Engineering the Computation of Minimal Absent Words

Carl Barton[1], Alice Heliou[2,3],
Laurent Mouchard[4] and Solon P. Pissis[5]

[1] The Blizard Institute, Barts and The London School of Medicine and Dentistry, Queen Mary University of London, UK
[2] Inria Saclay-Île de France, AMIB, Bâtiment Alan Turing, France
[3] Laboratoire d'Informatique de l'École Polytechnique (LIX), CNRS UMR 7161, France
[4] University of Rouen, LITIS EA 4108, TIBS, Rouen, France
[5] Department of Informatics, King's College London, London, UK

PPAM, 7^{th} September 2015

Table of Contents

$$y = \overset{0}{A}\,\overset{1}{A}\,\overset{2}{C}\,\overset{3}{A}\,\overset{4}{C}\,\overset{5}{A}\,\overset{6}{C}\,\overset{7}{C}$$

Proper Factor

A proper factor of a word y is a factor of y that is neither the empty word nor y itself.

$$y = \overset{0}{A}\ \overset{1}{A}\ \overset{2}{C}\ \overset{3}{A}\ \overset{4}{C}\ \overset{5}{A}\ \overset{6}{C}\ \overset{7}{C}$$

Proper Factor

A proper factor of a word y is a factor of y that is neither the empty word nor y itself.

$$y = \overset{0}{A}\,\overset{1}{A}\,\overset{2}{C}\,\overset{3}{A}\,\overset{4}{C}\,\overset{5}{A}\,\overset{6}{C}\,\overset{7}{C}$$

Proper Factor

A proper factor of a word y is a factor of y that is neither the empty word nor y itself.

$$\begin{array}{c} 0\ 1\ 2\ 3\ 4\ 5\ 6\ 7 \\ y = \mathrm{A\,A\,C\,A\,C\,A\,C\,C} \end{array}$$

Proper Factor

A proper factor of a word y is a factor of y that is neither the empty word nor y itself.

$$y = \overset{0}{A}\,\overset{1}{A}\,\overset{2}{C}\,\overset{3}{A}\,\overset{4}{C}\,\overset{5}{A}\,\overset{6}{C}\,\overset{7}{C}$$

Proper Factor

A proper factor of a word y is a factor of y that is neither the empty word nor y itself.

Minimal Absent Word (MAW)

A **word** is **absent** of y if it doesn't occur in y.
An **absent word** is **minimal** if all its proper factors occur in y.
The number of minimal absent words is **upper bounded** by $\mathcal{O}(\sigma n)$
Crochemore et al. 1998 and Mignosi et al. 2002

$$y = \overset{0}{A}\,\overset{1}{A}\,\overset{2}{C}\,\overset{3}{A}\,\overset{4}{C}\,\overset{5}{A}\,\overset{6}{C}\,\overset{7}{C}$$

AAA, AACACC, AACC, CAA, CACACA, CCA, CCC

Proper Factor

A proper factor of a word y is a factor of y that is neither the empty word nor y itself.

Minimal Absent Word (MAW)

A **word** is **absent** of y if it doesn't occur in y.
An **absent word** is **minimal** if all its proper factors occur in y.
The number of minimal absent words is **upper bounded** by $\mathcal{O}(\sigma n)$
Crochemore et al. 1998 and Mignosi et al. 2002

$$y = \overset{0}{A}\overset{1}{A}\overset{2}{C}\overset{3}{A}\overset{4}{C}\overset{5}{A}\overset{6}{C}\overset{7}{C}$$

AAA, AACACC, AACC, CAA, CACACA, CCA, CCC

Proper Factor

A proper factor of a word y is a factor of y that is neither the empty word nor y itself.

Minimal Absent Word (MAW)

A **word** is **absent** of y if it doesn't occur in y.
An **absent word** is **minimal** if all its proper factors occur in y.
The number of minimal absent words is **upper bounded** by $\mathcal{O}(\sigma n)$
Crochemore et al. 1998 and Mignosi et al. 2002

$$\overset{0}{A}\overset{1}{A}\overset{2}{C}\overset{3}{A}\overset{4}{C}\overset{5}{A}\overset{6}{C}\overset{7}{C}$$

$y = AACACACC$

AAA, AACACC, AACC, CAA, CACACA, CCA, CCC

Proper Factor

A proper factor of a word y is a factor of y that is neither the empty word nor y itself.

Minimal Absent Word (MAW)

A **word** is **absent** of y if it doesn't occur in y.

An **absent word** is **minimal** if all its proper factors occur in y.

The number of minimal absent words is **upper bounded** by $\mathcal{O}(\sigma n)$

Crochemore et al. 1998 and Mignosi et al. 2002

$$y = \overset{0}{A}\overset{1}{A}\overset{2}{C}\overset{3}{A}\overset{4}{C}\overset{5}{A}\overset{6}{C}\overset{7}{C}$$

AAA, AACACC, AACC, CAA, CACACA, CCA, CCC

Proper Factor

A proper factor of a word y is a factor of y that is neither the empty word nor y itself.

Minimal Absent Word (MAW)

A **word** is **absent** of y if it doesn't occur in y.
An **absent word** is **minimal** if all its proper factors occur in y.
The number of minimal absent words is **upper bounded** by $\mathcal{O}(\sigma n)$
Crochemore et al. 1998 and Mignosi et al. 2002

$$\overset{0}{A}\overset{1}{A}\overset{2}{C}\overset{3}{A}\overset{4}{C}\overset{5}{A}\overset{6}{C}\overset{7}{C}$$

$y = AACACACC$

AAA, AACACC, AACC, CAA, CACACA, CCA, CCC

Proper Factor

A proper factor of a word y is a factor of y that is neither the empty word nor y itself.

Minimal Absent Word (MAW)

A **word** is **absent** of y if it doesn't occur in y.

An **absent word** is **minimal** if all its proper factors occur in y.

The number of minimal absent words is **upper bounded** by $\mathcal{O}(\sigma n)$

Crochemore et al. 1998 and Mignosi et al. 2002

Context

Biology

- Linear-Time Sequence Comparison Using Minimal Absent Words & Applications, [Crochemore et al.], 2015
- Three minimal sequences found in Ebola virus genomes and absent from human DNA, [Silva et al.], 2015
- Minimal Absent Words in Prokaryotic and Eukaryotic Genomes, [Garcia et al.], 2011

Computer Science

- Data Compression Using Antidictionaries, [Crochemore et al.], 2000, [Fiala and Holub], 2008

State of the Art

References	Time for fixed size alphabet	Space	Structure
Crochemore et al. 1998 Automata and forbidden words	$\mathcal{O}(n)$	$\mathcal{O}(n)$	suffix automata
Pinho et al. 2009 On finding minimal absent words	$\mathcal{O}(n^2)$	$\mathcal{O}(n)$	suffix array
Belazzougui et al. 2013 Versatile Succinct Representations of the Bidirectional Burrows-Wheeler Transform.	$\mathcal{O}(n)$	$\mathcal{O}(n)$	compact bidirectional BWT
Barton et al. 2014 Linear-time computation of minimal absent words using suffix array.	$\mathcal{O}(n)$	$\mathcal{O}(n)$	suffix array
Belazzougui et al. 2015 Space-efficient detection of unusual words.	randomized $\mathcal{O}(n)$	$\mathcal{O}(n)$	BWT & few additional structures

Suffix Array Manber& Myers 1990 and Burrows-Wheeler Transform 1994

Suffix Array: Index allowing fast localisation of patterns.

BWT: Reversible permutation used in compression and indexing.

LCP: Longest Common Prefix between two rows of the suffix array.

Suffix Array Manber& Myers 1990 and Burrows-Wheeler Transform 1994

Suffix Array: Index allowing fast localisation of patterns.
BWT: Reversible permutation used in compression and indexing.
LCP: Longest Common Prefix between two rows of the suffix array.

$$y = \overset{0}{A}\overset{1}{A}\overset{2}{C}\overset{3}{A}\overset{4}{C}\overset{5}{A}\overset{6}{C}\overset{7}{C}\overset{8}{\#}$$

0	A	A	C	A	C	A	C	C	#
1	A	C	A	C	A	C	C	#	A
2	C	A	C	A	C	C	#	A	A
3	A	C	A	C	C	#	A	A	C
4	C	A	C	C	#	A	A	C	A
5	A	C	C	#	A	A	C	A	C
6	C	C	#	A	A	C	A	C	A
7	C	#	A	A	C	A	C	A	C
8	#	A	A	C	A	C	A	C	C

Rotations of y

Ordered rotations of y

Suffix Array Manber& Myers 1990 and Burrows-Wheeler Transform 1994

Suffix Array: Index allowing fast localisation of patterns.
BWT: Reversible permutation used in compression and indexing.
LCP: Longest Common Prefix between two rows of the suffix array.

$$y = \overset{0}{A}\overset{1}{A}\overset{2}{C}\overset{3}{A}\overset{4}{C}\overset{5}{A}\overset{6}{C}\overset{7}{C}\overset{8}{\#}$$

