Suffixes, Conjugates and Lyndon words

Silvia Bonomo, Sabrina Mantaci, Antonio Restivo,
Giovanna Rosone and Marinella Sciortino

Dipartimento di Matematica e Informatica, University of Palermo
Palermo, ITALY

DLT 2013

Outline

1. The problem
2. SA, BWT and EBWT
3. Lyndon words and sortings
4. The algorithm for sorting conjugates
5. Sorting suffixes by Lyndon factorization
6. Applications

The sorting problem in text processing tools

The sorting process is an important step for the construction of important tools for text processing (applied in pattern matching, data compression, sequences comparison, etc):

The sorting problem in text processing tools

The sorting process is an important step for the construction of important tools for text processing (applied in pattern matching, data compression, sequences comparison, etc):

- The Suffix Array (SA);

The sorting problem in text processing tools

The sorting process is an important step for the construction of important tools for text processing (applied in pattern matching, data compression, sequences comparison, etc):

- The Suffix Array (SA);
- The Burrows Wheeler Transform (BWT)

The sorting problem in text processing tools

The sorting process is an important step for the construction of important tools for text processing (applied in pattern matching, data compression, sequences comparison, etc):

- The Suffix Array (SA);
- The Burrows Wheeler Transform (BWT)
- The Extended Burrows-Wheeler Transform (EBWT)

Different sorting relations

In spite of the closeness of these tools, the sorting processes involve different sorting relations on different objects:

Different sorting relations

In spite of the closeness of these tools, the sorting processes involve different sorting relations on different objects:

- SA $\rightarrow$ lexicographic order among suffixes of a single word;

Different sorting relations

In spite of the closeness of these tools, the sorting processes involve different sorting relations on different objects:

- SA → lexicographic order among suffixes of a single word;
- BWT → lexicographic order among conjugates of a single word;

Different sorting relations

In spite of the closeness of these tools, the sorting processes involve different sorting relations on different objects:

- SA $\rightarrow$ lexicographic order among suffixes of a single word;
- BWT $\rightarrow$ lexicographic order among conjugates of a single word;
- EBWT $\rightarrow$ $\preceq_\omega$ order among conjugates of a set of words.

Different sorting relations

In spite of the closeness of these tools, the sorting processes involve different sorting relations on different objects:

- SA $\rightarrow$ lexicographic order among suffixes of a single word;
- BWT $\rightarrow$ lexicographic order among conjugates of a single word;
- EBWT $\rightarrow$ $\preceq_\omega$ order among conjugates of a set of words.

The aim of this talk is to find the combinatorial connection among all these sorting processes

Preliminaries

- Let Σ denote a non-empty finite alphabet.
- A word w over an alphabet Σ is a finite sequence of letters of Σ. We denote by Σ^* the set of all words over Σ
- Given a finite word $w = a_1a_2 \cdots a_n$, $a_i \in \Sigma$, a factor of w is written as $w[i,j] = a_i \cdots a_j$. A factor $w[1,j]$ is called a prefix, while a factor $w[i,n]$ is called a suffix.
- Two words $u, v \in \Sigma^*$ are conjugate, if $u = xy$ and $v = yx$ for some $x, y \in \Sigma^*$.

Here, we denote by $suf_k(w)$ (resp. $pref_k(w)$) the suffix (resp. prefix) of w that has length k and by $conj_k(w)$ the conjugate of w starting at position $|w| - k + 1$.

Suffix array

Given a word $w \in \Sigma^*$, with $|w| = n$, the suffix array (SA) of w ($SA(w)$) is the permutation of $\{1, 2, \ldots, n\}$ giving the starting positions of the suffixes of w, sorted in lexicographic order.

Example

Given the word $w = ababbaa$ its suffix array is $SA(w) = [6, 7, 1, 3, 5, 2, 4]$

In order to get the suffix array one need to lexicographically sort its suffixes.

The Burrows Wheeler Transform

Given a word $w \in \Sigma^*$, the Burrows-Wheeler Transform of w, $bwt(w)$ is a permutation of the letters in w, obtained as concatenation of the last letters of the lexicographically sorted list of its conjugates.

The Burrows Wheeler Transform

Given a word $w \in \Sigma^*$, the Burrows-Wheeler Transform of w, $bwt(w)$ is a permutation of the letters in w, obtained as concatenation of the last letters of the lexicographically sorted list of its conjugates.

