Average Complexity for Updating a Suffix Array

Martine Léonard, Laurent Mouchard, Mikaël Salson

LITIS, Université de Rouen

5 February 2010

Introduction

Suffix Array

- Index introduced in 1990.
- Matching a pattern of length m in a text T of length n in $O(m \log n)$ worst-case time (or $O(m + \log n)$ with a LCP array).
- Compressed suffix arrays (2000).

Introduction

Suffix Array

- Index introduced in 1990.
- Matching a pattern of length m in a text T of length n in $O(m \log n)$ worst-case time (or $O(m + \log n)$ with a LCP array).
- Compressed suffix arrays (2000).

Updating a suffix array when T is altered

- Gallé, Peterlongo, Coste (2008);
- Salson, Lecroq, Léonard, Mouchard (2009);
- Al-Hafeedh, Mouchard, Salson, Smyth.

Introduction

Suffix Array

- Index introduced in 1990.
- Matching a pattern of length m in a text T of length n in $O(m \log n)$ worst-case time (or $O(m + \log n)$ with a LCP array).
- Compressed suffix arrays (2000).

Updating a suffix array when T is altered

- Gallé, Peterlongo, Coste (2008);
- Salson, Lecroq, Léonard, Mouchard (2009);
- Al-Hafeedh, Mouchard, Salson, Smyth.

Question

- Why is it quicker than reconstructing the suffix array?[1]
- Do we reorder many suffixes?

[1] For a little number of modifications

Suffix Array

$$T = \begin{array}{cccccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ \mathsf{C} & \mathsf{G} & \mathsf{A} & \mathsf{G} & \mathsf{A} & \mathsf{C} & \mathsf{G} & \mathsf{A} & \mathsf{A} & \$ \end{array}$$

Suffix Array

$$T = \begin{array}{cccccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ \mathsf{C} & \mathsf{G} & \mathsf{A} & \mathsf{G} & \mathsf{A} & \mathsf{C} & \mathsf{G} & \mathsf{A} & \mathsf{A} & \$ \end{array}$$

Sorted suffixes

Suffix Array

$$T = \begin{array}{cccccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ \mathrm{C} & \mathrm{G} & \mathrm{A} & \mathrm{G} & \mathrm{A} & \mathrm{C} & \mathrm{G} & \mathrm{A} & \mathrm{A} & \$ \\ &&&&&&&&& \uparrow \end{array}$$

Sorted suffixes

9 $

Suffix Array

$$T = \begin{array}{cccccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ \mathrm{C} & \mathrm{G} & \mathrm{A} & \mathrm{G} & \mathrm{A} & \mathrm{C} & \mathrm{G} & \mathrm{A} & \mathrm{A} & \$ \\ & & & & & & & & \uparrow & \end{array}$$

Sorted suffixes

9	$
8	A $

Suffix Array

$$T = \begin{array}{cccccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ \mathrm{C} & \mathrm{G} & \mathrm{A} & \mathrm{G} & \mathrm{A} & \mathrm{C} & \mathrm{G} & \mathrm{A} & \mathrm{A} & \$ \\ & & & & & & & \uparrow & & \end{array}$$

	Sorted suffixes
9	$
8	A $
7	A A $

Suffix Array

$T =$

0	1	2	3	4	5	6	7	8	9
C	G	A	G	A	C	G	A	A	$
				↑					

	Sorted suffixes
9	$
8	A $
7	A A $
4	A C G A A $

Suffix Array

$$T = \begin{array}{cccccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ \mathrm{C} & \mathrm{G} & \mathrm{A} & \mathrm{G} & \mathrm{A} & \mathrm{C} & \mathrm{G} & \mathrm{A} & \mathrm{A} & \$ \\ & & \uparrow & & & & & & & \end{array}$$