0	A	A	C	A	C	A	C	C	#
1	A	C	A	C	A	C	C	#	A
2	C	A	C	A	C	C	#	A	A
3	A	C	A	C	C	#	A	A	C
4	C	A	C	C	#	A	A	C	A
5	A	C	C	#	A	A	C	A	C
6	C	C	#	A	A	C	A	C	A
7	C	#	A	A	C	A	C	A	C
8	#	**A**	**A**	**C**	**A**	**C**	**A**	**C**	**C**

Rotations of y

$\Rightarrow$

pos									*BWT*
8	#	A	A	C	A	C	A	C	C

Ordered rotations of y

Suffix Array Manber& Myers 1990 and Burrows-Wheeler Transform 1994

Suffix Array: Index allowing fast localisation of patterns.
BWT: Reversible permutation used in compression and indexing.
LCP: Longest Common Prefix between two rows of the suffix array.

$$y = \overset{0}{A}\overset{1}{A}\overset{2}{C}\overset{3}{A}\overset{4}{C}\overset{5}{A}\overset{6}{C}\overset{7}{C}\overset{8}{\#}$$

0	**A**	**A**	**C**	**A**	**C**	**A**	**C**	**C**	#
1	A	C	A	C	A	C	C	#	A
2	C	A	C	A	C	C	#	A	A
3	A	C	A	C	C	#	A	A	C
4	C	A	C	C	#	A	A	C	A
5	A	C	C	#	A	A	C	A	C
6	C	C	#	A	A	C	A	C	A
7	C	#	A	A	C	A	C	A	C
8	#	A	A	C	A	C	A	C	C

Rotations of y

$\Rightarrow$

pos									*BWT*
8	#	A	A	C	A	C	A	C	C
0	A	A	C	A	C	A	C	C	#

Ordered rotations of y

Suffix Array Manber& Myers 1990 and Burrows-Wheeler Transform 1994

Suffix Array: Index allowing fast localisation of patterns.
BWT: Reversible permutation used in compression and indexing.
LCP: Longest Common Prefix between two rows of the suffix array.

$$y = \overset{0}{A}\overset{1}{A}\overset{2}{C}\overset{3}{A}\overset{4}{C}\overset{5}{A}\overset{6}{C}\overset{7}{C}\overset{8}{\#}$$

0	A	A	C	A	C	A	C	C	#
1	**A**	**C**	**A**	**C**	**A**	**C**	**C**	#	**A**
2	C	A	C	A	C	C	#	A	A
3	A	C	A	C	C	#	A	A	C
4	C	A	C	C	#	A	A	C	A
5	A	C	C	#	A	A	C	A	C
6	C	C	#	A	A	C	A	C	A
7	C	#	A	A	C	A	C	A	C
8	#	A	A	C	A	C	A	C	C

Rotations of y

$\Rightarrow$

pos									*BWT*
8	#	A	A	C	A	C	A	C	C
0	A	A	C	A	C	A	C	C	#
1	A	C	A	C	A	C	C	#	A

Ordered rotations of y

Suffix Array Manber& Myers 1990 and Burrows-Wheeler Transform 1994

Suffix Array: Index allowing fast localisation of patterns.
BWT: Reversible permutation used in compression and indexing.
LCP: Longest Common Prefix between two rows of the suffix array.

$$y = \overset{0}{A}\overset{1}{A}\overset{2}{C}\overset{3}{A}\overset{4}{C}\overset{5}{A}\overset{6}{C}\overset{7}{C}\overset{8}{\#}$$

0	A	A	C	A	C	A	C	C	#
1	A	C	A	C	A	C	C	#	A
2	C	A	C	A	C	C	#	A	A
3	**A**	**C**	**A**	**C**	**C**	#	**A**	**A**	**C**
4	C	A	C	C	#	A	A	C	A
5	A	C	C	#	A	A	C	A	C
6	C	C	#	A	A	C	A	C	A
7	C	#	A	A	C	A	C	A	C
8	#	A	A	C	A	C	A	C	C

Rotations of y

$\Rightarrow$

pos									*BWT*
8	#	A	A	C	A	C	A	C	C
0	A	A	C	A	C	A	C	C	#
1	A	C	A	C	A	C	C	#	A
3	A	C	A	C	C	#	A	A	C

Ordered rotations of y

Suffix Array Manber& Myers 1990 and Burrows-Wheeler Transform 1994

Suffix Array: Index allowing fast localisation of patterns.
BWT: Reversible permutation used in compression and indexing.
LCP: Longest Common Prefix between two rows of the suffix array.

$$y = \overset{0\ 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8}{A\ A\ C\ A\ C\ A\ C\ C\ \#}$$

0	A	A	C	A	C	A	C	C	#
1	A	C	A	C	A	C	C	#	A
2	C	A	C	A	C	C	#	A	A
3	A	C	A	C	C	#	A	A	C
4	C	A	C	C	#	A	A	C	A
5	A	C	C	#	A	A	C	A	C
6	C	C	#	A	A	C	A	C	A
7	C	#	A	A	C	A	C	A	C
8	#	A	A	C	A	C	A	C	C

Rotations of y

$\Rightarrow$

SA									*BWT*
8	#	A	A	C	A	C	A	C	C
0	A	A	C	A	C	A	C	C	#
1	A	C	A	C	A	C	C	#	A
3	A	C	A	C	C	#	A	A	C
5	A	C	C	#	A	A	C	A	C
7	C	#	A	A	C	A	C	A	C
2	C	A	C	A	C	C	#	A	A
4	C	A	C	C	#	A	A	C	A
6	C	C	#	A	A	C	A	C	A

Ordered rotations of y

Suffix Array Manber& Myers 1990 and Burrows-Wheeler Transform 1994

Suffix Array: Index allowing fast localisation of patterns.
BWT: Reversible permutation used in compression and indexing.
LCP: Longest Common Prefix between two rows of the suffix array.

$$\overset{0\ 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8}{y = \mathrm{A\,A\,C\,A\,C\,A\,C\,C\,\#}}$$

$$BWT(y) = \mathrm{C\#ACCCAAA}, \text{ and } SA(y) = [8,0,1,3,5,7,2,4,6]$$

0	A	A	C	A	C	A	C	C	#
1	A	C	A	C	A	C	C	#	A
2	C	A	C	A	C	C	#	A	A
3	A	C	A	C	C	#	A	A	C
4	C	A	C	C	#	A	A	C	A
5	A	C	C	#	A	A	C	A	C
6	C	C	#	A	A	C	A	C	A
7	C	#	A	A	C	A	C	A	C
8	#	A	A	C	A	C	A	C	C

Rotations of y

$\Rightarrow$

LCP	*SA*									*BWT*
0	8	#	A	A	C	A	C	A	C	C
0	0	A	A	C	A	C	A	C	C	#
1	1	A	C	A	C	A	C	C	#	A
4	3	A	C	A	C	C	#	A	A	C
2	5	A	C	C	#	A	A	C	A	C
0	7	C	#	A	A	C	A	C	A	C
1	2	C	A	C	A	C	C	#	A	A
3	4	C	A	C	C	#	A	A	C	A
1	6	C	C	#	A	A	C	A	C	A

Ordered rotations of y

Parallel Computation of Suffix Array

Kulla F. and Sanders, P.: Scalable parallel suffix array construction, 2007

Parallel Construction of LCP-table

Shun, J.: Fast parallel computation of longest common prefixes, 2014

Table of Contents

Problem

Input: A word y of length n on Σ a fixed size alphabet ($\sigma = \mathcal{O}(1)$)
Output: Set of minimal absent words of y

Our contribution

- Algorithm linear in time and space based on the suffix array structure.
- Implementation of this algorithm for shared-memory multiprocessing programming

Definition: Maximal repeated pair

A maximal repeated pair in a y is a triple (i, j, w) such that:

- w occurs in y at positions i and j
- $y[i-1] \neq y[j-1]$
- $y[i+|w|] \neq y[j+|w|]$

Definition: Maximal repeated pair

A maximal repeated pair in a y is a triple (i, j, w) such that:

- w occurs in y at positions i and j
- $y[i-1] \neq y[j-1]$
- $y[i+|w|] \neq y[j+|w|]$

Lemma

If awb is a minimal absent word of y, then there exists i and j such that (i, j, w) is a maximal repeated pair of y.

Definition: Maximal repeated pair

A maximal repeated pair in a y is a triple (i, j, w) such that:

- w occurs in y at positions i and j
- $y[i-1] \neq y[j-1]$
- $y[i+|w|] \neq y[j+|w|]$

Lemma

If awb is a minimal absent word of y, then there exists i and j such that (i, j, w) is a maximal repeated pair of y.

Example, $AACC$ is a minimal absent word of y

$$\begin{array}{l} \overset{0\ 1\ 2\ 3\ 4\ 5\ 6\ 7}{y{=}\mathrm{A\,A\,C\,A\,C\,A\,C\,C}} \\ w{=}\mathrm{AC} \text{ occurs at position 1, 3 and 5.} \\ (1, 5, AC) \text{ is a maximal repeated pair.} \end{array}$$

Definition: Maximal repeated pair

A maximal repeated pair in a y is a triple (i, j, w) such that:

- w occurs in y at positions i and j
- $y[i-1] \neq y[j-1]$
- $y[i+|w|] \neq y[j+|w|]$

Lemma

If awb is a minimal absent word of y, then there exists i and j such that (i, j, w) is a maximal repeated pair of y.

