Permuted Longest-Common-Prefix Array

Juha Kärkkäinen

University of Helsinki

Joint work with Giovanni Manzini and Simon Puglisi

CPM 2009

Lille, France, June 2009

Outline

1. **Background**
2. Description
3. Construction
4. Compact Representation

Suffix Array and LCP Array

Suffix array (SA)

- Powerful text index
- Linear time construction

Longest-common-prefix (LCP) array

- Often used with suffix array
- Linear time construction from SA [Kasai & al., CPM 2001]

Constructing SA and LCP

SA

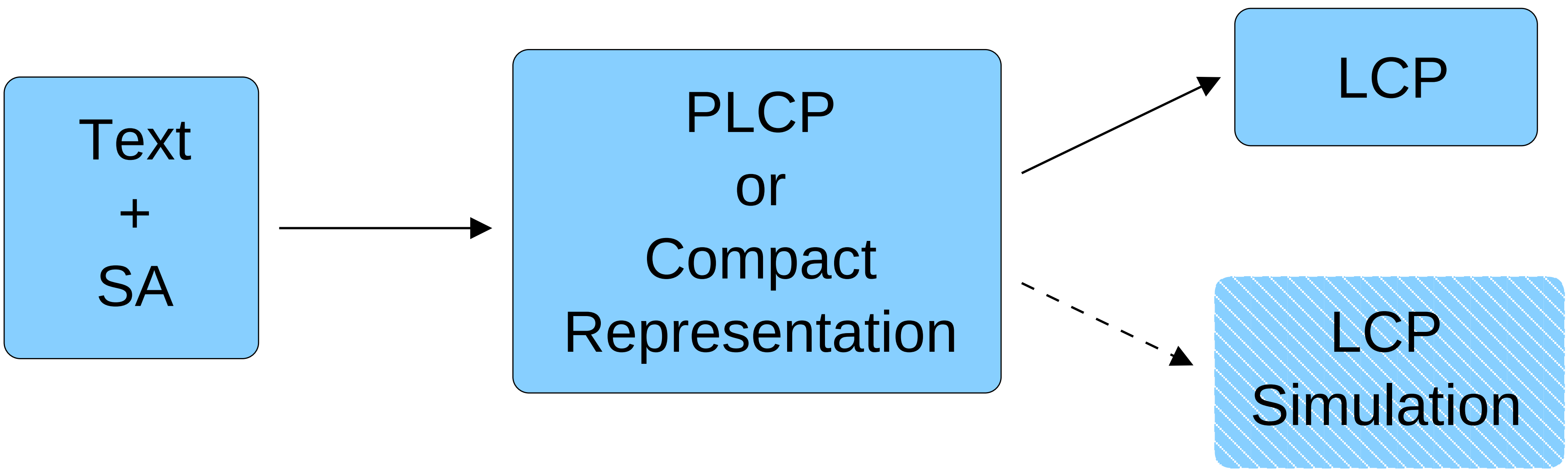
- 5–$6n$ bytes, fastest in practice
- $\sim 2n$ bytes, output to disk sequentially
- external memory

LCP array from SA

- $13n$ bytes, fastest [Kasai & al., CPM 2001]
- $9n$ bytes [Mäkinen, F. Inform. 2003]
- $9n$ or 6–$10n$ bytes [Manzini, SWAT 2004]
- $\sim 2n$ bytes [Puglisi & Turpin, ISAAC 2008]
  - semi-external: sequentially accessed arrays on disk

Permuted LCP (PLCP) Array

- LCP array in permuted order
- LCP construction (or simulation) via PLCP



- Faster and more space-efficient

Constructing LCP from SA

Best previous algorithms

- $13n$ bytes, fast [Kasai & al., CPM 2001]
- $\sim 2n$ bytes, semi-external [Puglisi & Turpin, ISAAC 2008]