	Sorted suffixes
9	$
8	A $
7	A A $
4	A C G A A $
2	A G A C G A A $

Suffix Array

$$T = \begin{array}{cccccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ \mathsf{C} & \mathsf{G} & \mathsf{A} & \mathsf{G} & \mathsf{A} & \mathsf{C} & \mathsf{G} & \mathsf{A} & \mathsf{A} & \$ \\ & & & & & \uparrow & & & & \end{array}$$

Sorted suffixes

9	$
8	A $
7	A A $
4	A C G A A $
2	A G A C G A A $
5	C G A A $

Suffix Array

$$T = \begin{array}{cccccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ \mathrm{C} & \mathrm{G} & \mathrm{A} & \mathrm{G} & \mathrm{A} & \mathrm{C} & \mathrm{G} & \mathrm{A} & \mathrm{A} & \$ \\ \uparrow & & & & & & & & & \end{array}$$

	Sorted suffixes
9	$
8	A $
7	A A $
4	A C G A A $
2	A G A C G A A $
5	C G A A $
0	C G A G A C G A A $

Suffix Array

$$T = \begin{array}{cccccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ \mathrm{C} & \mathrm{G} & \mathrm{A} & \mathrm{G} & \mathrm{A} & \mathrm{C} & \mathrm{G} & \mathrm{A} & \mathrm{A} & \$ \\ & & & & & & \uparrow & & & \end{array}$$

	Sorted suffixes
9	$
8	A $
7	A A $
4	A C G A A $
2	A G A C G A A $
5	C G A A $
0	C G A G A C G A A $
6	G A A $

Suffix Array

$$T = \begin{array}{cccccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ \mathsf{C} & \mathsf{G} & \mathsf{A} & \mathsf{G} & \mathsf{A} & \mathsf{C} & \mathsf{G} & \mathsf{A} & \mathsf{A} & \$ \\ & & & \uparrow & & & & & & \end{array}$$

	Sorted suffixes
9	$
8	A $
7	A A $
4	A C G A A $
2	A G A C G A A $
5	C G A A $
0	C G A G A C G A A $
6	G A A $
3	G A C G A A $

Suffix Array

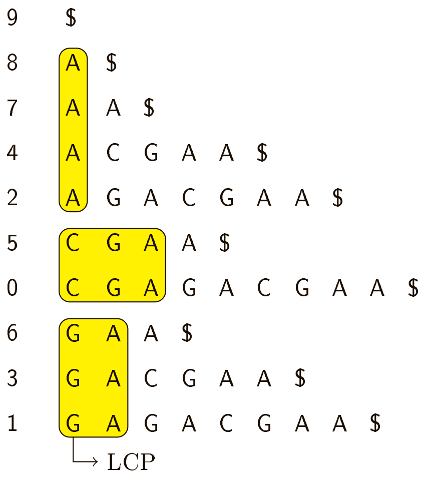
$$T = \begin{array}{cccccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ \mathrm{C} & \mathrm{G} & \mathrm{A} & \mathrm{G} & \mathrm{A} & \mathrm{C} & \mathrm{G} & \mathrm{A} & \mathrm{A} & \$ \\ & \uparrow & & & & & & & & \end{array}$$

	Sorted suffixes
9	$
8	A $
7	A A $
4	A C G A A $
2	A G A C G A A $
5	C G A A $
0	C G A G A C G A A $
6	G A A $
3	G A C G A A $
1	G A G A C G A A $

Suffix Array

$$T = \begin{array}{cccccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ C & G & A & G & A & C & G & A & A & \$ \end{array}$$

Sorted suffixes



9	$
8	A $
7	A A $
4	A C G A A $
2	A G A C G A A $
5	C G A A $
0	C G A G A C G A A $
6	G A A $
3	G A C G A A $
1	G A G A C G A A $

LCP

Suffix Array

$$T = \begin{array}{cccccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ \mathrm{C} & \mathrm{G} & \mathrm{A} & \mathrm{G} & \mathrm{A} & \mathrm{C} & \mathrm{G} & \mathrm{A} & \mathrm{A} & \$ \end{array}$$