Example, $AACC$ is a minimal absent word of y

$$y = \overset{0}{\mathrm{A}}\,\overset{1}{\mathrm{A}}\,\overset{2}{\mathrm{C}}\,\overset{3}{\mathrm{A}}\,\overset{4}{\mathrm{C}}\,\overset{5}{\mathrm{A}}\,\overset{6}{\mathrm{C}}\,\overset{7}{\mathrm{C}}$$

w=AC occurs at position 1, 3 and 5.

$(1, 5, AC)$ is a maximal repeated pair.

Definition: Maximal repeated pair

A maximal repeated pair in a y is a triple (i, j, w) such that:

- w occurs in y at positions i and j
- $y[i-1] \neq y[j-1]$
- $y[i+|w|] \neq y[j+|w|]$

Lemma

If awb is a minimal absent word of y, then there exists i and j such that (i, j, w) is a maximal repeated pair of y.

Example, $AACC$ is a minimal absent word of y

$$\begin{array}{c} \overset{0}{\mathrm{A}}\ \overset{1}{\mathrm{A}}\ \overset{2}{\mathrm{C}}\ \overset{3}{\mathrm{A}}\ \overset{4}{\mathrm{C}}\ \overset{5}{\mathrm{A}}\ \overset{6}{\mathrm{C}}\ \overset{7}{\mathrm{C}} \\ y = \mathrm{A\,A\,C\,A\,C\,A\,C\,C} \end{array}$$

w=AC occurs at position 1, 3 and 5.

$(1, 5, AC)$ is a maximal repeated pair.

Definition: Longest Common Prefix table

LCP[0]=0, for all $1 \leq r \leq n-1$ LCP[r] is the length of the longest common prefix of the rows r and $r-1$.

row	SA	LCP	Ordered rotations of $y = \overset{0}{A}\overset{1}{A}\overset{2}{A}\overset{3}{C}\overset{4}{C}\overset{5}{C}\overset{6}{A}\overset{7}{A}\overset{8}{A}\overset{9}{C}\overset{10}{A}\overset{11}{C}\overset{12}{C}\#$												
0	6	0	A	A	A	C	A	C	C	#	A	A	A	C	C	C
1	0	4	A	A	A	C	C	C	A	A	A	C	A	C	C	#
2	7	2	A	A	C	A	C	C	#	A	A	A	C	C	C	A
3	1	3	A	A	C	C	C	A	A	A	C	A	C	C	#	A
4	8	1	A	C	A	C	C	#	A	A	A	C	C	C	A	A
5	10	2	A	C	C	#	A	A	A	C	C	C	A	A	A	C
6	2	3	A	C	C	C	A	A	A	C	A	C	C	#	A	A
7	12	0	C	#	A	A	A	C	C	C	A	A	A	C	A	C
8	5	1	C	A	A	A	C	A	C	C	#	A	A	A	C	C
9	9	2	C	A	C	C	#	A	A	A	C	C	C	A	A	A
10	11	1	C	C	#	A	A	A	C	C	C	A	A	A	C	A
11	4	2	C	C	A	A	A	C	A	C	C	#	A	A	A	C
12	3	2	C	C	C	A	A	A	C	A	C	C	#	A	A	A

Lemma

Let y be a word of size n over an alphabet of size σ, we have:
$\forall k \geq 0, |\{s \in [0, n-1] : LCP[s] = k\}| \leq (\sigma - 1)\sigma^k + 1$
the equality holds iif every word of length $k+1$ occur in y.

Lemma

Let y be a word of size n over an alphabet of size σ, we have:
$\forall k \geq 0, |\{s \in [0, n-1] : LCP[s] = k\}| \leq (\sigma - 1)\sigma^k + 1$
the equality holds iif every word of length $k+1$ occur in y.

LCP	Ordered rotations of $y = AAACCCAAACACC\#$													
0	A	A	A	C	A	C	C	#	A	A	A	C	C	C
4	A	A	A	C	C	C	A	A	A	C	A	C	C	#
2	A	A	C	A	C	C	#	A	A	A	C	C	C	A
3	A	A	C	C	C	A	A	A	C	A	C	C	#	A
1	A	C	A	C	C	#	A	A	A	C	C	C	A	A
2	A	C	C	#	A	A	A	C	C	C	A	A	A	C
3	A	C	C	C	A	A	A	C	A	C	C	#	A	A
0	C	#	A	A	A	C	C	C	A	A	A	C	A	C
1	C	A	A	A	C	A	C	C	#	A	A	A	C	C
2	C	A	C	C	#	A	A	A	C	C	C	A	A	A
1	C	C	#	A	A	A	C	C	C	A	A	A	C	A
2	C	C	A	A	A	C	A	C	C	#	A	A	A	C
2	C	C	C	A	A	A	C	A	C	C	#	A	A	A

LCP value	number of occ.
0	
1	
2	
3	
4	

Lemma

Let y be a word of size n over an alphabet of size σ, we have:
$\forall k \geq 0, |\{s \in [0, n-1] : LCP[s] = k\}| \leq (\sigma - 1)\sigma^k + 1$
the equality holds iif every word of length $k+1$ occur in y.

LCP	Ordered rotations of $y = AAACCCAAACACC\#$													
0	A	A	A	C	A	C	C	#	A	A	A	C	C	C
4	A	A	A	C	C	C	A	A	A	C	A	C	C	#
2	A	A	C	A	C	C	#	A	A	A	C	C	C	A
3	A	A	C	C	C	A	A	A	C	A	C	C	#	A
1	A	C	A	C	C	#	A	A	A	C	C	C	A	A
2	A	C	C	#	A	A	A	C	C	C	A	A	A	C
3	A	C	C	C	A	A	A	C	A	C	C	#	A	A
0	C	#	A	A	A	C	C	C	A	A	A	C	A	C
1	C	A	A	A	C	A	C	C	#	A	A	A	C	C
2	C	A	C	C	#	A	A	A	C	C	C	A	A	A
1	C	C	#	A	A	A	C	C	C	A	A	A	C	A
2	C	C	A	A	A	C	A	C	C	#	A	A	A	C
2	C	C	C	A	A	A	C	A	C	C	#	A	A	A

LCP value	number of occ.
0	
1	
2	
3	
4	

Lemma

Let y be a word of size n over an alphabet of size σ, we have:
$\forall k \geq 0, |\{s \in [0, n-1] : LCP[s] = k\}| \leq (\sigma - 1)\sigma^k + 1$
the equality holds iif every word of length $k+1$ occur in y.

LCP	Ordered rotations of $y = AAACCCAAACACC\#$													
0	A	A	A	C	A	C	C	#	A	A	A	C	C	C
4	A	A	A	C	C	C	A	A	A	C	A	C	C	#
2	A	A	C	A	C	C	#	A	A	A	C	C	C	A
3	A	A	C	C	C	A	A	A	C	A	C	C	#	A
1	A	C	A	C	C	#	A	A	A	C	C	C	A	A
2	A	C	C	#	A	A	A	C	C	C	A	A	A	C
3	A	C	C	C	A	A	A	C	A	C	C	#	A	A
0	C	#	A	A	A	C	C	C	A	A	A	C	A	C
1	C	A	A	A	C	A	C	C	#	A	A	A	C	C
2	C	A	C	C	#	A	A	A	C	C	C	A	A	A
1	C	C	#	A	A	A	C	C	C	A	A	A	C	A
2	C	C	A	A	A	C	A	C	C	#	A	A	A	C
2	C	C	C	A	A	A	C	A	C	C	#	A	A	A

LCP value	number of occ.
0	2
1	
2	
3	
4	

Lemma

Let y be a word of size n over an alphabet of size σ, we have:
$\forall k \geq 0, |\{s \in [0, n-1] : LCP[s] = k\}| \leq (\sigma - 1)\sigma^k + 1$
the equality holds iif every word of length $k+1$ occur in y.

LCP	Ordered rotations of $y = AAACCCAAACACC\#$													
0	A	A	A	C	A	C	C	#	A	A	A	C	C	C
4	A	A	A	C	C	C	A	A	A	C	A	C	C	#
2	A	A	C	A	C	C	#	A	A	A	C	C	C	A
3	A	A	C	C	C	A	A	A	C	A	C	C	#	A
1	A	C	A	C	C	#	A	A	A	C	C	C	A	A
2	A	C	C	#	A	A	A	C	C	C	A	A	A	C
3	A	C	C	C	A	A	A	C	A	C	C	#	A	A
0	C	#	A	A	A	C	C	C	A	A	A	C	A	C
1	C	A	A	A	C	A	C	C	#	A	A	A	C	C
2	C	A	C	C	#	A	A	A	C	C	C	A	A	A
1	C	C	#	A	A	A	C	C	C	A	A	A	C	A
2	C	C	A	A	A	C	A	C	C	#	A	A	A	C
2	C	C	C	A	A	A	C	A	C	C	#	A	A	A

LCP value	number of occ.
0	2
1	
2	
3	
4	

Lemma

Let y be a word of size n over an alphabet of size σ, we have:
$\forall k \geq 0, |\{s \in [0, n-1] : LCP[s] = k\}| \leq (\sigma - 1)\sigma^k + 1$
the equality holds iif every word of length $k+1$ occur in y.

