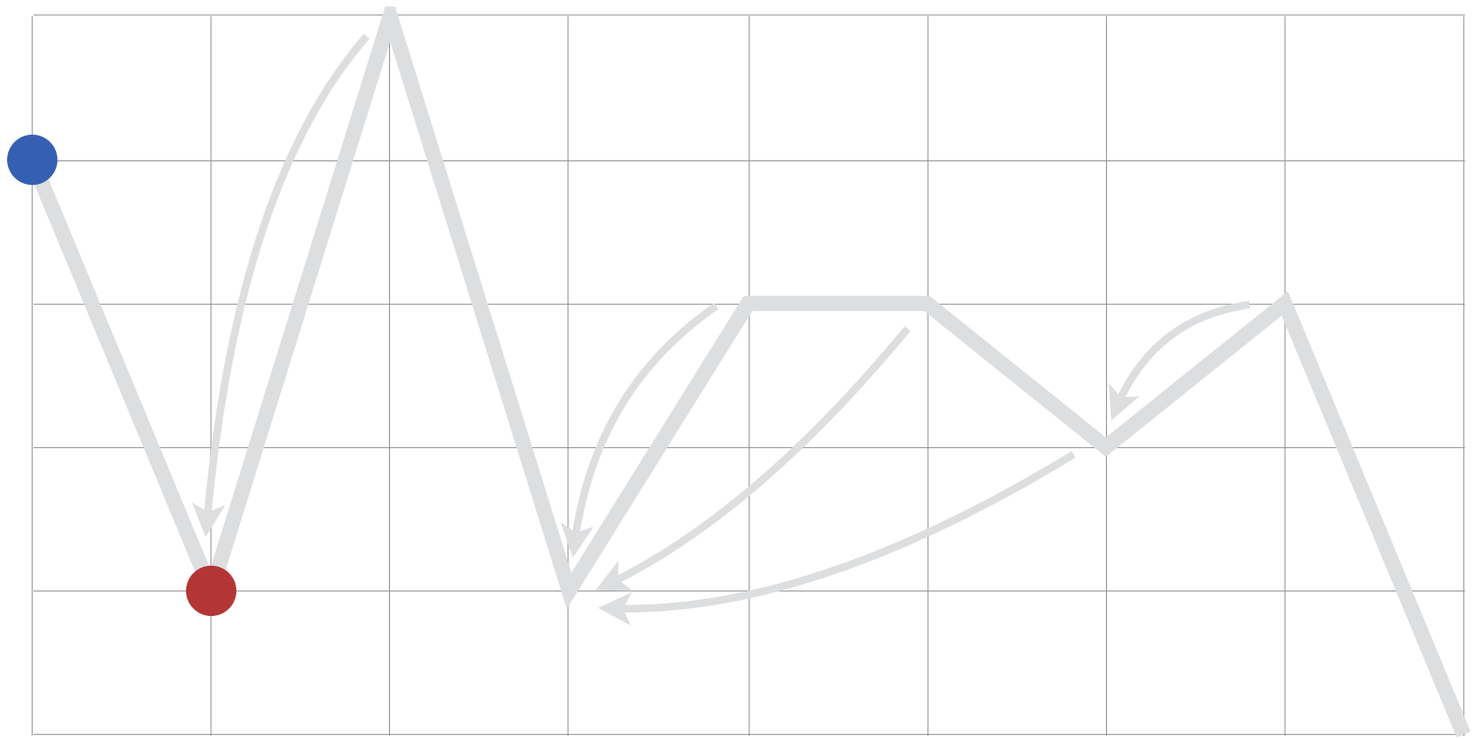
$	ALLEL$	ARALLEL$	EL$	L$	L		P	RALLEL$
9	4	2	7	8	5	6	1	3



