Minimal Absent Words

Alice Heliou

October 13, 2015

Outline

'Negative' information

Principle

Given a sequence of letters, we focus on words that don't occur.
Their absence may have a signification.

'Negative' information

Principle

Given a sequence of letters, we focus on words that don't occur. Their absence may have a signification.

Example

In a random sequence S, we expect that every word of size less than $\log_\sigma(|S|)$ occurs in S.

'Negative' information

Principle

Given a sequence of letters, we focus on words that don't occur.
Their absence may have a signification.

Example

In a random sequence S, we expect that every word of size less than $\log_\sigma(|S|)$ occurs in S.
The human genome contains around 3G nucleotides (A, C, G, T).
Yet some words of size 11, are absent $(11 < \log_4(3 * 10^9) = 15,7)$

'Negative' information

Application

Three minimal sequences found in Ebola virus genomes and absent from human DNA, [Silva et al.], 2015

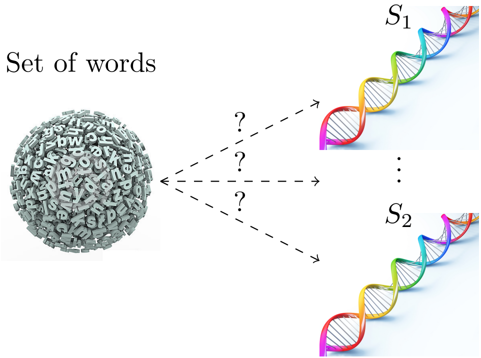
3 small sequences (of length between 12 and 14) that appear in the Ebola genome as coding for proteins, are absent from the Human genome.

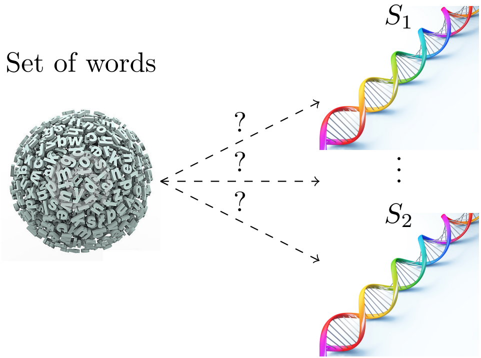
This was done by analyzing 99 virus and the Human genome reference GRC-38.

Sequence

Unicity $\longrightarrow$

Set of Absent Words


Set of words
?
?
?
S_1
$\vdots$
S_2



Property

For each set of words $\mathcal{M}$ if there exists a sequence $\mathcal{S}$ such that $\mathcal{M}$ is its set of absent words, then $\mathcal{S}$ is unique.

Absent words are too numerous

The number of absent words from a sequence of size n is **exponential** in n.

There are at most two words of size $n-1$ that occur in $S \Rightarrow$ at least $\sigma^{n-1} - 2$ absent words of size $n-1$
with σ the size of the alphabet.

Outline

Definition : Minimal Absent Word

A minimal absent word of a sequence is an absent word whose proper factors (longest prefix, and longest suffix) all occur in the sequence.

An upper bound on the number of minimal absent words is $\mathcal{O}(\sigma n)$.

Crochemore et al. 1998, Mignosi et al. 2002

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

Definition : Minimal Absent Word

A minimal absent word of a sequence is an absent word whose proper factors (longest prefix, and longest suffix) all occur in the sequence.

An upper bound on the number of minimal absent words is $\mathcal{O}(\sigma n)$.

Crochemore et al. 1998, Mignosi et al. 2002

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

Definition : Minimal Absent Word

A minimal absent word of a sequence is an absent word whose proper factors (longest prefix, and longest suffix) all occur in the sequence.

An upper bound on the number of minimal absent words is $\mathcal{O}(\sigma n)$.

Crochemore et al. 1998, Mignosi et al. 2002

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

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

Definition : Minimal Absent Word

A minimal absent word of a sequence is an absent word whose proper factors (longest prefix, and longest suffix) all occur in the sequence.

An upper bound on the number of minimal absent words is $\mathcal{O}(\sigma n)$.

Crochemore et al. 1998, Mignosi et al. 2002

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

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

Definition : Minimal Absent Word

A minimal absent word of a sequence is an absent word whose proper factors (longest prefix, and longest suffix) all occur in the sequence.

An upper bound on the number of minimal absent words is $\mathcal{O}(\sigma n)$.

Crochemore et al. 1998, Mignosi et al. 2002

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

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

Definition : Minimal Absent Word

A minimal absent word of a sequence is an absent word whose proper factors (longest prefix, and longest suffix) all occur in the sequence.