The Burrows Wheeler Transform

Given a word $w \in \Sigma^*$, the Burrows-Wheeler Transform of w, $bwt(w)$ is a permutation of the letters in w, obtained as concatenation of the last letters of the lexicographically sorted list of its conjugates.

$$i\ n\ t\ e\ r\ n\ a\ t\ i\ o\ n\ a\ l$$

The Burrows Wheeler Transform

Given a word $w \in \Sigma^*$, the Burrows-Wheeler Transform of w, $bwt(w)$ is a permutation of the letters in w, obtained as concatenation of the last letters of the lexicographically sorted list of its conjugates.

$$
\begin{array}{cccccccccccc}
i & n & t & e & r & n & a & t & i & o & n & a & l \\
n & t & e & r & n & a & t & i & o & n & a & l & i \\
t & e & r & n & a & t & i & o & n & a & l & i & n \\
e & r & n & a & t & i & o & n & a & l & i & n & t \\
r & n & a & t & i & o & n & a & l & i & n & t & e \\
n & a & t & i & o & n & a & l & i & n & t & e & r \\
a & t & i & o & n & a & l & i & n & t & e & r & n \\
t & i & o & n & a & l & i & n & t & e & r & n & a \\
i & o & n & a & l & i & n & t & e & r & n & a & t \\
o & n & a & l & i & n & t & e & r & n & a & t & i \\
n & a & l & i & n & t & e & r & n & a & t & i & o \\
a & l & i & n & t & e & r & n & a & t & i & o & n \\
l & i & n & t & e & r & n & a & t & i & o & n & a
\end{array}
$$

The Burrows Wheeler Transform

Given a word $w \in \Sigma^*$, the Burrows-Wheeler Transform of w, $bwt(w)$ is a permutation of the letters in w, obtained as concatenation of the last letters of the lexicographically sorted list of its conjugates.

		M
$i\ n\ t\ e\ r\ n\ a\ t\ i\ o\ n\ a\ l$	1	$a\ l\ i\ n\ t\ e\ r\ n\ a\ t\ i\ o\ n$
$n\ t\ e\ r\ n\ a\ t\ i\ o\ n\ a\ l\ i$	2	$a\ t\ i\ o\ n\ a\ l\ i\ n\ t\ e\ r\ n$
$t\ e\ r\ n\ a\ t\ i\ o\ n\ a\ l\ i\ n$	3	$e\ r\ n\ a\ t\ i\ o\ n\ a\ l\ i\ n\ t$
$e\ r\ n\ a\ t\ i\ o\ n\ a\ l\ i\ n\ t$	4	$i\ n\ t\ e\ r\ n\ a\ t\ i\ o\ n\ a\ l$
$r\ n\ a\ t\ i\ o\ n\ a\ l\ i\ n\ t\ e$	5	$i\ o\ n\ a\ l\ i\ n\ t\ e\ r\ n\ a\ t$
$n\ a\ t\ i\ o\ n\ a\ l\ i\ n\ t\ e\ r$	6	$l\ i\ n\ t\ e\ r\ n\ a\ t\ i\ o\ n\ a$
$a\ t\ i\ o\ n\ a\ l\ i\ n\ t\ e\ r\ n$	7	$n\ a\ l\ i\ n\ t\ e\ r\ n\ a\ t\ i\ o$
$t\ i\ o\ n\ a\ l\ i\ n\ t\ e\ r\ n\ a$	8	$n\ a\ t\ i\ o\ n\ a\ l\ i\ n\ t\ e\ r$
$i\ o\ n\ a\ l\ i\ n\ t\ e\ r\ n\ a\ t$	9	$n\ t\ e\ r\ n\ a\ t\ i\ o\ n\ a\ l\ i$
$o\ n\ a\ l\ i\ n\ t\ e\ r\ n\ a\ t\ i$	10	$o\ n\ a\ l\ i\ n\ t\ e\ r\ n\ a\ t\ i$
$n\ a\ l\ i\ n\ t\ e\ r\ n\ a\ t\ i\ o$	11	$r\ n\ a\ t\ i\ o\ n\ a\ l\ i\ n\ t\ e$
$a\ l\ i\ n\ t\ e\ r\ n\ a\ t\ i\ o\ n$	12	$t\ e\ r\ n\ a\ t\ i\ o\ n\ a\ l\ i\ n$
$l\ i\ n\ t\ e\ r\ n\ a\ t\ i\ o\ n\ a$	13	$t\ i\ o\ n\ a\ l\ i\ n\ t\ e\ r\ n\ a$

The Burrows Wheeler Transform

Given a word $w \in \Sigma^*$, the Burrows-Wheeler Transform of w, $bwt(w)$ is a permutation of the letters in w, obtained as concatenation of the last letters of the lexicographically sorted list of its conjugates.