P	A	R	A	L	L	E	L	$
1	2	3	4	5	6	7	8	9

Phase 2

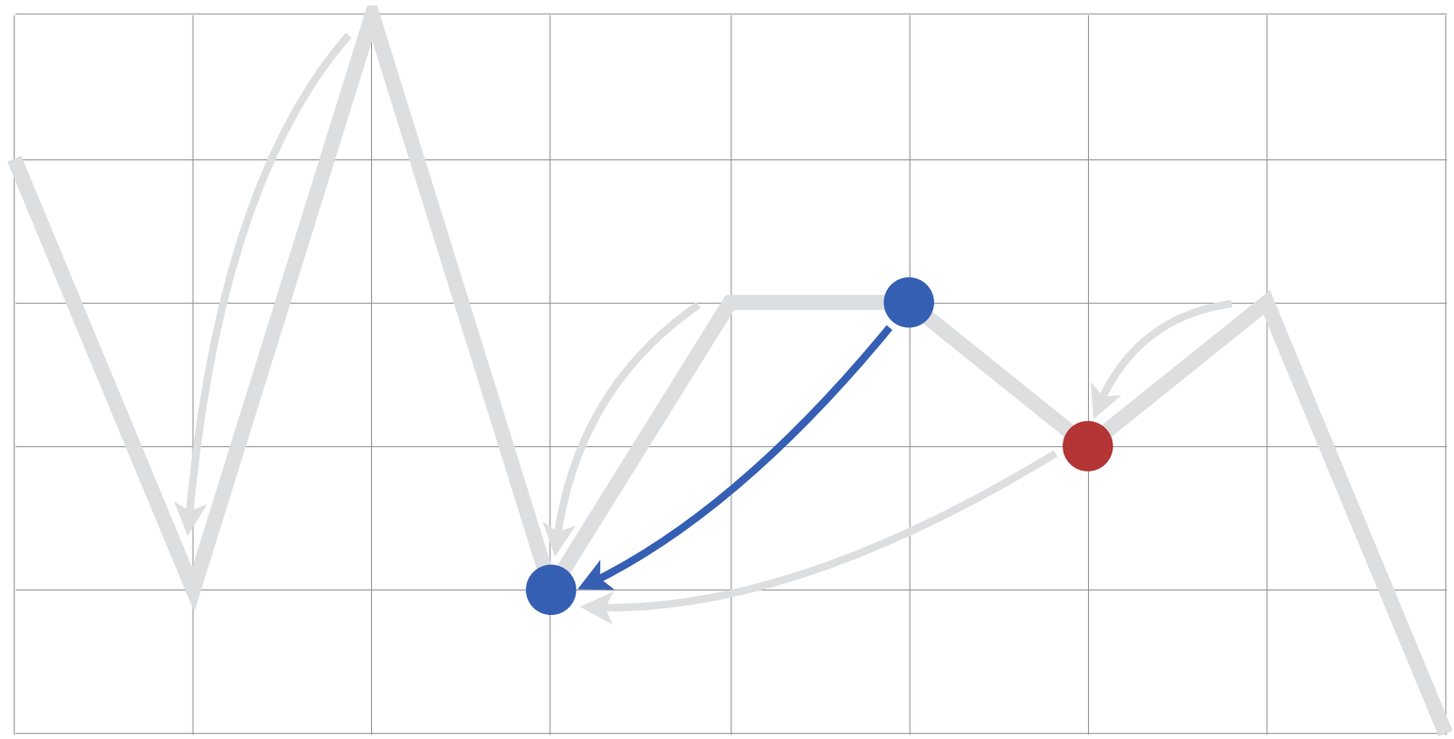
$	ALLEL$	ARALLEL$	EL$	L$	L		PARALLEL$	RALLEL$
9	4	2	7	8	5	6	1	3



P	A	R	A	L	L	E	L	$
1	2	3	4	5	6	7	8	9

Phase 2

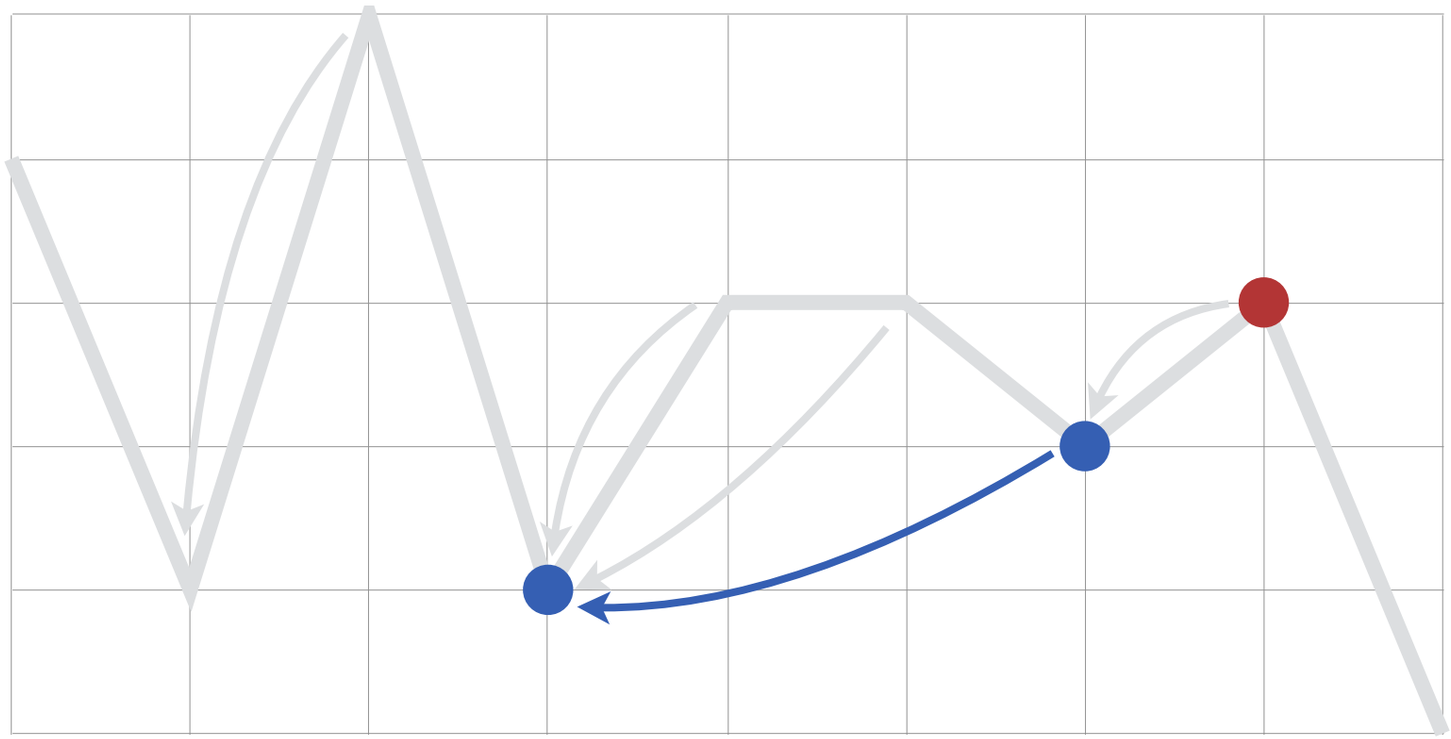