New algorithms via PLCP

- semi-external
- $5n$ bytes, $\mathcal{O}(n)$ time, faster
- $11n$ bits, $\mathcal{O}(n \log n)$ time
- $(1 + 4/q)n$ bytes, $\mathcal{O}(nq)$ time

Outline

1. Background
2. **Description**
3. Construction
4. Compact Representation

Suffix Array (SA)

Sorted array of all suffixes

$$T = \begin{matrix} \scriptstyle 0\,1\,2\,3\,4\,5\,6 \\ \text{BANANA\#} \end{matrix}$$

Pos. order

Position	
0	BANANA#
1	ANANA#
2	NANA#
3	ANA#
4	NA#
5	A#
6	#

$\longleftrightarrow$

Lex. order

Suffix array	
6	#
5	A#
3	ANA#
1	ANANA#
0	BANANA#
4	NA#
2	NANA#

LCP Array

Length of longest common prefix with preceding suffix

0 1 2 3 4 5 6

T = BANANA#

Pos. order

0	BANANA#
1	ANANA#
2	NANA#
3	ANA#
4	NA#
5	A#
6	#

$\longleftrightarrow$

Lex. order

-	6	#
0	5	A#
1	3	ANA#
3	1	ANANA#
0	0	BANANA#
0	4	NA#
2	2	NANA#

LCP array SA

PLCP Array

LCP array permuted into the position order

0 1 2 3 4 5 6

T = BANANA#

Pos. order

0	0	BANANA#
3	1	ANANA#
2	2	NANA#
1	3	ANA#
0	4	NA#
0	5	A#
-	6	#

PLCP

$\longleftrightarrow$

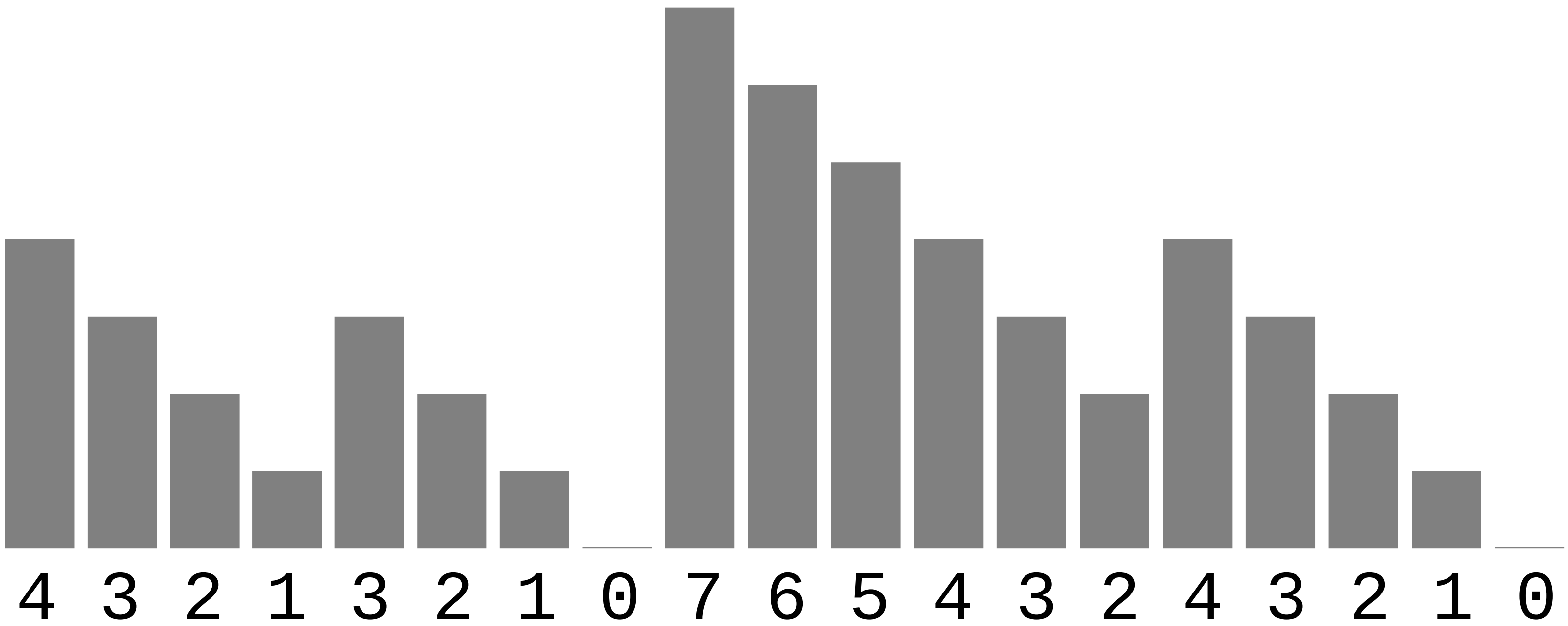
Lex. order

-	6	#
0	5	A#
1	3	ANA#
3	1	ANANA#
0	0	BANANA#
0	4	NA#
2	2	NANA#

LCP

PLCP Array Structure

Theorem. $PLCP[i] \geq PLCP[i-1]-1$.

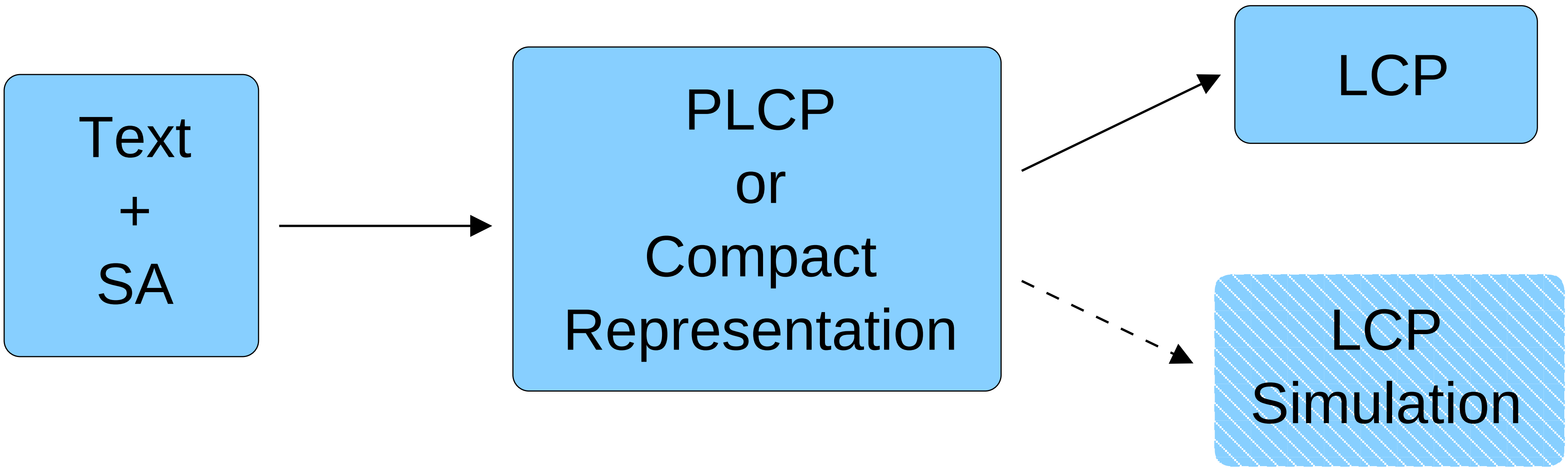


- Used in all efficient LCP-from-SA algorithms
- Enables compact representation

Outline

1. Background
2. Description
3. **Construction**
   - Irreducible LCP Algorithm
   - Φ Algorithm
4. Compact Representation