$i\ n\ t\ e\ r\ n\ a\ t\ i\ o\ n\ a\ l$
$n\ t\ e\ r\ n\ a\ t\ i\ o\ n\ a\ l\ i$
$t\ e\ r\ n\ a\ t\ i\ o\ n\ a\ l\ i\ n$
$e\ r\ n\ a\ t\ i\ o\ n\ a\ l\ i\ n\ t$
$r\ n\ a\ t\ i\ o\ n\ a\ l\ i\ n\ t\ e$
$n\ a\ t\ i\ o\ n\ a\ l\ i\ n\ t\ e\ r$
$a\ t\ i\ o\ n\ a\ l\ i\ n\ t\ e\ r\ n$
$t\ i\ o\ n\ a\ l\ i\ n\ t\ e\ r\ n\ a$
$i\ o\ n\ a\ l\ i\ n\ t\ e\ r\ n\ a\ t$
$o\ n\ a\ l\ i\ n\ t\ e\ r\ n\ a\ t\ i$
$n\ a\ l\ i\ n\ t\ e\ r\ n\ a\ t\ i\ o$
$a\ l\ i\ n\ t\ e\ r\ n\ a\ t\ i\ o\ n$
$l\ i\ n\ t\ e\ r\ n\ a\ t\ i\ o\ n\ a$

M

	F ↓		L ↓
1	a	$l\ i\ n\ t\ e\ r\ n\ a\ t\ i\ o$	n
2	a	$t\ i\ o\ n\ a\ l\ i\ n\ t\ e\ r$	n
3	e	$r\ n\ a\ t\ i\ o\ n\ a\ l\ i\ n$	t
4	i	$n\ t\ e\ r\ n\ a\ t\ i\ o\ n\ a$	l
5	i	$o\ n\ a\ l\ i\ n\ t\ e\ r\ n\ a$	t
6	l	$i\ n\ t\ e\ r\ n\ a\ t\ i\ o\ n$	a
7	n	$a\ l\ i\ n\ t\ e\ r\ n\ a\ t\ i$	o
8	n	$a\ t\ i\ o\ n\ a\ l\ i\ n\ t\ e$	r
9	n	$t\ e\ r\ n\ a\ t\ i\ o\ n\ a\ l$	i
10	o	$n\ a\ l\ i\ n\ t\ e\ r\ n\ a\ t$	i
11	r	$n\ a\ t\ i\ o\ n\ a\ l\ i\ n\ t$	e
12	t	$e\ r\ n\ a\ t\ i\ o\ n\ a\ l\ i$	n
13	t	$i\ o\ n\ a\ l\ i\ n\ t\ e\ r\ n$	a

The rows of M are conjugate of v sorted in lexicographic order.

The Burrows Wheeler Transform

Given a word $w \in \Sigma^*$, the Burrows-Wheeler Transform of w, $bwt(w)$ is a permutation of the letters in w, obtained as concatenation of the last letters of the lexicographically sorted list of its conjugates.

Conjugates			M ($F \downarrow$ … $L \downarrow$)
$i\ n\ t\ e\ r\ n\ a\ t\ i\ o\ n\ a\ l$		1	$a\ l\ i\ n\ t\ e\ r\ n\ a\ t\ i\ o\ n$
$n\ t\ e\ r\ n\ a\ t\ i\ o\ n\ a\ l\ i$		2	$a\ t\ i\ o\ n\ a\ l\ i\ n\ t\ e\ r\ n$
$t\ e\ r\ n\ a\ t\ i\ o\ n\ a\ l\ i\ n$		3	$e\ r\ n\ a\ t\ i\ o\ n\ a\ l\ i\ n\ t$
$e\ r\ n\ a\ t\ i\ o\ n\ a\ l\ i\ n\ t$	$I \rightarrow$	4	$i\ n\ t\ e\ r\ n\ a\ t\ i\ o\ n\ a\ l$
$r\ n\ a\ t\ i\ o\ n\ a\ l\ i\ n\ t\ e$		5	$i\ o\ n\ a\ l\ i\ n\ t\ e\ r\ n\ a\ t$
$n\ a\ t\ i\ o\ n\ a\ l\ i\ n\ t\ e\ r$		6	$l\ i\ n\ t\ e\ r\ n\ a\ t\ i\ o\ n\ a$
$a\ t\ i\ o\ n\ a\ l\ i\ n\ t\ e\ r\ n$		7	$n\ a\ l\ i\ n\ t\ e\ r\ n\ a\ t\ i\ o$
$t\ i\ o\ n\ a\ l\ i\ n\ t\ e\ r\ n\ a$		8	$n\ a\ t\ i\ o\ n\ a\ l\ i\ n\ t\ e\ r$
$i\ o\ n\ a\ l\ i\ n\ t\ e\ r\ n\ a\ t$		9	$n\ t\ e\ r\ n\ a\ t\ i\ o\ n\ a\ l\ i$
$o\ n\ a\ l\ i\ n\ t\ e\ r\ n\ a\ t\ i$		10	$o\ n\ a\ l\ i\ n\ t\ e\ r\ n\ a\ t\ i$
$n\ a\ l\ i\ n\ t\ e\ r\ n\ a\ t\ i\ o$		11	$r\ n\ a\ t\ i\ o\ n\ a\ l\ i\ n\ t\ e$
$a\ l\ i\ n\ t\ e\ r\ n\ a\ t\ i\ o\ n$		12	$t\ e\ r\ n\ a\ t\ i\ o\ n\ a\ l\ i\ n$
$l\ i\ n\ t\ e\ r\ n\ a\ t\ i\ o\ n\ a$		13	$t\ i\ o\ n\ a\ l\ i\ n\ t\ e\ r\ n\ a$

The rows of M are conjugate of v sorted in lexicographic order.