An upper bound on the number of minimal absent words is $\mathcal{O}(\sigma n)$.

Crochemore et al. 1998, Mignosi et al. 2002

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

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

Definition : Minimal Absent Word

A minimal absent word of a sequence is an absent word whose proper factors (longest prefix, and longest suffix) all occur in the sequence.

An upper bound on the number of minimal absent words is $\mathcal{O}(\sigma n)$.

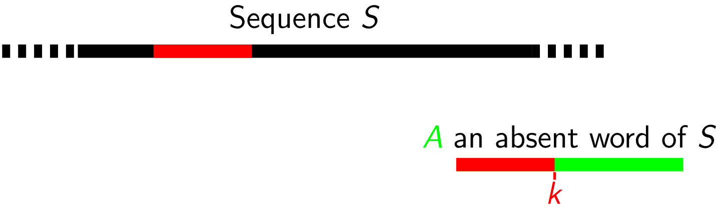
Crochemore et al. 1998, Mignosi et al. 2002

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

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

An absent word has a minimal absent word as factor

An absent word has a minimal absent word as factor

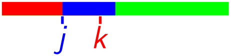


k, such that $A[0..k]$ occurs in S but not $A[0..k+1]$

An absent word has a minimal absent word as factor

Sequence S

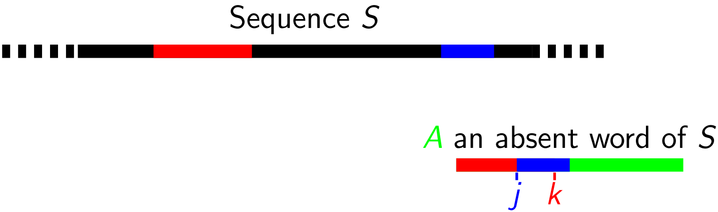
A an absent word of S



k, such that $A[0..k]$ occurs in S but not $A[0..k+1]$

j, such that $A[j..k+1]$ occurs S but not A[j-1..k+1]

An absent word has a minimal absent word as factor



k, such that $A[0..k]$ occurs in S but not $A[0..k+1]$

j, such that $A[j..k+1]$ occurs S but not A[j-1..k+1]

A[j-1..k+1] is a minimal absent word of S

because $A[j..k+1]$ and $A[j-1..k]$ occur in S.

Retrieving the sequence from its set of minimal absent words

Retrieving a sequence from its set of minimal absent words can be done in linear time $\Rightarrow$ Gabriele Fici thesis Minimal Forbidden Words and Applications (2006).

Retrieving the sequence from its set of minimal absent words

Retrieving a sequence from its set of minimal absent words can be done in linear time $\Rightarrow$ Gabriele Fici thesis Minimal Forbidden Words and Applications (2006).

Why can we retrieve it ?

Retrieving the sequence from its set of minimal absent words

Retrieving a sequence from its set of minimal absent words can be done in linear time $\Rightarrow$ Gabriele Fici thesis Minimal Forbidden Words and Applications (2006).

Why can we retrieve it ?

$$\begin{tikzcd}
\boxed{\text{Set of Minimal Absent Words}} \arrow[d, "\text{Append letters}"'] \\
\boxed{\text{Set of Absent Words}} \arrow[d, "\text{Language that forbids these words}"'] \\
\boxed{\text{Sequence}}
\end{tikzcd}$$

Definition: Maximal repeated pair

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

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

Definition: Maximal repeated pair

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

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

Lemma

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

Definition: Maximal repeated pair

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

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

Lemma

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

Sequence S

A a minimal absent word of S

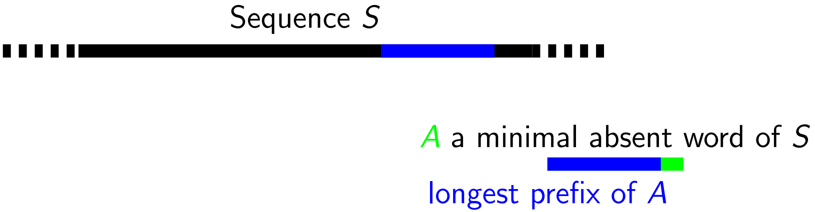
Definition: Maximal repeated pair

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

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

Lemma

If *awb* is a minimal absent word of S, then there exist positions i and j such that (i, j, w) is a maximal repeated pair of S.