PLCP Algorithms



- Irreducible LCP Algorithm
- Φ Algorithm

- $LCP[i] = PLCP[SA[i]]$

Irreducible LCP Values

$LCP[i]$ is irreducible $\iff$ $BWT[i] \neq BWT[i-1]$

$$T = \overset{0\,1\,2\,3\,4\,5\,6}{\text{BANANA\#}}$$

Pos. order

PLCP		
0	0	BANANA#
3	1	ANANA#
2	2	NANA#
1	3	ANA#
0	4	NA#
0	5	A#
-	6	#

$\longleftrightarrow$

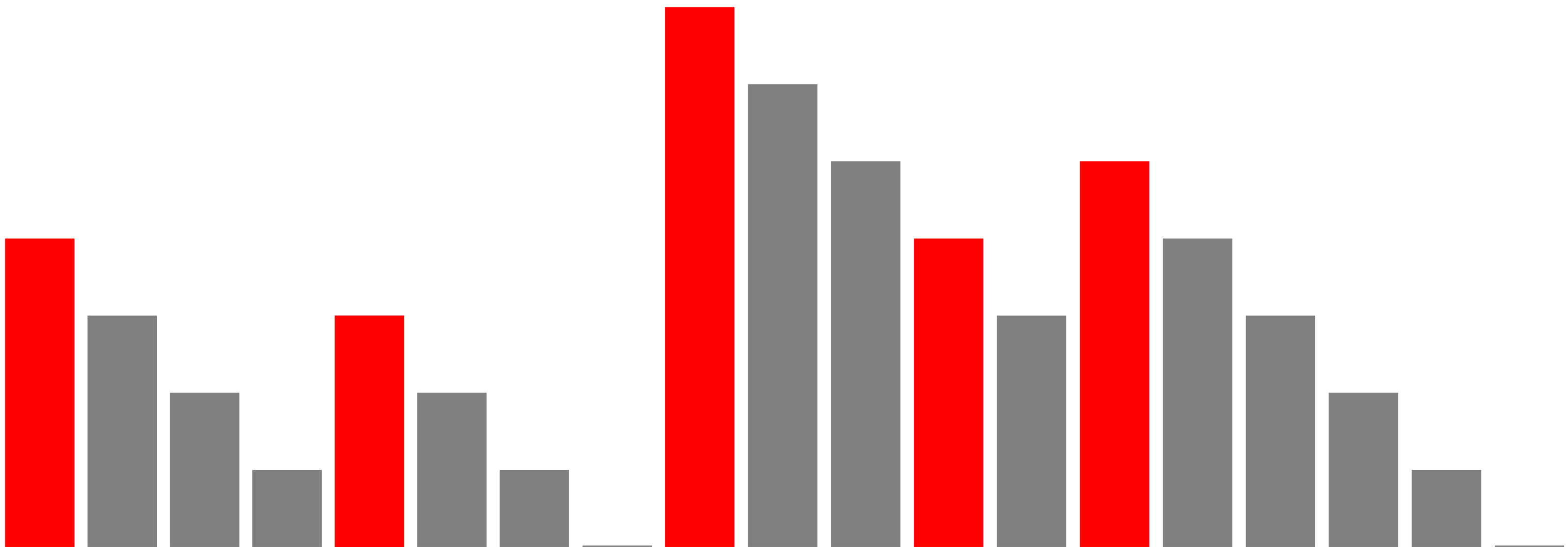
Lex. order

LCP	BWT		
-	A	6	#
0	N	5	A#
1	N	3	ANA#
3	B	1	ANANA#
0	#	0	BANANA#
0	A	4	NA#
2	A	2	NANA#

Irreducible LCP Values

Theorem. The sum of **irreducible** LCP values is $\leq 2n \log n$

Lemma. For **reducible** LCP value, [Manzini, SWAT 2004]
$PLCP[i] = PLCP[i-1] - 1$



Irreducible LCP Algorithm

1. Compute all irreducible PLCP values naively
2. Compute reducible PLCP values:
   $PLCP[i] := PLCP[i-1] - 1$
3. Compute LCP from PLCP