The Burrows Wheeler Transform

Given a word $w \in \Sigma^*$, the Burrows-Wheeler Transform of w, $bwt(w)$ is a permutation of the letters in w, obtained as concatenation of the last letters of the lexicographically sorted list of its conjugates.

Conjugates
$i\ n\ t\ e\ r\ n\ a\ t\ i\ o\ n\ a\ l$
$n\ t\ e\ r\ n\ a\ t\ i\ o\ n\ a\ l\ i$
$t\ e\ r\ n\ a\ t\ i\ o\ n\ a\ l\ i\ n$
$e\ r\ n\ a\ t\ i\ o\ n\ a\ l\ i\ n\ t$
$r\ n\ a\ t\ i\ o\ n\ a\ l\ i\ n\ t\ e$
$n\ a\ t\ i\ o\ n\ a\ l\ i\ n\ t\ e\ r$
$a\ t\ i\ o\ n\ a\ l\ i\ n\ t\ e\ r\ n$
$t\ i\ o\ n\ a\ l\ i\ n\ t\ e\ r\ n\ a$
$i\ o\ n\ a\ l\ i\ n\ t\ e\ r\ n\ a\ t$
$o\ n\ a\ l\ i\ n\ t\ e\ r\ n\ a\ t\ i$
$n\ a\ l\ i\ n\ t\ e\ r\ n\ a\ t\ i\ o$
$a\ l\ i\ n\ t\ e\ r\ n\ a\ t\ i\ o\ n$
$l\ i\ n\ t\ e\ r\ n\ a\ t\ i\ o\ n\ a$

M

		$F \downarrow$		$L \downarrow$
	1	a	$l\ i\ n\ t\ e\ r\ n\ a\ t\ i\ o$	n
	2	a	$t\ i\ o\ n\ a\ l\ i\ n\ t\ e\ r$	n
	3	e	$r\ n\ a\ t\ i\ o\ n\ a\ l\ i\ n$	t
$I \rightarrow$	4	i	$n\ t\ e\ r\ n\ a\ t\ i\ o\ n\ a$	l
	5	i	$o\ n\ a\ l\ i\ n\ t\ e\ r\ n\ a$	t
	6	l	$i\ n\ t\ e\ r\ n\ a\ t\ i\ o\ n$	a
	7	n	$a\ l\ i\ n\ t\ e\ r\ n\ a\ t\ i$	o
	8	n	$a\ t\ i\ o\ n\ a\ l\ i\ n\ t\ e$	r
	9	n	$t\ e\ r\ n\ a\ t\ i\ o\ n\ a\ l$	i
	10	o	$n\ a\ l\ i\ n\ t\ e\ r\ n\ a\ t$	i
	11	r	$n\ a\ t\ i\ o\ n\ a\ l\ i\ n\ t$	e
	12	t	$e\ r\ n\ a\ t\ i\ o\ n\ a\ l\ i$	n
	13	t	$i\ o\ n\ a\ l\ i\ n\ t\ e\ r\ n$	a

$bwt(v) = L = nntltaoriiena$ and $I = 4$.

The rows of M are conjugate of v sorted in lexicographic order.

The Burrows Wheeler Transform

Given a word $w \in \Sigma^*$, the Burrows-Wheeler Transform of w, $bwt(w)$ is a permutation of the letters in w, obtained as concatenation of the last letters of the lexicographically sorted list of its conjugates.

$i\ n\ t\ e\ r\ n\ a\ t\ i\ o\ n\ a\ l$
$n\ t\ e\ r\ n\ a\ t\ i\ o\ n\ a\ l\ i$
$t\ e\ r\ n\ a\ t\ i\ o\ n\ a\ l\ i\ n$
$e\ r\ n\ a\ t\ i\ o\ n\ a\ l\ i\ n\ t$
$r\ n\ a\ t\ i\ o\ n\ a\ l\ i\ n\ t\ e$
$n\ a\ t\ i\ o\ n\ a\ l\ i\ n\ t\ e\ r$
$a\ t\ i\ o\ n\ a\ l\ i\ n\ t\ e\ r\ n$
$t\ i\ o\ n\ a\ l\ i\ n\ t\ e\ r\ n\ a$
$i\ o\ n\ a\ l\ i\ n\ t\ e\ r\ n\ a\ t$
$o\ n\ a\ l\ i\ n\ t\ e\ r\ n\ a\ t\ i$
$n\ a\ l\ i\ n\ t\ e\ r\ n\ a\ t\ i\ o$
$a\ l\ i\ n\ t\ e\ r\ n\ a\ t\ i\ o\ n$
$l\ i\ n\ t\ e\ r\ n\ a\ t\ i\ o\ n\ a$