LCP	Ordered rotations of $y = AAACCCAAACACC\#$													
0	A	A	A	C	A	C	C	#	A	A	A	C	C	C
4	A	A	A	C	C	C	A	A	A	C	A	C	C	#
2	A	A	C	A	C	C	#	A	A	A	C	C	C	A
3	A	A	C	C	C	A	A	A	C	A	C	C	#	A
1	A	C	A	C	C	#	A	A	A	C	C	C	A	A
2	A	C	C	#	A	A	A	C	C	C	A	A	A	C
3	A	C	C	C	A	A	A	C	A	C	C	#	A	A
0	C	#	A	A	A	C	C	C	A	A	A	C	A	C
1	C	A	A	A	C	A	C	C	#	A	A	A	C	C
2	C	A	C	C	#	A	A	A	C	C	C	A	A	A
1	C	C	#	A	A	A	C	C	C	A	A	A	C	A
2	C	C	A	A	A	C	A	C	C	#	A	A	A	C
2	C	C	C	A	A	A	C	A	C	C	#	A	A	A

LCP value	number of occ.
0	2
1	3
2	
3	
4	

Lemma

Let y be a word of size n over an alphabet of size σ, we have:
$\forall k \geq 0, |\{s \in [0, n-1] : LCP[s] = k\}| \leq (\sigma - 1)\sigma^k + 1$
the equality holds iif every word of length $k+1$ occur in y.

LCP	Ordered rotations of $y = AAACCCAAACACC\#$													
0	A	A	A	C	A	C	C	#	A	A	A	C	C	C
4	A	A	A	C	C	C	A	A	A	C	A	C	C	#
2	A	A	C	A	C	C	#	A	A	A	C	C	C	A
3	A	A	C	C	C	A	A	A	C	A	C	C	#	A
1	A	C	A	C	C	#	A	A	A	C	C	C	A	A
2	A	C	C	#	A	A	A	C	C	C	A	A	A	C
3	A	C	C	C	A	A	A	C	A	C	C	#	A	A
0	C	#	A	A	A	C	C	C	A	A	A	C	A	C
1	C	A	A	A	C	A	C	C	#	A	A	A	C	C
2	C	A	C	C	#	A	A	A	C	C	C	A	A	A
1	C	C	#	A	A	A	C	C	C	A	A	A	C	A
2	C	C	A	A	A	C	A	C	C	#	A	A	A	C
2	C	C	C	A	A	A	C	A	C	C	#	A	A	A

LCP value	number of occ.
0	2
1	3
2	
3	
4	

Lemma

Let y be a word of size n over an alphabet of size σ, we have:
$\forall k \geq 0, |\{s \in [0, n-1] : LCP[s] = k\}| \leq (\sigma - 1)\sigma^k + 1$
the equality holds iif every word of length $k+1$ occur in y.

LCP	Ordered rotations of $y = AAACCCAAACACC\#$													
0	A	A	A	C	A	C	C	#	A	A	A	C	C	C
4	A	A	A	C	C	C	A	A	A	C	A	C	C	#
2	A	A	C	A	C	C	#	A	A	A	C	C	C	A
3	A	A	C	C	C	A	A	A	C	A	C	C	#	A
1	A	C	A	C	C	#	A	A	A	C	C	C	A	A
2	A	C	C	#	A	A	A	C	C	C	A	A	A	C
3	A	C	C	C	A	A	A	C	A	C	C	#	A	A
0	C	#	A	A	A	C	C	C	A	A	A	C	A	C
1	C	A	A	A	C	A	C	C	#	A	A	A	C	C
2	C	A	C	C	#	A	A	A	C	C	C	A	A	A
1	C	C	#	A	A	A	C	C	C	A	A	A	C	A
2	C	C	A	A	A	C	A	C	C	#	A	A	A	C
2	C	C	C	A	A	A	C	A	C	C	#	A	A	A

LCP value	number of occ.
0	2
1	3
2	5
3	
4	

Lemma

Let y be a word of size n over an alphabet of size σ, we have:
$\forall k \geq 0, |\{s \in [0, n-1] : LCP[s] = k\}| \leq (\sigma - 1)\sigma^k + 1$
the equality holds iif every word of length $k+1$ occur in y.

LCP	Ordered rotations of $y = AAACCCAAACACC\#$													
0	A	A	A	C	A	C	C	#	A	A	A	C	C	C
4	A	A	A	C	C	C	A	A	A	C	A	C	C	#
2	A	A	C	A	C	C	#	A	A	A	C	C	C	A
3	A	A	C	C	C	A	A	A	C	A	C	C	#	A
1	A	C	A	C	C	#	A	A	A	C	C	C	A	A
2	A	C	C	#	A	A	A	C	C	C	A	A	A	C
3	A	C	C	C	A	A	A	C	A	C	C	#	A	A
0	C	#	A	A	A	C	C	C	A	A	A	C	A	C
1	C	A	A	A	C	A	C	C	#	A	A	A	C	C
2	C	A	C	C	#	A	A	A	C	C	C	A	A	A
1	C	C	#	A	A	A	C	C	C	A	A	A	C	A
2	C	C	A	A	A	C	A	C	C	#	A	A	A	C
2	C	C	C	A	A	A	C	A	C	C	#	A	A	A

LCP value	number of occ.
0	2
1	3
2	5
3	
4	

Lemma

Let y be a word of size n over an alphabet of size σ, we have:
$\forall k \geq 0, |\{s \in [0, n-1] : LCP[s] = k\}| \leq (\sigma - 1)\sigma^k + 1$
the equality holds iif every word of length $k+1$ occur in y.

LCP	Ordered rotations of $y = AAACCCAAACACC\#$													
0	A	A	A	C	A	C	C	#	A	A	A	C	C	C
4	A	A	A	C	C	C	A	A	A	C	A	C	C	#
2	A	A	C	A	C	C	#	A	A	A	C	C	C	A
3	A	A	C	C	C	A	A	A	C	A	C	C	#	A
1	A	C	A	C	C	#	A	A	A	C	C	C	A	A
2	A	C	C	#	A	A	A	C	C	C	A	A	A	C
3	A	C	C	C	A	A	A	C	A	C	C	#	A	A
0	C	#	A	A	A	C	C	C	A	A	A	C	A	C
1	C	A	A	A	C	A	C	C	#	A	A	A	C	C
2	C	A	C	C	#	A	A	A	C	C	C	A	A	A
1	C	C	#	A	A	A	C	C	C	A	A	A	C	A
2	C	C	A	A	A	C	A	C	C	#	A	A	A	C
2	C	C	C	A	A	A	C	A	C	C	#	A	A	A

LCP value	number of occ.
0	2
1	3
2	5
3	2
4	

absent words are of length ≥ 4

Lemma

Let y be a word of size n over an alphabet of size σ, we have:
$\forall k \geq 0, |\{s \in [0, n-1] : LCP[s] = k\}| \leq (\sigma - 1)\sigma^k + 1$
the equality holds iif every word of length $k+1$ occur in y.

LCP	Ordered rotations of $y = AAACCCAAACACC\#$													
0	A	A	A	C	A	C	C	#	A	A	A	C	C	C
4	A	A	A	C	C	C	A	A	A	C	A	C	C	#
2	A	A	C	A	C	C	#	A	A	A	C	C	C	A
3	A	A	C	C	C	A	A	A	C	A	C	C	#	A
1	A	C	A	C	C	#	A	A	A	C	C	C	A	A
2	A	C	C	#	A	A	A	C	C	C	A	A	A	C
3	A	C	C	C	A	A	A	C	A	C	C	#	A	A
0	C	#	A	A	A	C	C	C	A	A	A	C	A	C
1	C	A	A	A	C	A	C	C	#	A	A	A	C	C
2	C	A	C	C	#	A	A	A	C	C	C	A	A	A
1	C	C	#	A	A	A	C	C	C	A	A	A	C	A
2	C	C	A	A	A	C	A	C	C	#	A	A	A	C
2	C	C	C	A	A	A	C	A	C	C	#	A	A	A

LCP value	number of occ.
0	2
1	3
2	5
3	2
4	

absent words are of length ≥ 4

Lemma

Let y be a word of size n over an alphabet of size σ, we have:
$\forall k \geq 0, |\{s \in [0, n-1] : LCP[s] = k\}| \leq (\sigma - 1)\sigma^k + 1$
the equality holds iif every word of length $k+1$ occur in y.

LCP	Ordered rotations of $y = AAACCCAAACACC\#$													
0	A	A	A	C	A	C	C	#	A	A	A	C	C	C
4	A	A	A	C	C	C	A	A	A	C	A	C	C	#
2	A	A	C	A	C	C	#	A	A	A	C	C	C	A
3	A	A	C	C	C	A	A	A	C	A	C	C	#	A
1	A	C	A	C	C	#	A	A	A	C	C	C	A	A
2	A	C	C	#	A	A	A	C	C	C	A	A	A	C
3	A	C	C	C	A	A	A	C	A	C	C	#	A	A
0	C	#	A	A	A	C	C	C	A	A	A	C	A	C
1	C	A	A	A	C	A	C	C	#	A	A	A	C	C
2	C	A	C	C	#	A	A	A	C	C	C	A	A	A
1	C	C	#	A	A	A	C	C	C	A	A	A	C	A
2	C	C	A	A	A	C	A	C	C	#	A	A	A	C
2	C	C	C	A	A	A	C	A	C	C	#	A	A	A

LCP value	number of occ.
0	2
1	3
2	5
3	2
4	1

absent words are of length ≥ 4

Lemma

Let y be a word of size n over an alphabet of size σ, we have:
$\forall k \geq 0, |\{s \in [0, n-1] : LCP[s] = k\}| \leq (\sigma - 1)\sigma^k + 1$
the equality holds iif every word of length $k+1$ occur in y.