- $\mathcal{O}(n \log n)$ time
  - Faster than previous algorithms
- $5n$ bytes
  - semi-external $\implies$ 10% slower

Φ *Algorithm*

1. Compute Φ: **for** $i := 1$ **to** n **do** $\Phi[SA[i]] := SA[i-1]$
2. Compute PLCP: $\ell := 0$
   for $i := 0$ **to** $n-1$ **do**
   while $T[i+\ell] = T[\Phi[i]+\ell]$ **do ++**ℓ
   $PLCP[i] := \ell$; $\ell := \max(\ell - 1, 0)$
3. Compute LCP: **for** $i := 1$ **to** n **do** $LCP[i] := PLCP[SA[i]]$

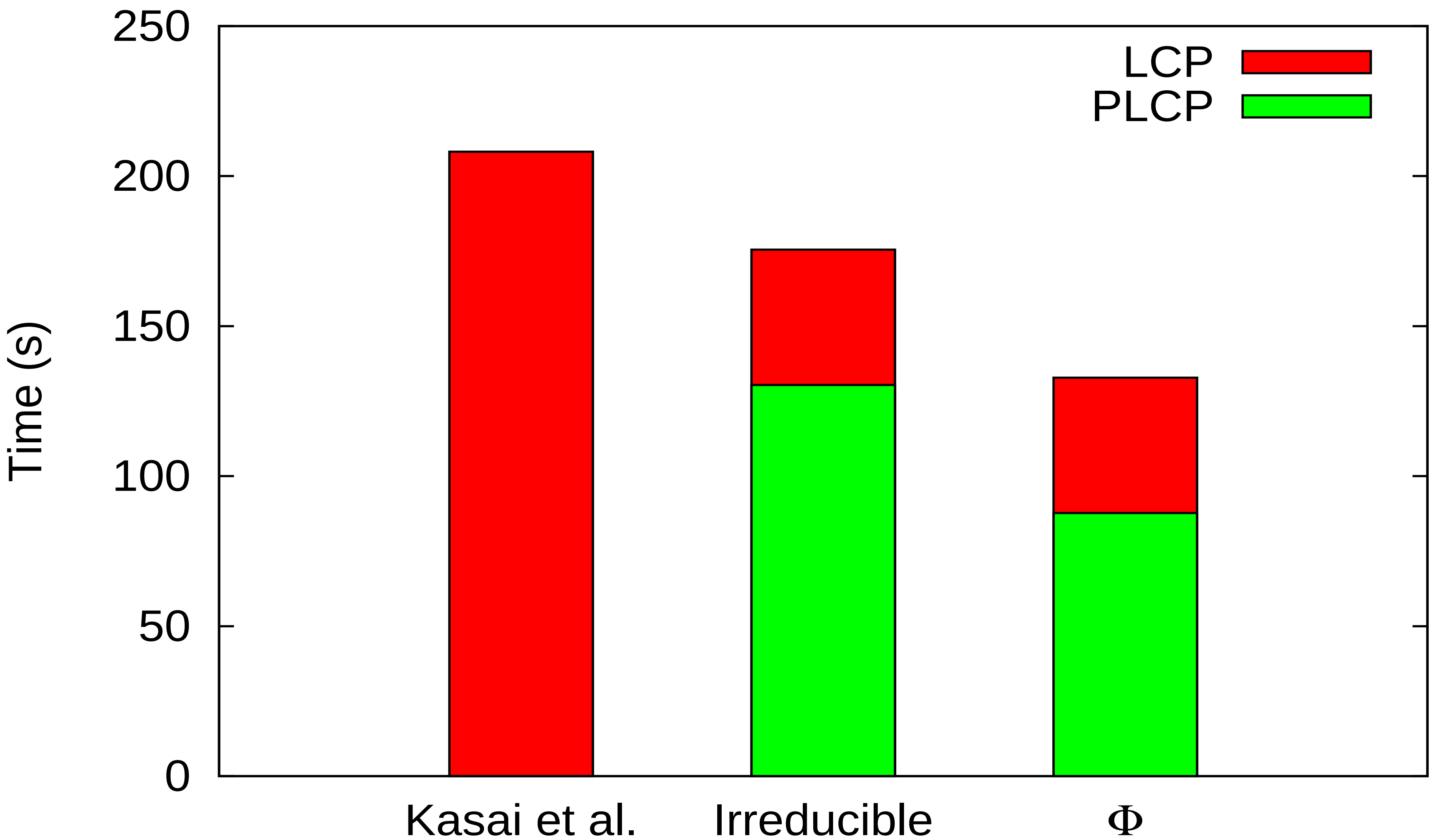
Over 30% faster than any previous algorithm