$	ALLEL$	ARALLEL$	EL$	L$	LEL$	L	PARALLEL$	RALLEL$
9	4	2	7	8	6	5	1	3



P	A	R	A	L	L	E	L	$
1	2	3	4	5	6	7	8	9

Phase 2

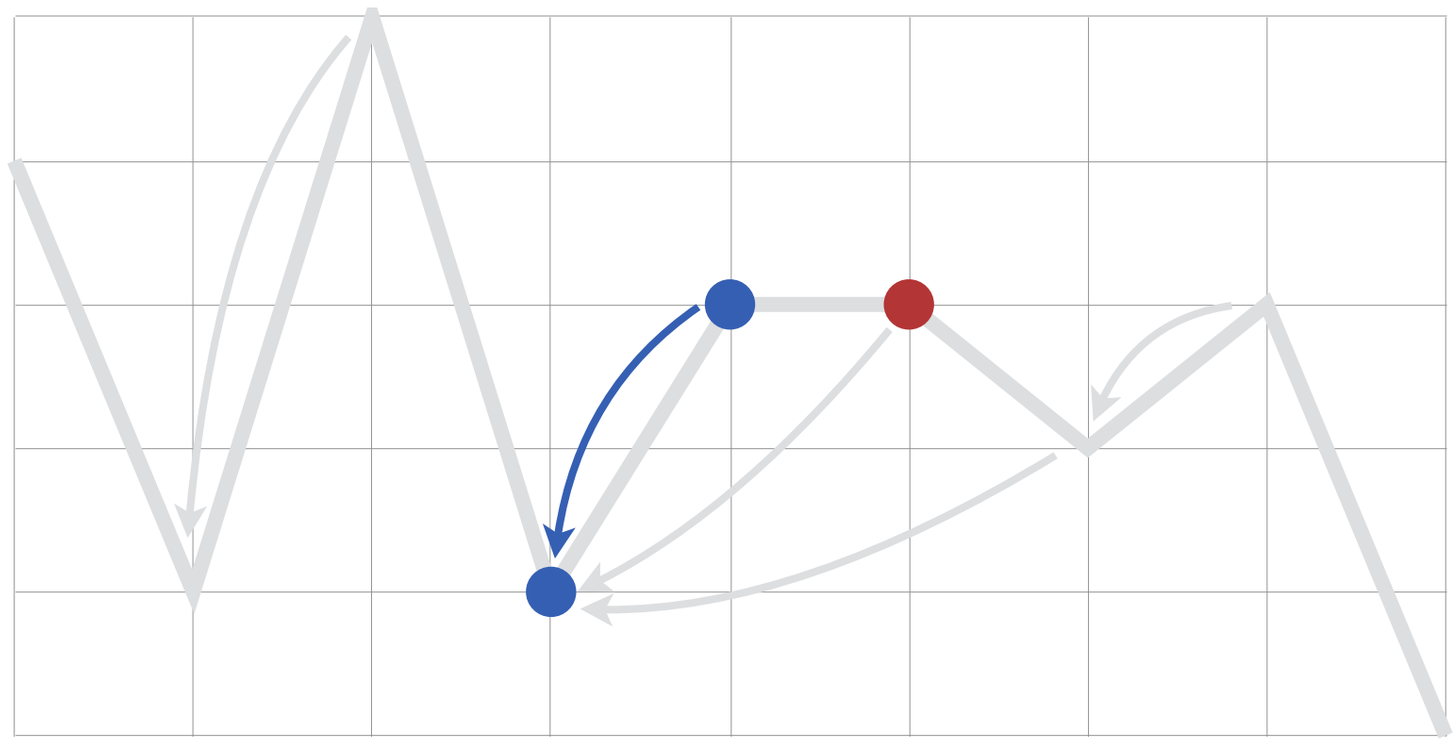
$	ALLEL$	ARALLEL$	EL$	L$	LEL$	L	PARALLEL$	RALLEL$