LCP	Ordered rotations of $y = AAACCCAAACACC\#$													
0	A	A	A	C	A	C	C	#	A	A	A	C	C	C
4	A	A	A	C	C	C	A	A	A	C	A	C	C	#
2	A	A	C	A	C	C	#	A	A	A	C	C	C	A
3	A	A	C	C	C	A	A	A	C	A	C	C	#	A
1	A	C	A	C	C	#	A	A	A	C	C	C	A	A
2	A	C	C	#	A	A	A	C	C	C	A	A	A	C
3	A	C	C	C	A	A	A	C	A	C	C	#	A	A
0	C	#	A	A	A	C	C	C	A	A	A	C	A	C
1	C	A	A	A	C	A	C	C	#	A	A	A	C	C
2	C	A	C	C	#	A	A	A	C	C	C	A	A	A
1	C	C	#	A	A	A	C	C	C	A	A	A	C	A
2	C	C	A	A	A	C	A	C	C	#	A	A	A	C
2	C	C	C	A	A	A	C	A	C	C	#	A	A	A

LCP value	number of occ.
0	2
1	3
2	5
3	2
4	1

absent words are of length ≥ 4

LCP	SA	BWT	Factor	MAWs
0	6	C	A A A C A	
4	0	#	A A A C C	
2	7	A	A A C A	
3	1	A	A A C C	
1	8	A	A C A	
2	10	C	A C C #	
3	2	A	A C C C	
0	12	C	C #	
1	5	C	C A A	
2	9	A	C A C	
1	11	A	C C #	
3	4	C	C C A	
2	3	A	C C C	

Algorithm step by step for the word $y =$ $\overset{0}{A}\,\overset{1}{A}\,\overset{2}{A}\,\overset{3}{C}\,\overset{4}{C}\,\overset{5}{C}\,\overset{6}{A}\,\overset{7}{A}\,\overset{8}{A}\,\overset{9}{C}\,\overset{10}{A}\,\overset{11}{C}\,\overset{12}{C}$

LCP	SA	BWT	Factor	MAWs
0	6	C	A	A A C A	
4	0	#	A	A A C C	CAAACC
2	7	A	A	A C A	
3	1	A	A	A C C	
1	8	A	A	C A	
2	10	C	A	C C #	
3	2	A	A	C C C	
0	12	C	C	#	
1	5	C	C	A A	
2	9	A	C	A C	
1	11	A	C	C #	
3	4	C	C	C A	
2	3	A	C	C C	

Algorithm step by step for the word $y =$ $\overset{0}{A}\overset{1}{A}\overset{2}{A}\overset{3}{C}\overset{4}{C}\overset{5}{C}\overset{6}{A}\overset{7}{A}\overset{8}{A}\overset{9}{C}\overset{10}{A}\overset{11}{C}\overset{12}{C}$

LCP	SA	BWT	Factor	MAWs
0	6	C	A A A C A	
4	0	#	A A A C C	CAAACC
2	7	A	A A C A	
3	1	A	A A C C	
1	8	A	A C A	
2	10	C	A C C #	
3	2	A	A C C C	
0	12	C	C #	
1	5	C	C A A	
2	9	A	C A C	
1	11	A	C C #	
3	4	C	C C A	
2	3	A	C C C	

Algorithm step by step for the word $y =$ $\overset{0}{A}\,\overset{1}{A}\,\overset{2}{A}\,\overset{3}{C}\,\overset{4}{C}\,\overset{5}{C}\,\overset{6}{A}\,\overset{7}{A}\,\overset{8}{A}\,\overset{9}{C}\,\overset{10}{A}\,\overset{11}{C}\,\overset{12}{C}$

LCP	SA	BWT	Factor	MAWs
0	6	C	A A A C A	
4	0	#	A A A C C	CAAACC
2	7	A	A A C A	
3	1	A	A A C C	
1	8	A	A C A	
2	10	C	A C C #	
3	2	A	A C C C	
0	12	C	C #	
1	5	C	C A A	
2	9	A	C A C	
1	11	A	C C #	
3	4	C	C C A	
2	3	A	C C C	

Algorithm step by step for the word $y =$ $\overset{0}{A}\overset{1}{A}\overset{2}{A}\overset{3}{C}\overset{4}{C}\overset{5}{C}\overset{6}{A}\overset{7}{A}\overset{8}{A}\overset{9}{C}\overset{10}{A}\overset{11}{C}\overset{12}{C}$

LCP	SA	BWT	Factor	MAWs
0	6	C	A	A A C A	
4	0	#	A	A A C C	CAAACC
2	7	A	A	A C A	AAAA, CAAC
3	1	A	A	A C C	
1	8	A	A	C A	
2	10	C	A	C C #	
3	2	A	A	C C C	
0	12	C	C	#	
1	5	C	C	A A	
2	9	A	C	A C	
1	11	A	C	C #	
3	4	C	C	C A	
2	3	A	C	C C	

Algorithm step by step for the word $y =$ $\overset{0}{A}\overset{1}{A}\overset{2}{A}\overset{3}{C}\overset{4}{C}\overset{5}{C}\overset{6}{A}\overset{7}{A}\overset{8}{A}\overset{9}{C}\overset{10}{A}\overset{11}{C}\overset{12}{C}$

LCP	SA	BWT	Factor	MAWs
0	6	C	A	A A C A	
4	0	#	A	A A C C	CAAACC
2	7	A	A	A C A	AAAA, CAAC
3	1	A	A	A C C	
1	8	A	A	C A	
2	10	C	A	C C #	
3	2	A	A	C C C	CACCC
0	12	C	C	#	
1	5	C	C	A A	
2	9	A	C	A C	
1	11	A	C	C #	
3	4	C	C	C A	
2	3	A	C	C C	

Algorithm step by step for the word $y =$ $\overset{0}{A}\overset{1}{A}\overset{2}{A}\overset{3}{C}\overset{4}{C}\overset{5}{C}\overset{6}{A}\overset{7}{A}\overset{8}{A}\overset{9}{C}\overset{10}{A}\overset{11}{C}\overset{12}{C}$

LCP	SA	BWT	Factor	MAWs
0	6	C	A	A A C A	
4	0	#	A	A A C C	CAAACC
2	7	A	A	A C A	AAAA, CAAC
3	1	A	A	A C C	
1	8	A	A	C A	
2	10	C	A	C C #	CACA
3	2	A	A	C C C	CACCC
0	12	C	C	#	
1	5	C	C	A A	
2	9	A	C	A C	
1	11	A	C	C #	
3	4	C	C	C A	
2	3	A	C	C C	

Algorithm step by step for the word $y =$ $\overset{0}{A}\overset{1}{A}\overset{2}{A}\overset{3}{C}\overset{4}{C}\overset{5}{C}\overset{6}{A}\overset{7}{A}\overset{8}{A}\overset{9}{C}\overset{10}{A}\overset{11}{C}\overset{12}{C}$

LCP	SA	BWT	Factor	MAWs
0	6	C	A	
4	0	#	A A A C	C	CAAACC
2	7	A	A A C A	AAAA, CAAC
3	1	A	A A C C	
1	8	A	A C A	
2	10	C	A C C #	CACA
3	2	A	A C C C	CACCC
0	12	C	C #	
1	5	C	C A A	
2	9	A	C A C	CCAC, ACAA
1	11	A	C C #	
3	4	C	C C A	
2	3	A	C C C	

Algorithm step by step for the word $y =$ $\overset{0}{A}\,\overset{1}{A}\,\overset{2}{A}\,\overset{3}{C}\,\overset{4}{C}\,\overset{5}{C}\,\overset{6}{A}\,\overset{7}{A}\,\overset{8}{A}\,\overset{9}{C}\,\overset{10}{A}\,\overset{11}{C}\,\overset{12}{C}$

LCP	SA	BWT	Factor	MAWs
0	6	C	A A A C A	
4	0	#	A A A C C	CAAACC
2	7	A	A A C A	AAAA, CAAC
3	1	A	A A C C	
1	8	A	A C A	
2	10	C	A C C #	CACA
3	2	A	A C C C	CACCC
0	12	C	C #	
1	5	C	C A A	
2	9	A	C A C	CCAC, ACAA
1	11	A	C C #	
3	4	C	C C A	
2	3	A	C C C	

Algorithm step by step for the word $y =$ $\overset{0}{A}\,\overset{1}{A}\,\overset{2}{A}\,\overset{3}{C}\,\overset{4}{C}\,\overset{5}{C}\,\overset{6}{A}\,\overset{7}{A}\,\overset{8}{A}\,\overset{9}{C}\,\overset{10}{A}\,\overset{11}{C}\,\overset{12}{C}$