- Only one random access in each round of main loop

$5n$ bytes

- semi-external $\Longrightarrow$ 10% slower

Time for Manzini Corpus (708 MiB)

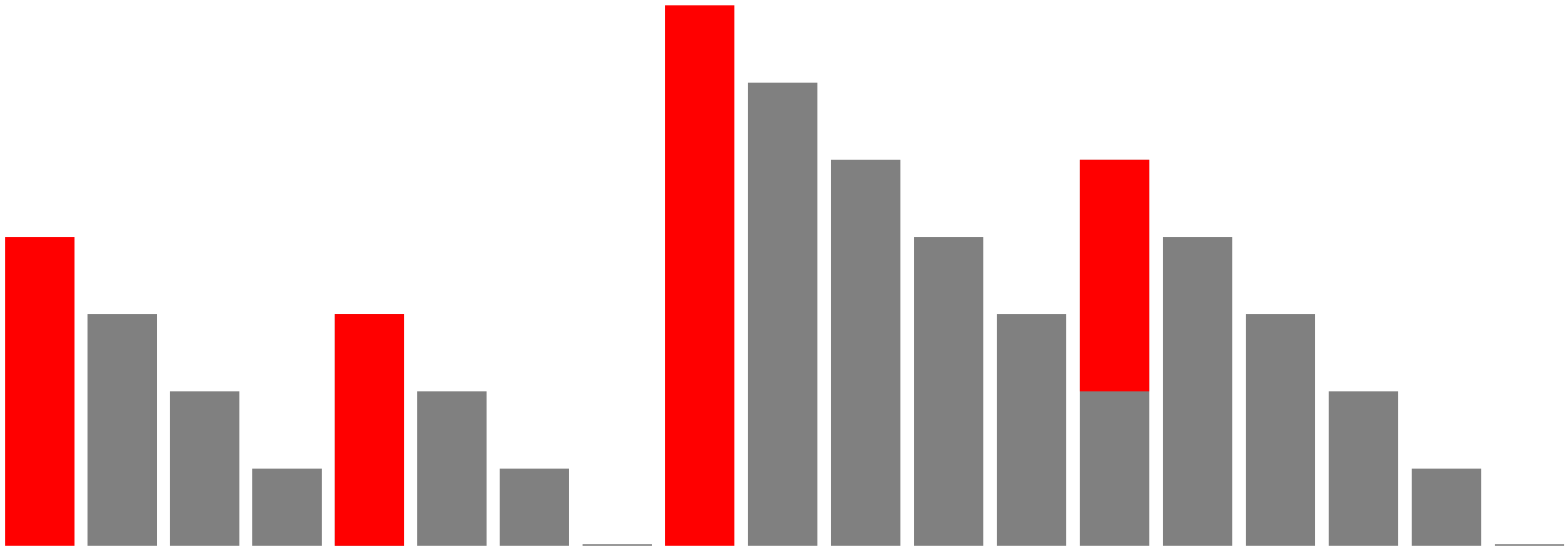


Outline

1. Background
2. Description
3. Construction
4. **Compact Representation**
   - Succinct Bitvector
   - Sparse PLCP Array