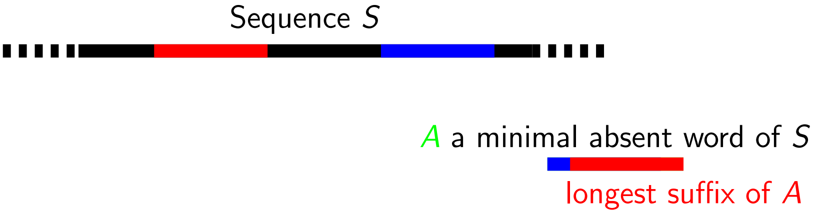
Definition: Maximal repeated pair

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

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

Lemma

If *awb* is a minimal absent word of S, then there exist positions i and j such that (i, j, w) is a maximal repeated pair of S.



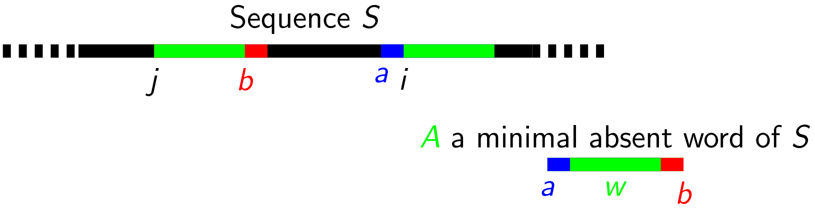
Definition: Maximal repeated pair

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

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

Lemma

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



Lemma

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

What is an upper bound of the number of maximal repeated pairs of a sequence of size n ?

Lemma

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

What is an upper bound of the number of maximal repeated pairs of a sequence of size n ?
$\mathcal{O}(n)$

Lemma

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

What is an upper bound of the number of maximal repeated pairs of a sequence of size n ?
$\mathcal{O}(n)$

How to find all maximal repeated pairs ?

Lemma

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

What is an upper bound of the number of maximal repeated pairs of a sequence of size n ?
$\mathcal{O}(n)$

How to find all maximal repeated pairs ?
By sorting all the suffixes of S, $\Rightarrow$ Suffix Array

Suffix Array by Manber& Myers in 1990

Table containing the starting position of the suffixes when they are in alphabetical order. It allows fast localisation of patterns.

Suffix Array by Manber& Myers in 1990

Table containing the starting position of the suffixes when they are in alphabetical order. It allows fast localisation of patterns.

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

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

Suffixes of y

Ordered suffixes of S

Suffix Array by Manber& Myers in 1990

Table containing the starting position of the suffixes when they are in alphabetical order. It allows fast localisation of patterns.

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

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

Suffixes of y

$\Rightarrow$

pos	
8	#

Ordered suffixes of S

Suffix Array by Manber& Myers in 1990

Table containing the starting position of the suffixes when they are in alphabetical order. It allows fast localisation of patterns.

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

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

Suffixes of y

$\Rightarrow$

pos									
8	#								
0	A	A	C	A	C	A	C	C	#

Ordered suffixes of S

Suffix Array by Manber& Myers in 1990

Table containing the starting position of the suffixes when they are in alphabetical order. It allows fast localisation of patterns.

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

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

Suffixes of y

$\Rightarrow$

pos									
8	#								
0	A	A	C	A	C	A	C	C	#
1	A	C	A	C	A	C	C	#	

Ordered suffixes of S

Suffix Array by Manber& Myers in 1990

Table containing the starting position of the suffixes when they are in alphabetical order. It allows fast localisation of patterns.

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

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

Suffixes of y

$\Rightarrow$

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

Ordered suffixes of S

Suffix Array by Manber& Myers in 1990

Table containing the starting position of the suffixes when they are in alphabetical order. It allows fast localisation of patterns.

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

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

Suffixes of y

$\Rightarrow$

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

Ordered suffixes of S

Suffix Array by Manber& Myers in 1990

Table containing the starting position of the suffixes when they are in alphabetical order. It allows fast localisation of patterns.

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

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

Suffixes of y

$\Rightarrow$

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

Ordered suffixes of S

Suffix Array by Manber& Myers in 1990

Table containing the starting position of the suffixes when they are in alphabetical order. It allows fast localisation of patterns.

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

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

Suffixes of y

$\Rightarrow$

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

Ordered suffixes of S

Computation of minimal absent words

Pre-computation

Construction :

- Suffix Array, linear time and space since 2003
- Longest Common Prefix table, linear time and space with the SA and the sequence as input

Computation

- Travel twice through those tables, in order to construct the set of letters that occurs just before each right-maximal repetition.
- Deduce the set of minimal absent words.

