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 = \text{CGAGACGAA}\$
Suffix Array

\[ T = \text{CGAGACGAA}\$ \]

Sorted suffixes
Suffix Array

\[ T = \text{CGAGACGAA} \]

Sorted suffixes

\[ 0 \ 1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \ 8 \ 9 \]

\[ 9 \ \$ \]
Suffix Array

\[ T = \text{CGAGACGAA}$ \]

Sorted suffixes

9 \quad \$

8 \quad A \quad$

LCP
Suffix Array

$T = \text{CGAGACGA}$

Sorted suffixes:

9  $ $
8  A  $ $
7  A  A  $ $

Suffix Array

\[ T = \text{CGAGACGA}$ \]

Sorted suffixes

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

Average Complexity for Updating a Suffix Array
Suffix Array

\[ T = \begin{align*} 0 & \quad 1 & \quad 2 & \quad 3 & \quad 4 & \quad 5 & \quad 6 & \quad 7 & \quad 8 & \quad 9 \\ C & \quad G & \quad A & \quad G & \quad A & \quad C & \quad G & \quad A & \quad A & \quad $ \end{align*} \]

Sorted suffixes

9  $  
8  A  $  
7  A  A  $  
4  A  C  G  A  A  $  
2  A  G  A  C  G  A  A  $
Suffix Array

\[ T = \text{CGAGA} \uparrow \text{CGAA} $ \]

Sorted suffixes

9  $ $
8  A $ $
7  A A $ $
4  ACGAA $ $
2  AGACAAGAA $ $
5  CGA $ $

Average Complexity for Updating a Suffix Array

Léonard, Mouchard, Salson
**Suffix Array**

\[ T = \ C G A G A C G A A $ \]

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 = \text{CGAGACA$AA}$

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

$$T = \text{CGAGACGAA}$$

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

\[ T = \text{CGAGACGAA} \]

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  $
Average Complexity for Updating a Suffix Array

### Suffix Array

\[ T = \text{CGAGACGAA$} \]

Sorted suffixes

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

LCP
Suffix Array

\[ T = \text{CGAGACGAA} \]$ 

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

\[ T' = \text{CGAGAGCAGA$} \]

Sorted suffixes

<table>
<thead>
<tr>
<th>Sorted suffixes</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
</tr>
<tr>
<td>8</td>
</tr>
<tr>
<td>7</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>G</td>
</tr>
</tbody>
</table>
Suffix Array

$$T' = \text{C G A G A G C G A A$}$$

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^nB\$).
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^nB\$).

**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 \( \ell \) their LCP.

\[
\begin{array}{c|c|c}
0 & \ell \\
T[i ..] & \text{[dotted]} & \text{[dotted]} \\
T[j ..] & \text{[dotted]} & \text{[dotted]}
\end{array}
\]
Number of Suffixes Moved

Idea

Let us consider two consecutive suffixes in the suffix array, and \( \ell \) their LCP.

\[
\begin{array}{c}
T[i \ldots] \\
\hline
0 & \cdots & \ell \\
T[j \ldots]
\end{array}
\]

No more than \( \ell \) 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 \( \ell \) their LCP.

\[
\begin{array}{c}
T[i..] \\
T[j..]
\end{array}
\]

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

Proof

\[
T[i + \ell ..] \quad \text{To be moved, LCP must be } \geq 1.
\]

\[\text{Modification}\]
Number of Suffixes Moved

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

\[ T[i..] \]

\[ T[j..] \]

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

**Proof**

\[ T[i + ℓ ..] \]

To be moved, LCP must be ≥ 1.

\[ T[i + ℓ − 1 ..] \]

To be moved, LCP must be ≥ 2.
Number of Suffixes Moved

Idea

Let us consider two consecutive suffixes in the suffix array, and \( \ell \) their LCP.

\[
\begin{array}{c}
T[i..] \\
T[j..]
\end{array}
\]

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

Proof

\[
\begin{array}{c}
T[i + \ell ..] \\
T[i + \ell - 1 ..] \\
T[i ..]
\end{array}
\]

To be moved, LCP must be \( \geq 1 \).

To be moved, LCP must be \( \geq 2 \).

To be moved, LCP must be \( \geq \ell + 1 \).
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.
On average, $L_{ave}$ suffixes are moved when updating the suffix array.
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).
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.
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.

<table>
<thead>
<tr>
<th>Name</th>
<th>Length</th>
<th>$L_{\text{max}}$</th>
<th>$L_{\text{ave}}$</th>
<th>Sequences of interest</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>M. agellatus</em></td>
<td>2,971</td>
<td>1,519</td>
<td>1,034</td>
<td></td>
</tr>
<tr>
<td><em>S. agalactiae</em></td>
<td>2,211</td>
<td>1,485</td>
<td>706</td>
<td></td>
</tr>
<tr>
<td><em>D. melanogaster</em></td>
<td>1,290</td>
<td>946</td>
<td>389</td>
<td></td>
</tr>
<tr>
<td><em>C. elegans</em></td>
<td>1,269</td>
<td>917</td>
<td>387</td>
<td></td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Name</th>
<th>Length</th>
<th>$L_{max}$</th>
<th>$L_{ave}$</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Most repeated sequences</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><em>M. flagellatus</em></td>
<td>2,971,519</td>
<td>143,034</td>
<td>3,452</td>
</tr>
<tr>
<td><em>S. agalactiae</em></td>
<td>2,211,485</td>
<td>47,068</td>
<td>546</td>
</tr>
<tr>
<td><strong>Sequences of interest</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><em>D. melanogaster</em></td>
<td>120,290,946</td>
<td>30,892</td>
<td>66</td>
</tr>
<tr>
<td><em>C. elegans</em></td>
<td>100,269,917</td>
<td>38,987</td>
<td>45</td>
</tr>
</tbody>
</table>
Average Complexity for Updating a Suffix Array

$L_{ave}$ for natural language texts

Digitalised books and texts from Gutenberg project.
For natural language texts,

Wiki-formatted corpora from the Wikipedia encyclopedia.

151 corpora from the Wikipedia encyclopedia

\( L_{ave} \) vs. Length of the text (MB)
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

$\approx \frac{1}{1000}$, for pathological cases; much less in other cases; logarithmic for plain natural-language 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**

- $\approx 1/1000$, 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

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