M

		F ↓		L ↓
	1	a	$l\ i\ n\ t\ e\ r\ n\ a\ t\ i\ o$	n
	2	a	$t\ i\ o\ n\ a\ l\ i\ n\ t\ e\ r$	n
	3	e	$r\ n\ a\ t\ i\ o\ n\ a\ l\ i\ n$	t
$I \rightarrow$	4	i	$n\ t\ e\ r\ n\ a\ t\ i\ o\ n\ a$	l
	5	i	$o\ n\ a\ l\ i\ n\ t\ e\ r\ n\ a$	t
	6	l	$i\ n\ t\ e\ r\ n\ a\ t\ i\ o\ n$	a
	7	n	$a\ l\ i\ n\ t\ e\ r\ n\ a\ t\ i$	o
	8	n	$a\ t\ i\ o\ n\ a\ l\ i\ n\ t\ e$	r
	9	n	$t\ e\ r\ n\ a\ t\ i\ o\ n\ a\ l$	i
	10	o	$n\ a\ l\ i\ n\ t\ e\ r\ n\ a\ t$	i
	11	r	$n\ a\ t\ i\ o\ n\ a\ l\ i\ n\ t$	e
	12	t	$e\ r\ n\ a\ t\ i\ o\ n\ a\ l\ i$	n
	13	t	$i\ o\ n\ a\ l\ i\ n\ t\ e\ r\ n$	a

$bwt(v) = L = nntltaoriiena$ and $I = 4$.

The rows of M are conjugate of v sorted in lexicographic order.

The Extended Burrows-Wheeler Transform

Given two words $u, v \in \Sigma^*$ we define the following order relation:

$$u \preceq_\omega v \Longleftrightarrow u^\omega <_{lex} v^\omega$$

Example

This order is different from lexicographic one. For instance $ab <_{lex} aba$ but $aba \preceq_\omega ab$ since $aba^\omega = abaabaaba\cdots$ and $ab^\omega = abababab\cdots$.

The Extended Burrows-Wheeler Transform

Given two words $u, v \in \Sigma^*$ we define the following order relation:

$$u \preceq_\omega v \Longleftrightarrow u^\omega <_{lex} v^\omega$$

Example

This order is different from lexicographic one. For instance $ab <_{lex} aba$ but $aba \preceq_\omega ab$ since $aba^\omega = abaabaaba\cdots$ and $ab^\omega = abababab\cdots$.

Given a set of words , $W = \{w_1, w_2, \ldots w_n\}$ with $w_1, w_2, \ldots w_n \in \Sigma^*$ the Extended Burrows-Wheeler Transform of W, denoted $ebwt(W)$, is a transformation that produces a word as follows:

- sort according to the $\preceq_\omega$ order the conjugates of the words in W;
- take the concatenation of the last letters of the sorted list.

Example

Consider the set $W = \{abac, bca, cbab, cba\}$.

$a\ b\ a\ c\ a\ b \cdots$		1	$a\ b\ a\ c$
$a\ b\ c\ a\ b\ c \cdots$		2	$a\ b\ c$
$a\ b\ c\ b\ a\ b \cdots$		3	$a\ b\ c\ b$
$a\ c\ a\ b\ a\ c \cdots$		4	$a\ c\ a\ b$
$a\ c\ b\ a\ c\ b \cdots$		5	$a\ c\ b$
$b\ a\ b\ c\ b\ a \cdots$		6	$b\ a\ b\ c$
$b\ a\ c\ a\ b\ a \cdots$	$\Longrightarrow$	7	$b\ a\ c\ a$
$b\ a\ c\ b\ a\ c \cdots$		8	$b\ a\ c$
$b\ c\ a\ b\ c\ a \cdots$		9	$b\ c\ a$
$b\ c\ b\ a\ b\ c \cdots$		10	$b\ c\ b\ a$
$c\ a\ b\ a\ c\ a \cdots$		11	$c\ a\ b\ a$
$c\ a\ b\ c\ a\ b \cdots$		12	$c\ a\ b$
$c\ b\ a\ b\ c\ b \cdots$		13	$c\ b\ a\ b$
$c\ b\ a\ c\ b\ a \cdots$		14	$c\ b\ a$

Figure : The output of $E(S)$ is $L = ccbbbcacaaabba$

Lyndon Words

A *Lyndon word* is a primitive word which is the minimum in its conjugacy class, with respect to the lexicographic order relation.