9	4	2	7	8	6	5	1	3



P	A	R	A	L	L	E	L	$
1	2	3	4	5	6	7	8	9

Phase 2

$	ALLEL$	ARALLEL$	EL$	L$	LEL$	LLEL$	PARALLEL$	RALLEL$
9	4	2	7	8	6	5	1	3



P	A	R	A	L	L	E	L	$
1	2	3	4	5	6	7	8	9

Phase 2

$	ALLEL$	ARALLEL$	EL$	L$	LEL$	LLEL$	PARALLEL$	RALLEL$
SA =	9	4	2	7	8	6	5	1	3

Performance

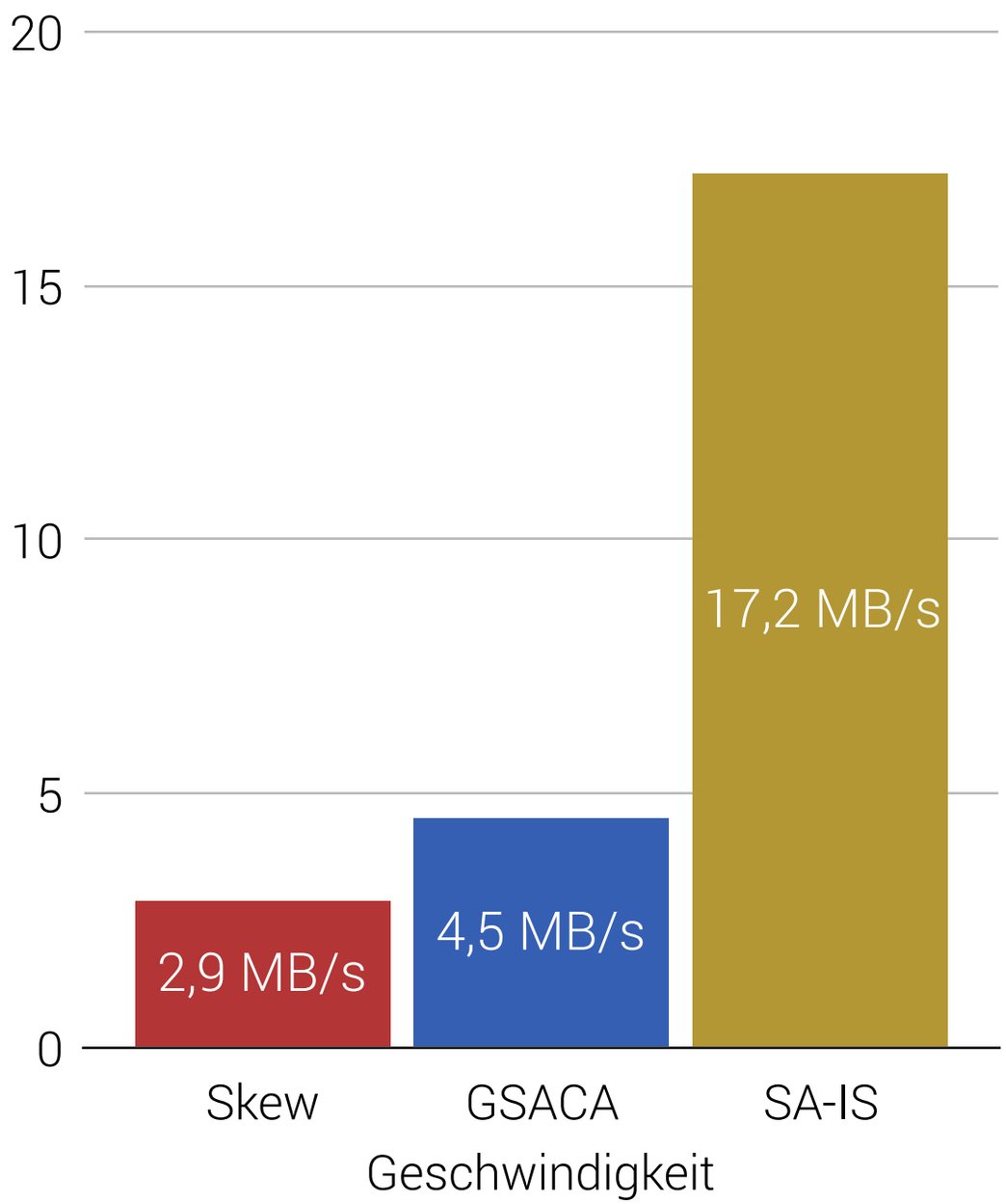
Linearzeit Ansätze

	Skew	SA-IS	GSACA
Art	rekursiv	rekursiv	iterativ
Zeit	$O(n)$	$O(n)$	$O(n)$
Speicher	$O(\log n) + \max 24n$	$O(\log n) + \max 2n$	$O(1) + ?$