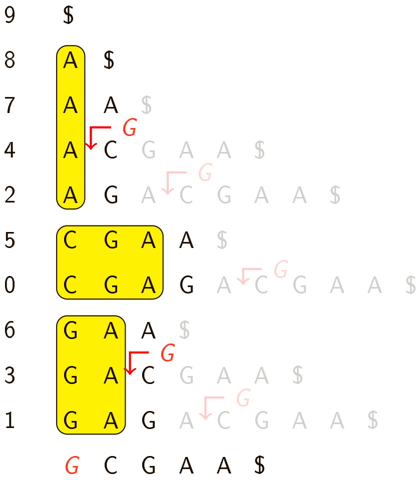
	Sorted suffixes
9	$
8	A $
7	A A $
4	A C G A A $
2	A G A C G A A $
5	C G A A $
0	C G A G A C G A A $
6	G A A $
3	G A C G A A $
1	G A G A C G A A $

Suffix Array

0	1	2	3	4	5	6	7	8	9	10
C	G	A	G	A	*G*	C	G	A	A	$

$T' =$ CGAGA*G*CGAA$

Sorted suffixes



9 $

8 A $

7 A A $

4 A C G A A $

2 A G A C G A A $

5 C G A A $

0 C G A G A C G A A $

6 G A A $

3 G A C G A A $

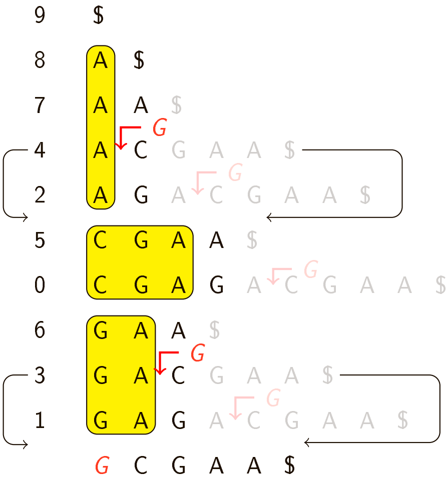
1 G A G A C G A A $

G C G A A $

Suffix Array

$$
\begin{array}{rccccccccccc}
 & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\
T' = & C & G & A & G & A & G & C & G & A & A & \$
\end{array}
$$

Sorted suffixes



9 $

8 A $

7 A A $

4 A C G A A $

2 A G A C G A A $

5 C G A A $

0 C G A G A C G A A $

6 G A A $

3 G A C G A A $

1 G A G A C G A A $

G C G A A $

Number of Suffixes Moved

Question

How many suffixes are not well ordered, after a modification in the text?

Number of Suffixes Moved

Question

How many suffixes are not well ordered, after a modification in the text?

A partial answer

In the worst case $n-1$ (*e.g.* A^n\$, insertion of a B: A^{n}B\$).

Number of Suffixes Moved

Question

How many suffixes are not well ordered, after a modification in the text?

A partial answer

In the worst case $n - 1$ (*e.g.* A^n\$, insertion of a B: A^{n}B\$).

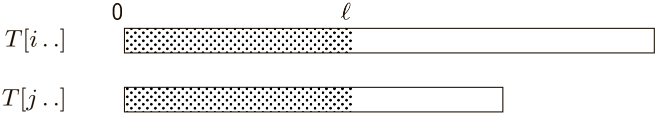
Remark

Depending on the LCP value, only few suffixes may be moved.

Number of Suffixes Moved

Idea

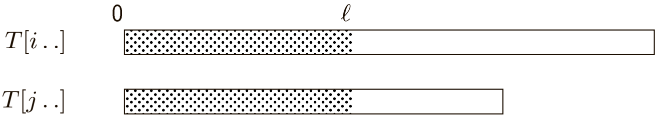
Let us consider two consecutive suffixes in the suffix array, and ℓ their LCP.



Number of Suffixes Moved

Idea

Let us consider two consecutive suffixes in the suffix array, and ℓ their LCP.