Succinct Bitvector Representation

PLCP stored in bitvector B of $2n$ bits [Sadakane, SODA 2002]



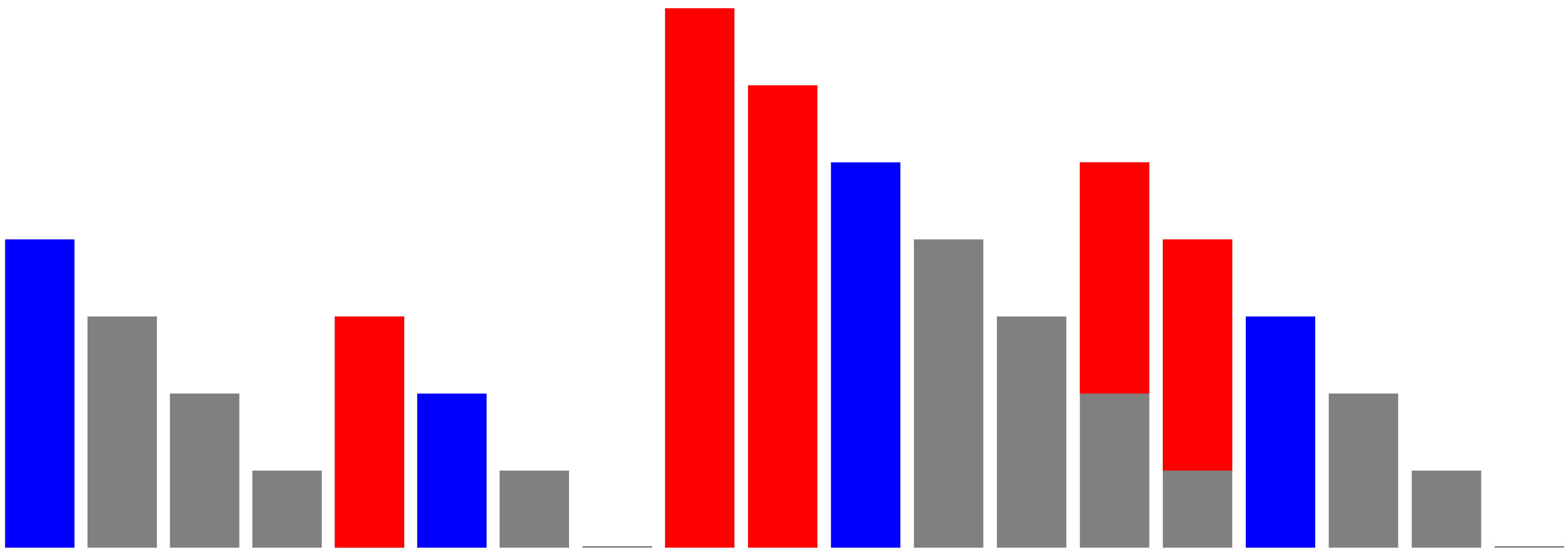
B = 00001111000111100000000111110001111

Irreducible LCP Algorithm

- $\mathcal{O}(n \log n)$ time
- $11n$ bits (semi-external)