Lyndon Words

A *Lyndon word* is a primitive word which is the minimum in its conjugacy class, with respect to the lexicographic order relation.

There exist linear algorithms that for any word $w \in \Sigma^*$ compute the Lyndon word of its conjugacy class. We call it the Lyndon word of w and we denote it by T_w.

Lyndon Words

A *Lyndon word* is a primitive word which is the minimum in its conjugacy class, with respect to the lexicographic order relation.

There exist linear algorithms that for any word $w \in \Sigma^*$ compute the Lyndon word of its conjugacy class. We call it the Lyndon word of w and we denote it by T_w.

Theorem (Chen, Fox, Lyndon 1958)

Every word $w \in \Sigma^+$ has a unique factorization $w = w_1 \cdots w_s$ such that $w_1 \geq_{lex} \cdots \geq_{lex} w_s$ is a non-increasing sequence of Lyndon words.

Lyndon Words

A *Lyndon word* is a primitive word which is the minimum in its conjugacy class, with respect to the lexicographic order relation.

There exist linear algorithms that for any word $w \in \Sigma^*$ compute the Lyndon word of its conjugacy class. We call it the Lyndon word of w and we denote it by T_w.

Theorem (Chen, Fox, Lyndon 1958)

Every word $w \in \Sigma^+$ has a unique factorization $w = w_1 \cdots w_s$ such that $w_1 \geq_{lex} \cdots \geq_{lex} w_s$ is a non-increasing sequence of Lyndon words.

The Lyndon factorization of a given word can be computed in linear time [Duval 1983].

One Word

Theorem (Giancarlo, Restivo, Sciortino 2007)

Let T be a Lyndon word. For any integers h, k with $1 \leq h, k \leq |T|$, the following statements are equivalent:

i. $conj_h(T) <_{lex} conj_k(T)$;

ii. $conj_h(T) \preceq_\omega conj_k(T)$;

iii. $suf_h(T) <_{lex} suf_k(T)$.

This means that for a Lyndon word lexicographic sorting of the suffixes, lexicographic sorting of the conjugates and $\preceq_\omega$ sorting of the conjugates are equivalent

Two words

Theorem

Let T_1 and T_2 be two Lyndon words, then $T_1 \leq_{lex} T_2$ if and only if $T_1 \preceq_{\omega} T_2$.

Two words

Theorem

Let T_1 and T_2 be two Lyndon words, then $T_1 \leq_{lex} T_2$ if and only if $T_1 \preceq_\omega T_2$.

Lemma

Let T_1 and T_2 be two distinct Lyndon words. If $T_1 <_{lex} T_2$ and $h \leq k$ then

$$conj_h(T_1) \prec_\omega conj_k(T_2) \Leftrightarrow suf_h(T_1) \leq_{lex} suf_k(T_2).$$

Two words

Theorem

Let T_1 and T_2 be two Lyndon words, then $T_1 \leq_{lex} T_2$ if and only if $T_1 \preceq_\omega T_2$.

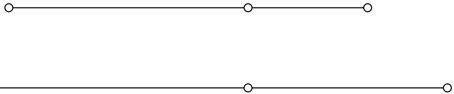
Lemma

Let T_1 and T_2 be two distinct Lyndon words. If $T_1 <_{lex} T_2$ and $h \leq k$ then

$$conj_h(T_1) \prec_\omega conj_k(T_2) \Leftrightarrow suf_h(T_1) \leq_{lex} suf_k(T_2).$$

$T_1 =$

$T_2 =$



Two words

Theorem

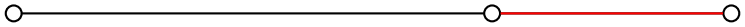
Let T_1 and T_2 be two Lyndon words, then $T_1 \leq_{lex} T_2$ if and only if $T_1 \preceq_\omega T_2$.

Lemma

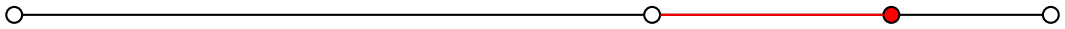
Let T_1 and T_2 be two distinct Lyndon words. If $T_1 <_{lex} T_2$ and $h \leq k$ then

$$conj_h(T_1) \prec_\omega conj_k(T_2) \Leftrightarrow suf_h(T_1) \leq_{lex} suf_k(T_2).$$

$T_1 =$



$T_2 =$



Lemma

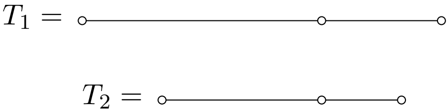
Let T_1 and T_2 be two distinct Lyndon words. If $T_1 <_{lex} T_2$ and $h > k$, then

$$conj_h(T_1) \prec_\omega conj_k(T_2) \Leftrightarrow suf_h(T_1) \leq_{lex} suf_k(T_2)pref_{h-k}(T_2^\omega).$$