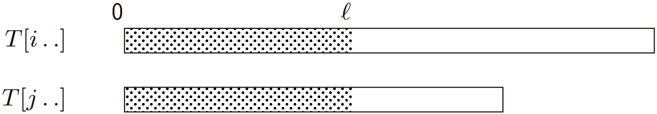


No more than ℓ suffixes will be moved if the text is modified at position $i+\ell+1$.

Number of Suffixes Moved

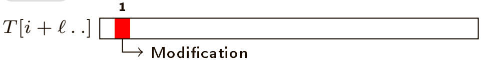
Idea

Let us consider two consecutive suffixes in the suffix array, and ℓ their LCP.



No more than ℓ suffixes will be moved if the text is modified at position $i + \ell + 1$.

Proof

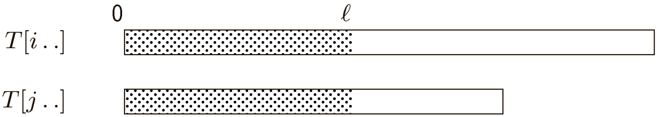


To be moved, LCP must be ≥ 1.

Number of Suffixes Moved

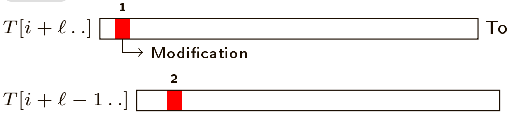
Idea

Let us consider two consecutive suffixes in the suffix array, and ℓ their LCP.



No more than ℓ suffixes will be moved if the text is modified at position $i+\ell+1$.

Proof



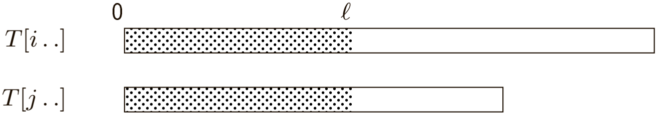
To be moved, LCP must be ≥ 1.

To be moved, LCP must be ≥ 2.

Number of Suffixes Moved

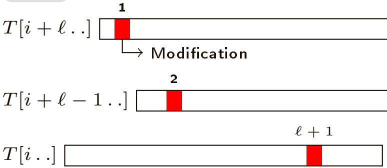
Idea

Let us consider two consecutive suffixes in the suffix array, and ℓ their LCP.



No more than ℓ suffixes will be moved if the text is modified at position $i+\ell+1$.

Proof



To be moved, LCP must be ≥ 1.

To be moved, LCP must be ≥ 2.

To be moved, LCP must be $\geq \ell+1$.

Number of Suffixes Moved

Let $r[i]$ be the maximal number of suffixes to be reordered when updating the text at position i.

Property

The r array is a permutation of the LCP array.

Corollary

The average number of suffixes moved when updating the suffix array is L_{ave}, the average LCP value of the text.

Average LCP Value

On average, L_{ave} suffixes are moved when updating the suffix array.

Average LCP Value

On average, L_{ave} suffixes are moved when updating the suffix array.

What do we know about L_{ave}?

- ranges from 0 to $n/2$, depending on texts;
- logarithmic for texts generated using a Markovian source of order one (Fayolle and Ward, 2005).

Average LCP Value

On average, L_{ave} suffixes are moved when updating the suffix array.

What do we know about L_{ave}?

- ranges from 0 to $n/2$, depending on texts;
- logarithmic for texts generated using a Markovian source of order one (Fayolle and Ward, 2005).

In practice

Texts indexed are usually:

- genome sequences;
- natural language texts.

L_{ave} on genome sequences

How repetitive are genomes?

Haubold and Wiehe (2006) classified genome sequences according to an index of repetitiveness.
Result: *Methylobacillus Flagellatus* is the most repeated one.

L_{ave} on genome sequences

How repetitive are genomes?

Haubold and Wiehe (2006) classified genome sequences according to an index of repetitiveness.
Result: *Methylobacillus Flagellatus* is the most repeated one.