Sparse PLCP array

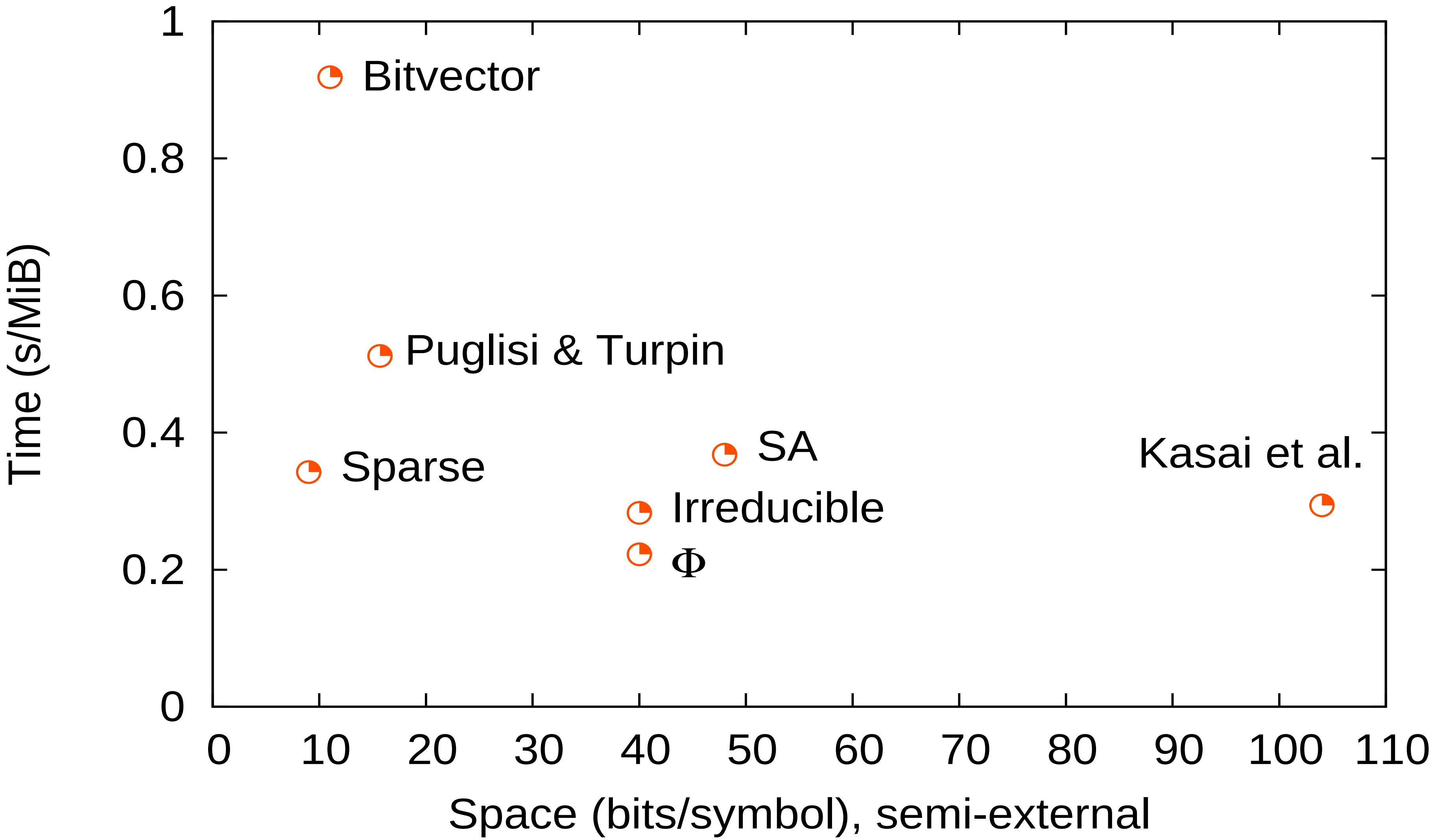
Store every qth entry of PLCP array



Φ Algorithm

- $\mathcal{O}(qn)$ time
- $(1 + 4/q)n$ bytes (semi-external)

Space and Time for Manzini Corpus (708 MiB)



Conclusion

Summary

- Faster and more space-efficient algorithms for (P)LCP array construction
- Irreducible LCP Theorem

Open problem

- External memory algorithm for LCP-from-SA construction

Dmitry Khmelev

- Died at young age in 2004
- Mathematician doing stringology as hobby
- Conjectured the Irreducible LCP Theorem
- LCP construction algorithm
  - Implementation, no publication
  - Uses sparse PLCP array
  - Analysis relies on Irreducible LCP Theorem