Lemma

Let T_1 and T_2 be two distinct Lyndon words. If $T_1 <_{lex} T_2$ and $h > k$, then

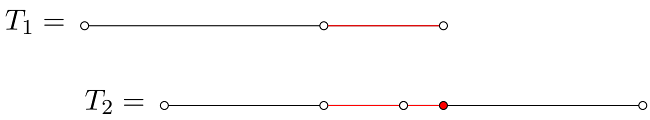
$$conj_h(T_1) \prec_\omega conj_k(T_2) \Leftrightarrow suf_h(T_1) \leq_{lex} suf_k(T_2)pref_{h-k}(T_2^\omega).$$



Lemma

Let T_1 and T_2 be two distinct Lyndon words. If $T_1 <_{lex} T_2$ and $h > k$, then

$$conj_h(T_1) \prec_\omega conj_k(T_2) \Leftrightarrow suf_h(T_1) \leq_{lex} suf_k(T_2)pref_{h-k}(T_2^\omega).$$



A more general formulation of previuos lemmas is the following:

Theorem

Let u and v be primitive words, let T_u and T_v be their corresponding Lyndon words. Let suppose $T_u <_{lex} T_v$ and let r be the integer such that $u = conj_r(T_u)$. Then

$$u \prec_\omega v \Leftrightarrow pref_r(u^\omega) \leq_{lex} pref_r(v^\omega).$$

An algorithm for sorting the conjugates of a multiset of Lyndon words

As an application of the previous results, given a set $\mathcal{T} = \{T_1, T_2, \ldots, T_m\}$ of lexicographically sorted multiset of Lyndon words, we provide an algorithm that produces the $\preceq_\omega$ sorted list of the conjugates of words in $\mathcal{T}$.

An algorithm for sorting the conjugates of a multiset of Lyndon words

As an application of the previous results, given a set $\mathcal{T} = \{T_1, T_2, \ldots, T_m\}$ of lexicographically sorted multiset of Lyndon words, we provide an algorithm that produces the $\preceq_\omega$ sorted list of the conjugates of words in $\mathcal{T}$.

Our strategy consists in inserting a new conjugate in an already sorted list of other conjugates, analyzing them from right to left and proceeding from the greatest Lyndon word down to the smallest one.

Actually our algorithm implicitely uses the above results, and do not use any comparison.
More details in the proceedings.

Sorting suffixes by Lyndon factorization

Let $w = a_1 \cdots a_n \in \Sigma^*$ and let $w_1 w_2 \cdots w_s$ be its Lyndon factorization. Let $j_1, j_2, \ldots, j_s$ be the positions in w of the last characters of factors $w_1, w_2, \ldots, w_s$, respectively. Obviously $j_s = n$.

Sorting suffixes by Lyndon factorization

Let $w = a_1 \cdots a_n \in \Sigma^*$ and let $w_1 w_2 \cdots w_s$ be its Lyndon factorization.
Let $j_1, j_2, \ldots, j_s$ be the positions in w of the last characters of factors $w_1, w_2, \ldots, w_s$, respectively. Obviously $j_s = n$.
For any position p in w, we define

$$L(p) = \min\{j_k \mid 1 \leq j_k \leq s \text{ and } p \leq j_k\}$$

and

$$l(p) = L(p) - p + 1.$$

Sorting suffixes by Lyndon factorization

Let $w = a_1 \cdots a_n \in \Sigma^*$ and let $w_1 w_2 \cdots w_s$ be its Lyndon factorization.
Let $j_1, j_2, \ldots, j_s$ be the positions in w of the last characters of factors $w_1, w_2, \ldots, w_s$, respectively. Obviously $j_s = n$.
For any position p in w, we define

$$L(p) = \min\{j_k \mid 1 \leq j_k \leq s \text{ and } p \leq j_k\}$$

and

$$l(p) = L(p) - p + 1.$$

Theorem

Let $w \in \Sigma^*$ and let p and q be two positions in w, $p < q$. Then

$$w[q,n] < w[p,n] \Leftrightarrow pref_{l(q)}(w[q,n]) \leq pref_{l(q)}(w[p,n]).$$

Applications

REMARK 1:
If p and q are two positions in w with $p < q$, $lcp(p, q)$ denotes the length of the longest common prefix between the suffixes $w[p, n]$ and $w[q, n]$.

Applications

REMARK 1:
If p and q are two positions in w with $p < q$, $lcp(p, q)$ denotes the length of the longest common prefix between the suffixes $w[p, n]$ and $w[q, n]$.

A consequence of the previous theorem is that in order to get the mutual order between $w[p, n]$ and $w[q, n]$ it is sufficient to execute $\min(lcp(p, q) + 1, l(q))$ symbol comparisons.