Linearzeit Ansätze

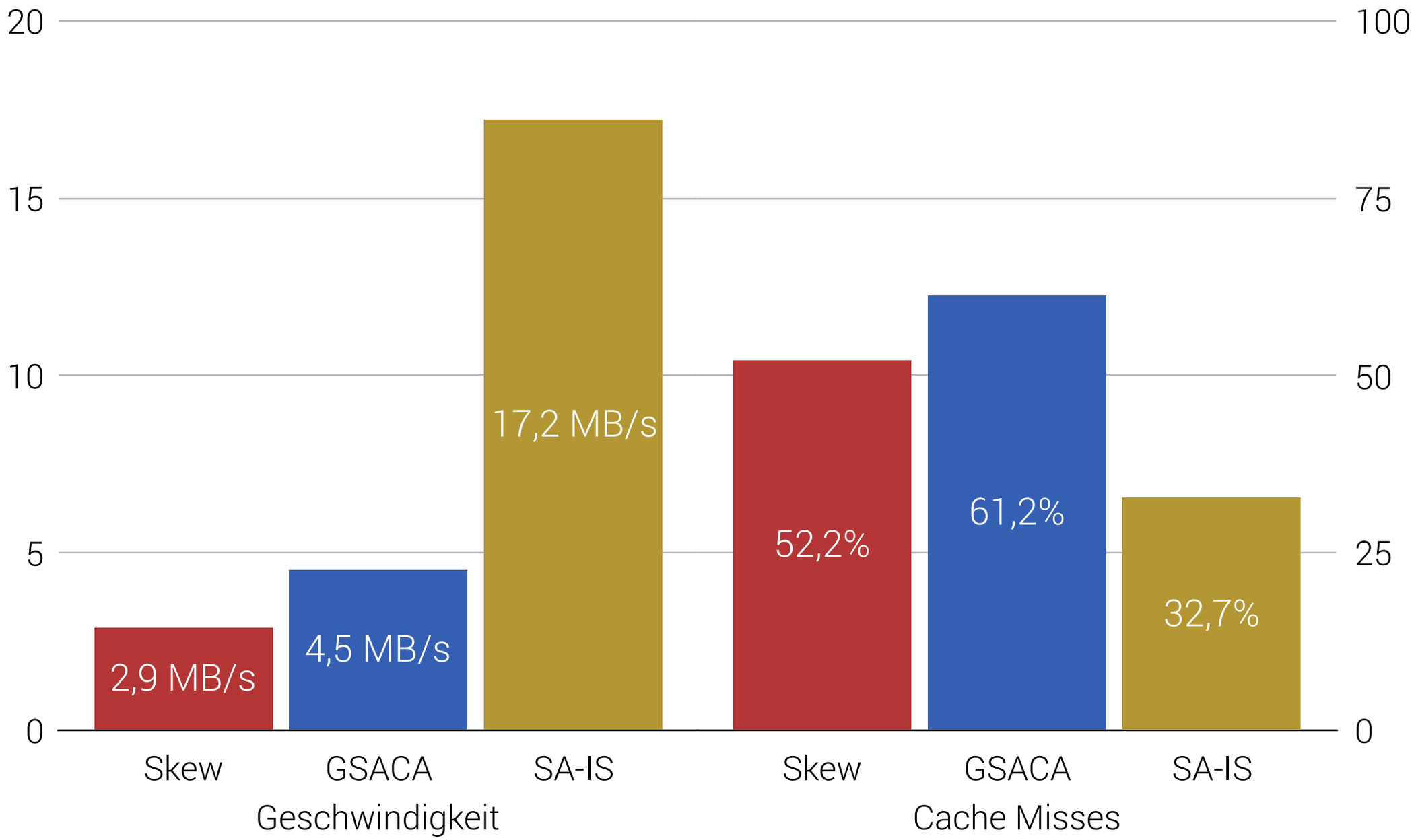
	Skew	SA-IS	GSACA
Art	rekursiv	rekursiv	iterativ
Zeit	$O(n)$	$O(n)$	$O(n)$
Speicher	$O(\log n) + \max 24n$	$O(\log n) + \max 2n$	$O(1) + 12n$