Applications

Biology

- Linear-Time Sequence Comparison Using Minimal Absent Words & Applications, [Crochemore 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

Outline

1. Introduction
2. Minimal Absent Words
3. Computation of Minimal Absent Words

Before 2014

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

Constants reduction

References	Time for fixed size alphabet	Space	Structure
Ota et al. 2014 Dynamic construction of an antidictionary with linear complexity **Theoretical Computer Science**	$\mathcal{O}(n)$	$\mathcal{O}(n)$	suffix tree, dynamic approach
Barton et al. 2014 Linear-time computation of minimal absent words **BMC Bioinfo**	$\mathcal{O}(n)$	$\mathcal{O}(n)$	suffix array
Belazzougui et al. 2015 Space-efficient detection of unusual words. **CPM**	randomized $\mathcal{O}(n)$	$\mathcal{O}(n)$	BWT & few additional structures
Barton et al. 2015 Engineering the Computation of Minimal Absent Words. **PPAM**	$\mathcal{O}(n/p)$	$\mathcal{O}(n)$	suffix array

Perspectives

- Use external memory computation
  to find a trade-off between running time and RAM usage.

- Knowing the set of minimal absent words of a sequence,
  deduce the set of a circular shift.

Thank you

Thank you

Questions ?

Linear time construction of the suffix array

over integer alphabets

The skew algorithm, Karkkainen and Sanders 2003

Main idea : Divide suffixes into 2 groups :

- Those starting a position $i \not\equiv 0 \mod 3$
- Those starting a position $i \equiv 0 \mod 3$

Linear time construction of the suffix array

over integer alphabets

The skew algorithm, Karkkainen and Sanders 2003

Main idea : Divide suffixes into 2 groups :

- Those starting a position $i \not\equiv 0 \mod 3$
- Those starting a position $i \equiv 0 \mod 3$

	0	1	2	3	4	5	6	7	8	9	10	11	12
S=	M	I	S	S	I	S	S	I	P	P	I	#	#

Linear time construction of the suffix array

over integer alphabets

The skew algorithm, Karkkainen and Sanders 2003

Main idea : Divide suffixes into 2 groups :

- Those starting a position $i \not\equiv 0 \mod 3$
- Those starting a position $i \equiv 0 \mod 3$

	0	1	2	3	4	5	6	7	8	9	10	11	12
$S=$	M	I	S	S	I	S	S	I	P	P	I	#	#

Outline :

- recursively sort the suffixes of the first group
- merge with the second group

first step, recursively sort the suffixes starting at $i \not\equiv 0 \mod 3$

- Consider all the triples of starting at positions $i \not\equiv 0 \mod 3$.

first step, recursively sort the suffixes starting at $i \not\equiv 0 \mod 3$

- Consider all the triples of starting at positions $i \not\equiv 0 \mod 3$.
- Assign them lexicographical names $\Rightarrow S'$
  This can be done in linear time by radix sort and recursion
  - if some of them get the same lexicographic name, we compute recursively the suffix array of the string S^{12}

first step, recursively sort the suffixes starting at $i \not\equiv 0 \mod 3$

- Consider all the triples of starting at positions $i \not\equiv 0 \mod 3$.
- Assign them lexicographical names $\Rightarrow S'$
  This can be done in linear time by radix sort and recursion
  - if some of them get the same lexicographic name, we compute recursively the suffix array of the string S^{12}
- Once all the triples are ordered, we have the ordering of the suffixes starting at $i \not\equiv 0 \mod 3$

handling 0 suffixes

- Sort the group 0 suffixes, using the representation (S[i], S_{i+1})

handling 0 suffixes

- Sort the group 0 suffixes, using the representation (S[i], S_{i+1})
- Merge the two ordered groups by comparing 0-suffix S_j with 1 or 2-suffix S_i

handling 0 suffixes

- Sort the group 0 suffixes, using the representation (S[i], S_{i+1})
- Merge the two ordered groups by comparing 0-suffix S_j with 1 or 2-suffix S_i
  - if $i \equiv 1 \mod 3$, we compare the pair representations (s[j], S_{j+1}) and (s[i],S_{i+1})

handling 0 suffixes

- Sort the group 0 suffixes, using the representation (S[i], S_{i+1})
- Merge the two ordered groups by comparing 0-suffix S_j with 1 or 2-suffix S_i
    - if $i \equiv 1 \mod 3$, we compare the pair representations (s[j], S_{j+1}) and (s[i],S_{i+1})
    - if $i \equiv 2 \mod 3$, we compare the triple representations (s[j], s[j+1] S_{j+2}) and (s[i], s[i+1], S_{i+2})