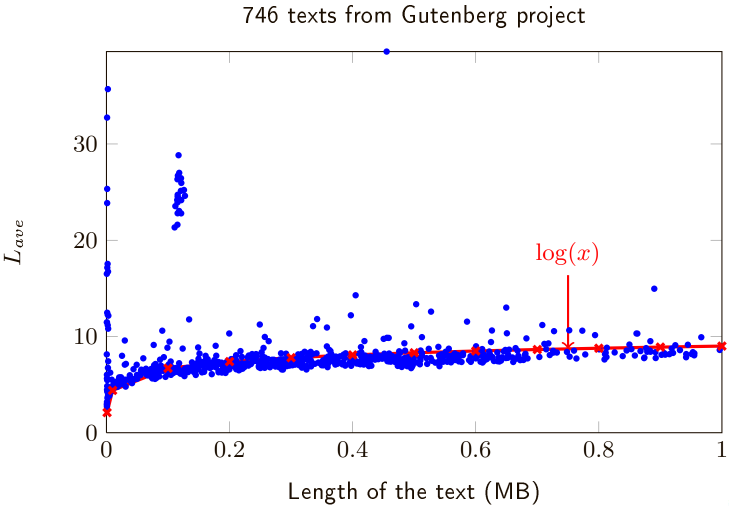
Results

Name	Length	L_{max}	L_{ave}
Most repeated sequences			
M. flagellatus	$2,971,519$	$143,034$	$3,452$
S. agalactiae	$2,211,485$	$47,068$	546
Sequences of interest			
D. melanogaster	$120,290,946$	$30,892$	66
C. elegans	$100,269,917$	$38,987$	45

L_{ave} for natural language texts

Plain texts

Digitalised books and texts from Gutenberg project.

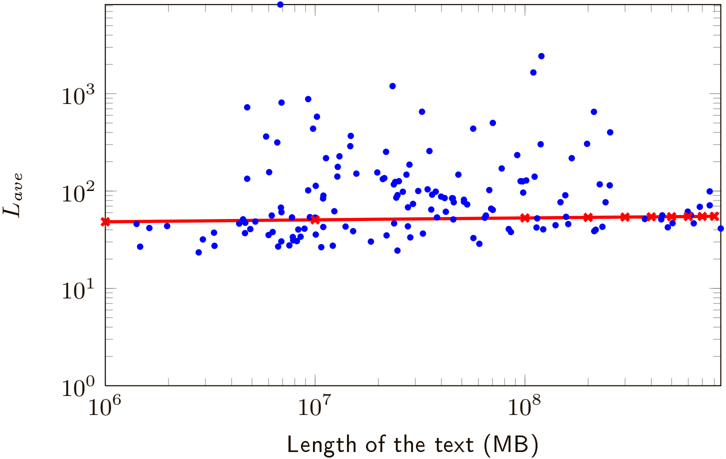


L_{ave} for natural language texts

Formatted texts

Wiki-formatted corpora from the Wikipedia encyclopedia.

151 corpora from the Wikipedia encyclopedia



Conclusion

Theoretically

- L_{ave} suffixes are moved on average, when updating a suffix array;
- L_{ave} is logarithmic for random texts.

Conclusion

Theoretically

- L_{ave} suffixes are moved on average, when updating a suffix array;
- L_{ave} is logarithmic for random texts.

Number of suffixes moved in practice

Conclusion

Theoretically

- L_{ave} suffixes are moved on average, when updating a suffix array;
- L_{ave} is logarithmic for random texts.

Number of suffixes moved in practice

- $\simeq 1/1000$, for pathological cases; much less in other cases;

Conclusion

Theoretically

- L_{ave} suffixes are moved on average, when updating a suffix array;
- L_{ave} is logarithmic for random texts.

Number of suffixes moved in practice

- $\simeq 1/1000$, for pathological cases; much less in other cases;
- logarithmic for plain natural-language texts.