GSACA im Vergleich



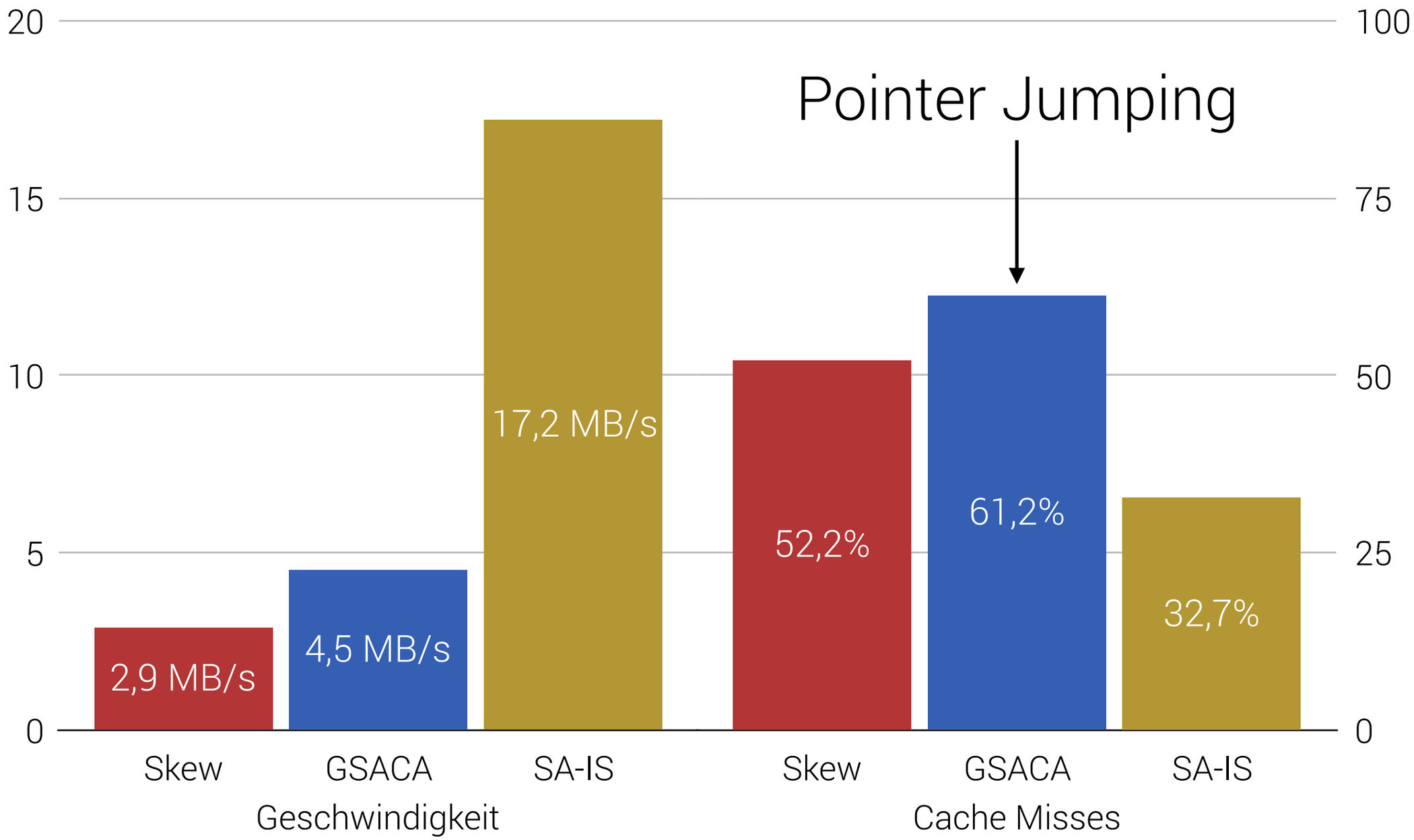
Testdaten: Silesia Corpus

GSACA im Vergleich



Testdaten: Silesia Corpus

GSACA im Vergleich

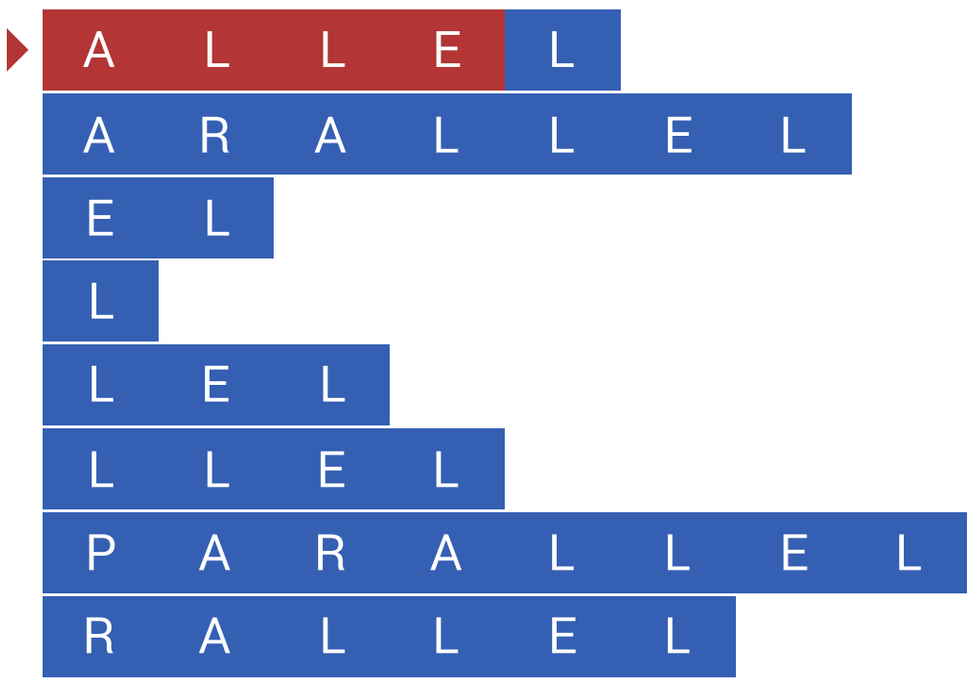


Testdaten: Silesia Corpus

Rückblick

Einsatzgebiete

Substringsuche



LZ77 Kompression

GSACA

Eingabe

Phase 1 Sortierte Folge von Gruppen berechnen $O(n)$

Phase 2 Suffixe innerhalb der Gruppen sortieren $O(n)$

Performance

Performance

Noch nicht praxistauglich.

Performance

~~Noch nicht praxistauglich.~~

Performance

~~Noch nicht praxistauglich.~~

Neuartiges Konzept mit vielen spannenden noch zu lösenden Problemen…

Danke!