LCP	SA	BWT	Factor	MAWs
0	6	C	A A A C A	
4	0	#	A A A C C	CAAACC
2	7	A	A A C A	AAAA, CAAC
3	1	A	A A C C	
1	8	A	A C A	
2	10	C	A C C #	CACA
3	2	A	A C C C	CACCC
0	12	C	C #	
1	5	C	C A A	
2	9	A	C A C	CCAC, ACAA
1	11	A	C C #	
3	4	C	C C A	ACCA, CCCC
2	3	A	C C C	

Algorithm step by step for the word $y =$ $\overset{0}{A}\,\overset{1}{A}\,\overset{2}{A}\,\overset{3}{C}\,\overset{4}{C}\,\overset{5}{C}\,\overset{6}{A}\,\overset{7}{A}\,\overset{8}{A}\,\overset{9}{C}\,\overset{10}{A}\,\overset{11}{C}\,\overset{12}{C}$

i	LCP	SA	BWT	Factor	MAWs
0	0	6	C	A A A C A	
1	4	0	#	A A A C C	
2	2	7	A	A A C A	
3	3	1	A	A A C C	
4	1	8	A	A C A	
5	2	10	C	A C C #	
6	3	2	A	A C C C	
7	0	12	C	C #	
8	1	5	C	C A A	
9	2	9	A	C A C	
10	1	11	A	C C #	
11	2	4	C	C C A	
12	2	3	A	C C C	

Algorithm step by step for the word $y =$ $\overset{0}{A}\,\overset{1}{A}\,\overset{2}{A}\,\overset{3}{C}\,\overset{4}{C}\,\overset{5}{C}\,\overset{6}{A}\,\overset{7}{A}\,\overset{8}{A}\,\overset{9}{C}\,\overset{10}{A}\,\overset{11}{C}\,\overset{12}{C}$

i	LCP	SA	BWT	Factor	MAWs
0	0	6	C	A A A C A	
1	4	0	#	A A A C C	CAAACC
2	2	7	A	A A C A	
3	3	1	A	A A C C	
4	1	8	A	A C A	
5	2	10	C	A C C #	
6	3	2	A	A C C C	CACCC
7	0	12	C	C #	
8	1	5	C	C A A	
9	2	9	A	C A C	CCAC, ACAA
10	1	11	A	C C #	
11	2	4	C	C C A	
12	2	3	A	C C C	

Algorithm step by step for the word $y =$ $\overset{0}{A}\,\overset{1}{A}\,\overset{2}{A}\,\overset{3}{C}\,\overset{4}{C}\,\overset{5}{C}\,\overset{6}{A}\,\overset{7}{A}\,\overset{8}{A}\,\overset{9}{C}\,\overset{10}{A}\,\overset{11}{C}\,\overset{12}{C}$

i	LCP	SA	BWT	Factor	MAWs
0	0	6	C	A A A C A	
1	4	0	#	A A A C C	CAAACC
2	2	7	A	A A C A	
3	3	1	A	A A C C	
4	1	8	A	A C A	
5	2	10	C	A C C #	CACA
6	3	2	A	A C C C	CACCC
7	0	12	C	C #	
8	1	5	C	C A A	
9	2	9	A	C A C	CCAC, ACAA
10	1	11	A	C C #	
11	2	4	C	C C A	ACCA, CCCC
12	2	3	A	C C C	

Algorithm step by step for the word $y =$ $\overset{0}{A}\,\overset{1}{A}\,\overset{2}{A}\,\overset{3}{C}\,\overset{4}{C}\,\overset{5}{C}\,\overset{6}{A}\,\overset{7}{A}\,\overset{8}{A}\,\overset{9}{C}\,\overset{10}{A}\,\overset{11}{C}\,\overset{12}{C}$

i	LCP	SA	BWT	Factor	MAWs
0	0	6	C	A A A C A	
1	4	0	#	A A A C C	CAAACC
2	2	7	A	A A C A	
3	3	1	A	A A C C	
4	1	8	A	A C A	
5	2	10	C	A C C #	CACA
6	3	2	A	A C C C	CACCC
7	0	12	C	C #	
8	1	5	C	C A A	
9	2	9	A	C A C	CCAC, ACAA
10	1	11	A	C C #	
11	2	4	C	C C A	ACCA, CCCC
12	2	3	A	C C C	

Algorithm step by step for the word $y =$ $\overset{0}{A}\,\overset{1}{A}\,\overset{2}{A}\,\overset{3}{C}\,\overset{4}{C}\,\overset{5}{C}\,\overset{6}{A}\,\overset{7}{A}\,\overset{8}{A}\,\overset{9}{C}\,\overset{10}{A}\,\overset{11}{C}\,\overset{12}{C}$

i	LCP	SA	BWT	Factor	MAWs
0	0	6	C	A A A C A	
1	4	0	#	A A A C C	CAAACC
2	2	7	A	A A C A	AAAA, CAAC
3	3	1	A	A A C C	
4	1	8	A	A C A	
5	2	10	C	A C C #	CACA
6	3	2	A	A C C C	CACCC
7	0	12	C	C #	
8	1	5	C	C A A	
9	2	9	A	C A C	CCAC, ACAA
10	1	11	A	C C #	
11	2	4	C	C C A	ACCA, CCCC
12	2	3	A	C C C	

Algorithm step by step for the word $y =$ $\overset{0}{A}\,\overset{1}{A}\,\overset{2}{A}\,\overset{3}{C}\,\overset{4}{C}\,\overset{5}{C}\,\overset{6}{A}\,\overset{7}{A}\,\overset{8}{A}\,\overset{9}{C}\,\overset{10}{A}\,\overset{11}{C}\,\overset{12}{C}$

i	LCP	SA	BWT	Factor	MAWs
0	0	6	C	A A A C A	
1	4	0	#	A A A C C	CAAACC
2	2	7	A	A A C A	AAAA, CAAC
3	3	1	A	A A C C	
4	1	8	A	A C A	
5	2	10	C	A C C #	CACA
6	3	2	A	A C C C	CACCC
7	0	12	C	C #	
8	1	5	C	C A A	
9	2	9	A	C A C	CCAC, ACAA
10	1	11	A	C C #	
11	2	4	C	C C A	ACCA, CCCC
12	2	3	A	C C C	

Algorithm step by step for the word $y =$ $\overset{0}{A}\,\overset{1}{A}\,\overset{2}{A}\,\overset{3}{C}\,\overset{4}{C}\,\overset{5}{C}\,\overset{6}{A}\,\overset{7}{A}\,\overset{8}{A}\,\overset{9}{C}\,\overset{10}{A}\,\overset{11}{C}\,\overset{12}{C}$

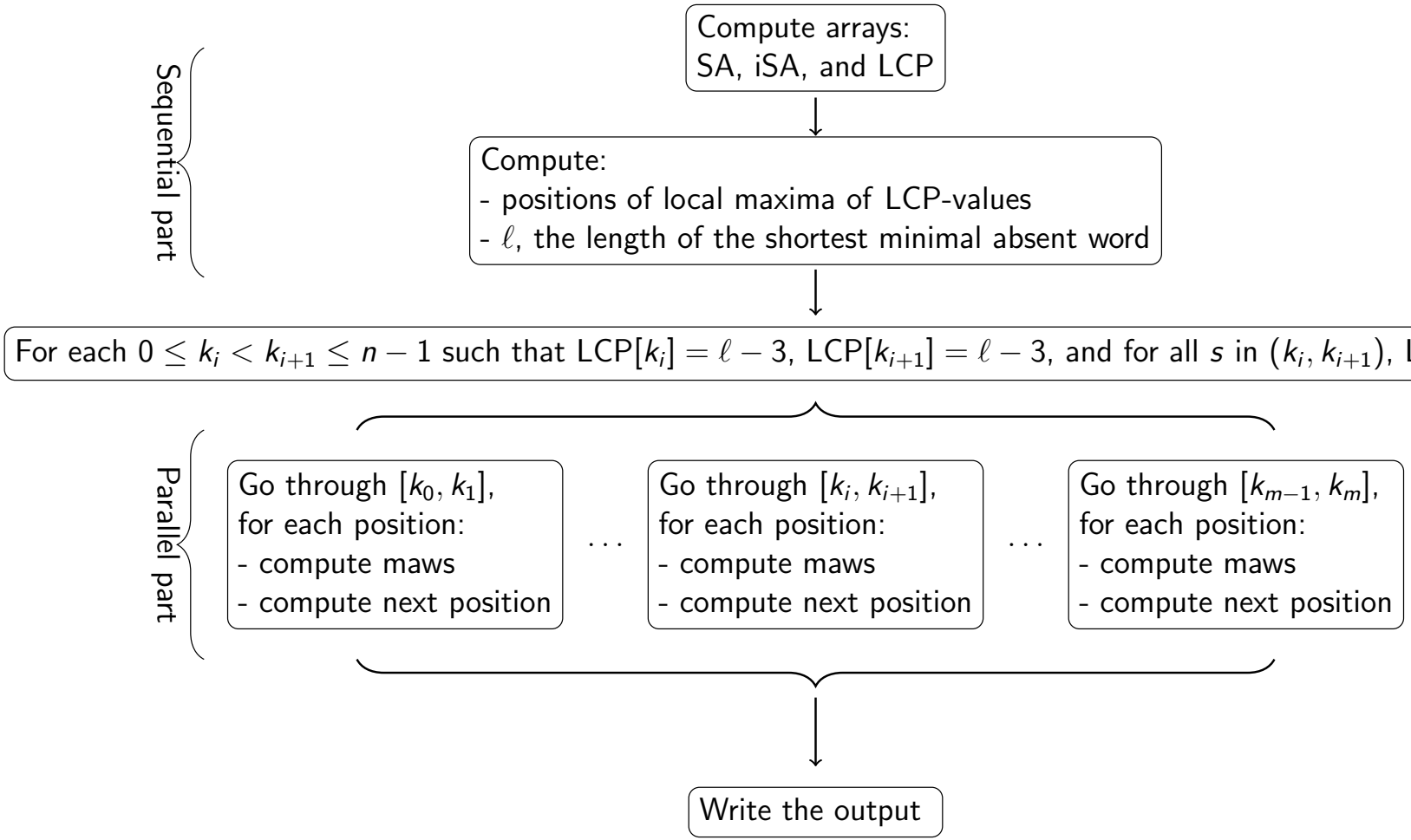
Sequential part
Compute arrays:
SA, iSA, and LCP
Compute:
- positions of local maxima of LCP-values
- ℓ, the length of the shortest minimal absent word
For each $0 \leq k_i < k_{i+1} \leq n-1$ such that $\mathrm{LCP}[k_i] = \ell - 3$, $\mathrm{LCP}[k_{i+1}] = \ell - 3$, and for all s in (k_i, k_{i+1}), $\mathrm{LCP}[s] \neq \ell - 3$
Parallel part
Go through $[k_0, k_1]$,
for each position:
- compute maws
- compute next position
...
Go through $[k_i, k_{i+1}]$,
for each position:
- compute maws
- compute next position
...
Go through $[k_{m-1}, k_m]$,
for each position:
- compute maws
- compute next position
Write the output

Table of Contents

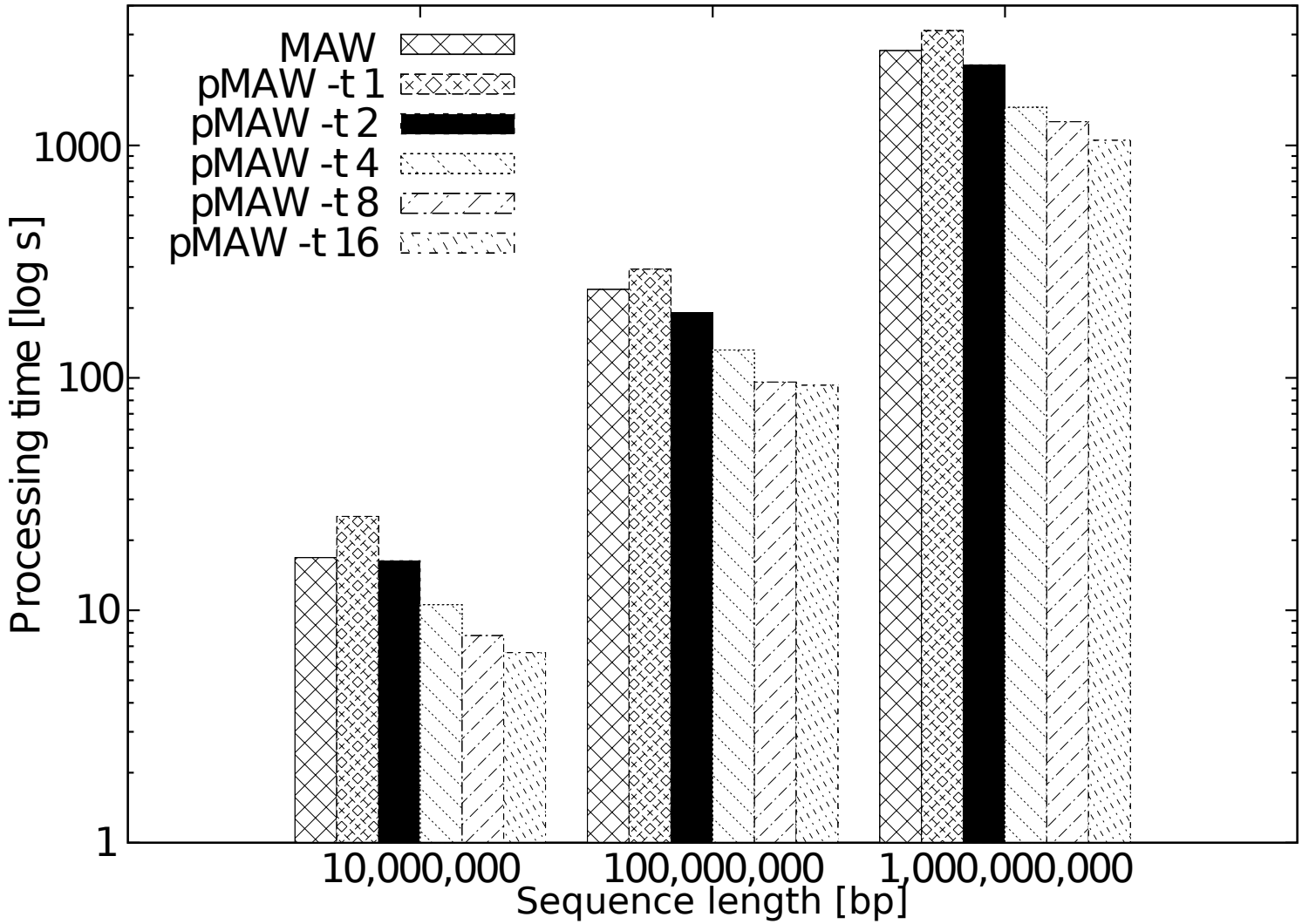

Figure: Elapsed-time comparison of pMAW and MAW for computing minimal absent words using synthetic DNA sequences

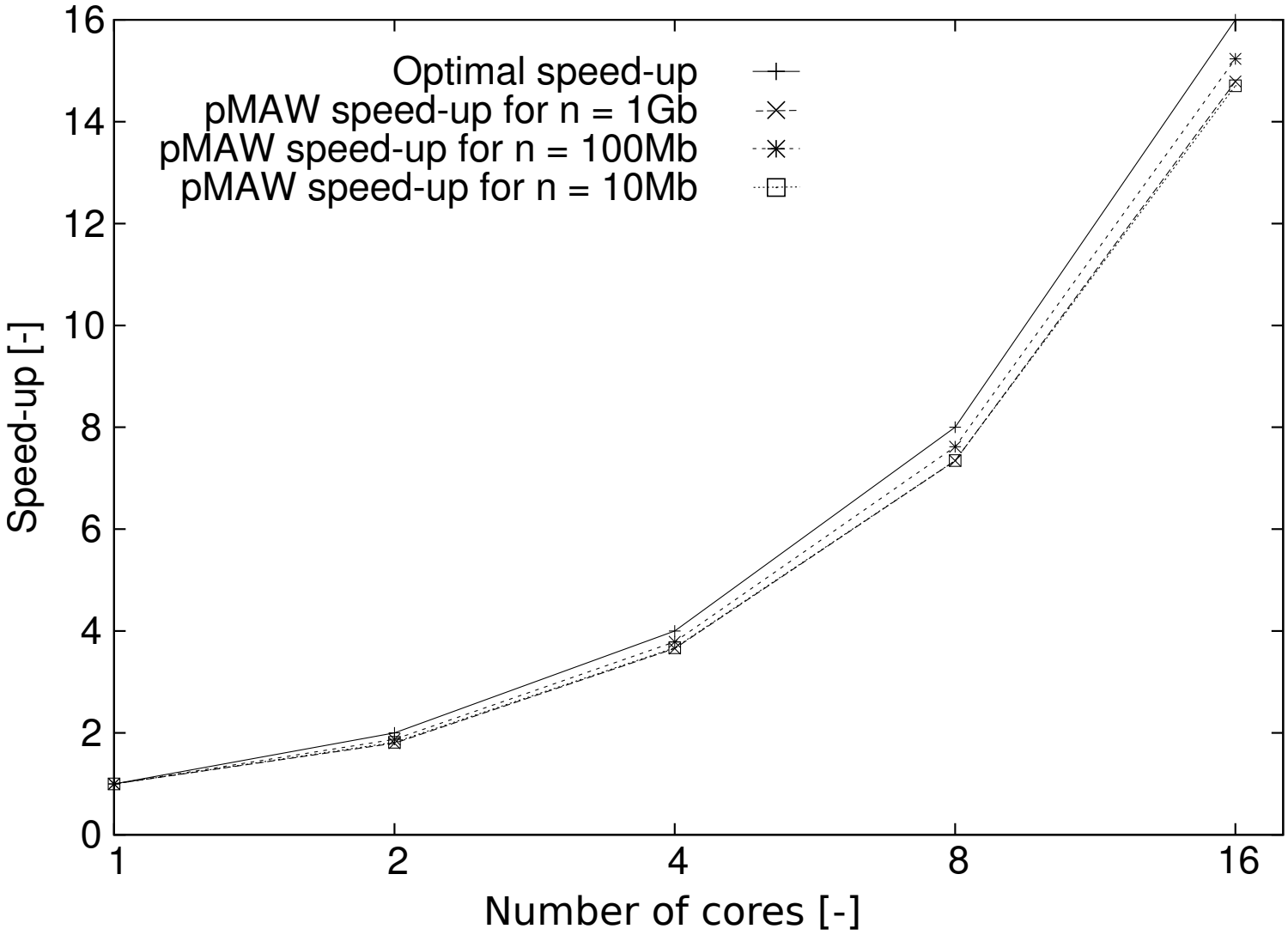

Figure: Relative speed-up of pMAW for computing minimal absent words using synthetic DNA sequences

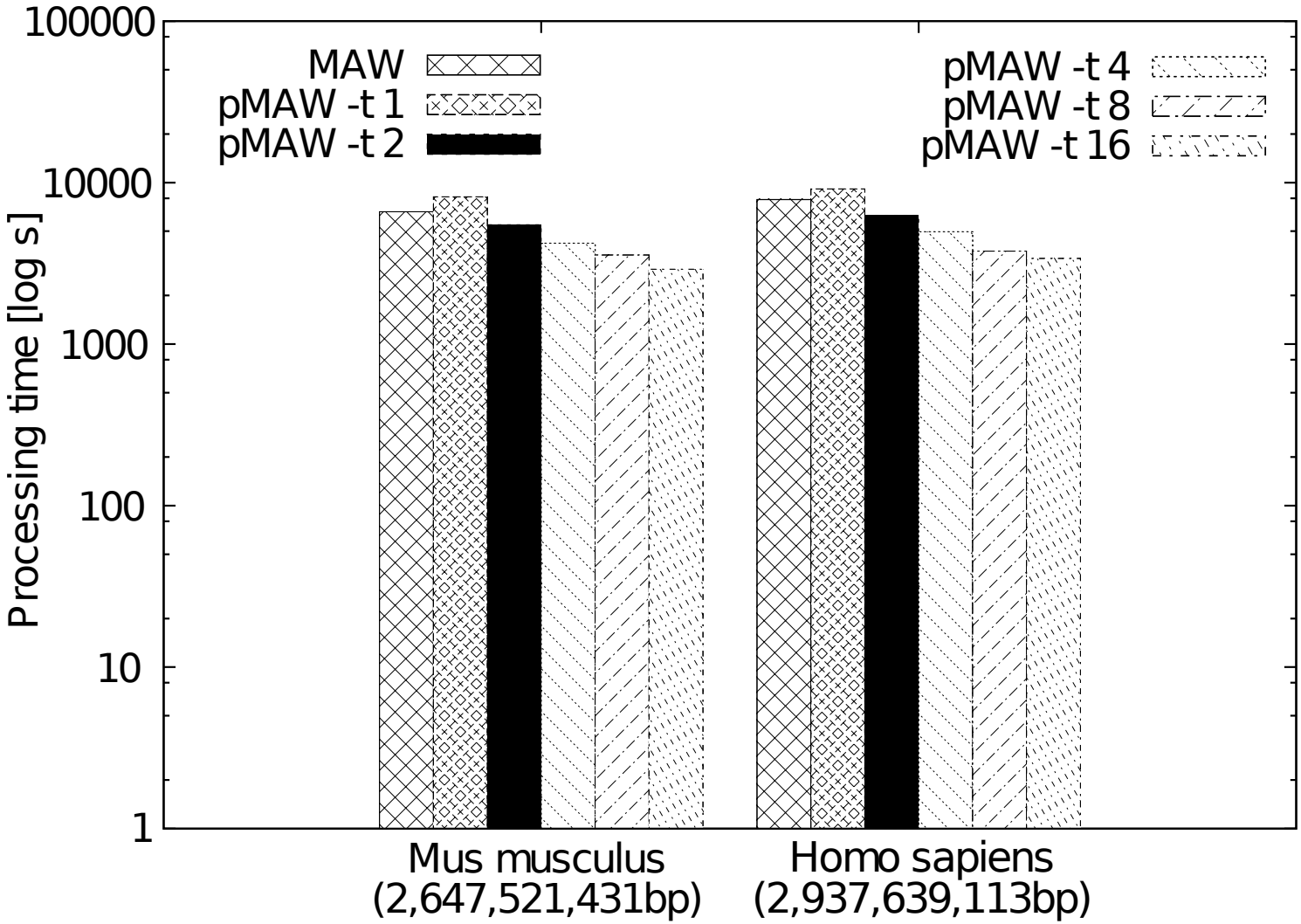

Figure: Elapsed-time comparison of pMAW and MAW for computing minimal absent words using real DNA sequences

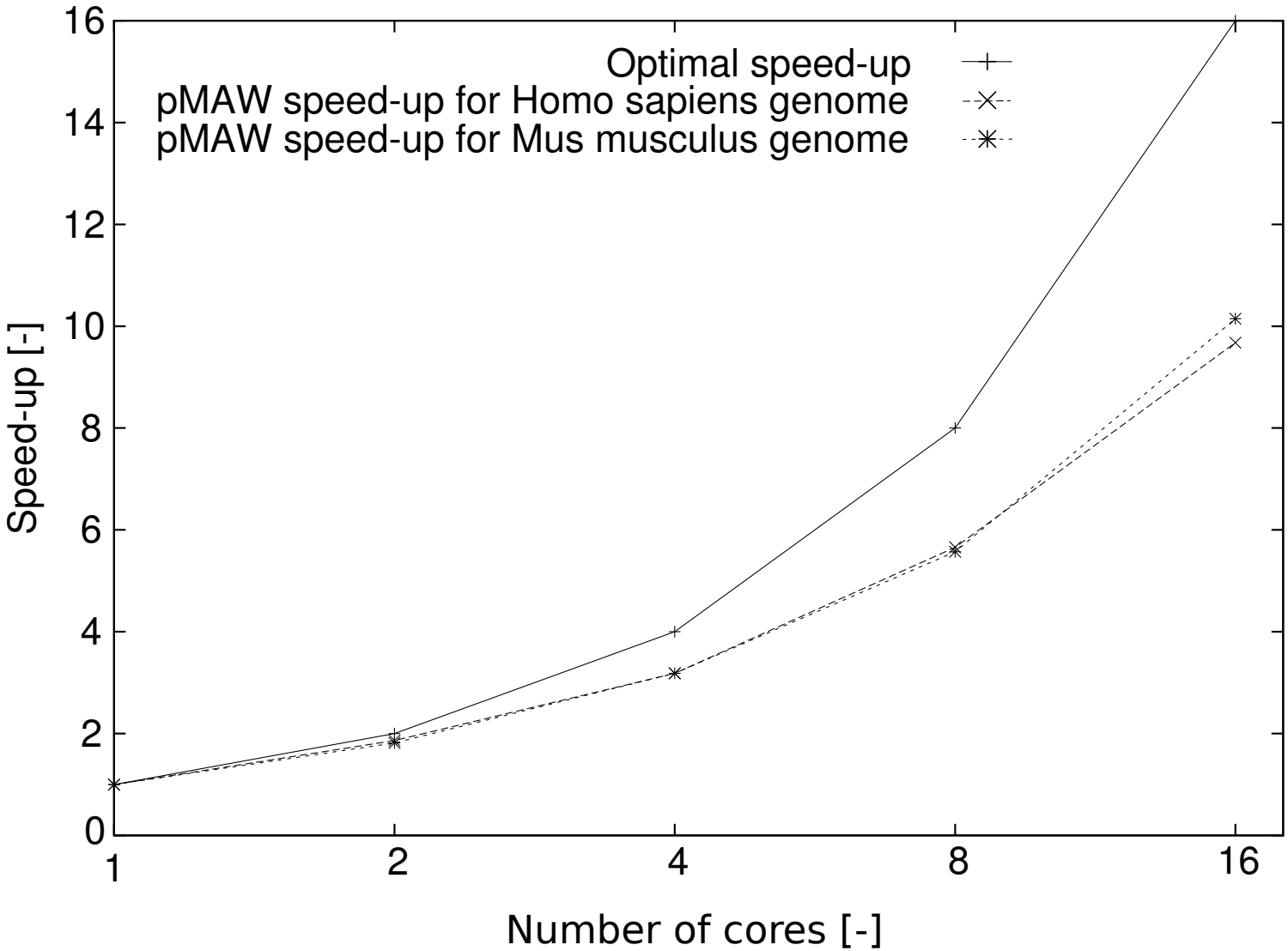

Figure: Relative speed-up of pMAW for computing minimal absent words using real DNA sequences

Our contribution

- A $\mathcal{O}(n)$-time and $\mathcal{O}(n)$-space algorithm for computing all minimal absent words based on the suffix array;
- An available implementation for shared-memory multiprocessing programming that outperforms existing tools for 2 or more cores:`http://github.com/solonas13/maw`

Our contribution

- A $\mathcal{O}(n)$-time and $\mathcal{O}(n)$-space algorithm for computing all minimal absent words based on the suffix array;
- An available implementation for shared-memory multiprocessing programming that outperforms existing tools for 2 or more cores:`http://github.com/solonas13/maw`

Perspectives

- Obtain a fully parallel algorithm by using parallel algorithms for constructing the suffix array and the longest common prefix array,