Applications

REMARK 1:
If p and q are two positions in w with $p < q$, $lcp(p, q)$ denotes the length of the longest common prefix between the suffixes $w[p, n]$ and $w[q, n]$.

A consequence of the previous theorem is that in order to get the mutual order between $w[p, n]$ and $w[q, n]$ it is sufficient to execute $\min(lcp(p, q) + 1, l(q))$ symbol comparisons.

This would allow the decreasing of the number of comparisons in algorithms for SA based on symbol comparisons.

The following example shows that $l(q)$ can be much smaller than $lcp(p, q) + 1$.

Example

Example

Let $w = abaaaabaaaaabaaaabaaaaaab$. The Lyndon factorization of w is

$$ab|aaaab|aaaaabaaaab|aaaaaab$$

Consider the suffixes $w[2,25] = baaaabaaaaabaaaabaaaaaab$ and $w[13,25] = baaaabaaaaaab$. We have $lcp(2,13) = 11$ and $l(13) = 6$.

$$\begin{array}{rccccc} & & l(13) & & lcp(2,13)+1 & \\ & & \downarrow & & \downarrow & \\ w[2,25] = & baaaa & b & aaaaa & b & aaaabaaaaaab \\ w[13,25] = & baaaa & b & aaaaa & a & b \end{array}$$

Applications

REMARK 2:

If p and q are in the same Lyndon factor, i.e. $L(p) = L(q)$, then the order of the two suffixes is the same as the order of the suffixes inside their common Lyndon factor. This property can be extended to blocks of consecutive Lyndon factors (Lyndon blocks).

Applications

REMARK 2:
If p and q are in the same Lyndon factor, i.e. $L(p) = L(q)$, then the order of the two suffixes is the same as the order of the suffixes inside their common Lyndon factor. This property can be extended to blocks of consecutive Lyndon factors (Lyndon blocks).

Application: new strategy for the construction of the suffix array of a word [MRRS, to be presented in PSC2013]:

1. find the Lyndon factorization of a word;
2. find separately the suffix array of every Lyndon factor;
3. merge the suffix arrays of pairs of adjacent Lyndon blocks into one.

Applications

REMARK 2:
If p and q are in the same Lyndon factor, i.e. $L(p) = L(q)$, then the order of the two suffixes is the same as the order of the suffixes inside their common Lyndon factor. This property can be extended to blocks of consecutive Lyndon factors (Lyndon blocks).

Application: new strategy for the construction of the suffix array of a word [MRRS, to be presented in PSC2013]:

1. find the Lyndon factorization of a word;
2. find separately the suffix array of every Lyndon factor;
3. merge the suffix arrays of pairs of adjacent Lyndon blocks into one.

This strategy allows:

- Parallelization;
- Online sorting.

Applications

The suffix permutation of a word $w = a_1 a_2 \cdots a_n$ is the permutation π_w over $\{1, \ldots, n\}$, where $\pi_w(i)$ is the *rank* of the suffix $w[i, n]$ in the set of the lexicographically sorted suffixes of w (i.e. the inverse of the permutation defined by the suffix array SA_w).

Given a permutation π over $\{1, \ldots, n\}$, an integer i $(1 \leq i \leq n)$ is a left-to-right minimum of π if $\pi(j) > \pi(i)$, for all $j < i$.

Applications

The suffix permutation of a word $w = a_1 a_2 \cdots a_n$ is the permutation π_w over $\{1, \ldots, n\}$, where $\pi_w(i)$ is the *rank* of the suffix $w[i, n]$ in the set of the lexicographically sorted suffixes of w (i.e. the inverse of the permutation defined by the suffix array SA_w).

Given a permutation π over $\{1, \ldots, n\}$, an integer i $(1 \leq i \leq n)$ is a left-to-right minimum of π if $\pi(j) > \pi(i)$, for all $j < i$.

Theorem (Hohlweg, Reutenauer 2003)

Let w be a word, let $i_1 = 1, i_2, \ldots, i_k$ be the start positions of the factors in its Lyndon factorization and let π_w be its suffix permutation. Then the values $i_1, i_2, \ldots, i_k$ correspond to the positions of the left to right minima of π_w.

This theorem can be deduced from the results in the present paper.

Example

The suffix array of the word $w = abaaaabaaaabaaaabaaaaaab$ is

$$[19, 20, 8, 21, 14, 3, 9, 22, 15, 4, 10, 23, 16, 5, 11, 24, 17, 6, 12, 1, 25, 18, 7, 13, 2].$$

The suffix permutation is given by:

$$[\textcolor{red}{20}, 25, \textcolor{red}{6}, 10, 14, 18, 23, \textcolor{red}{3}, 7, 11, 15, 19, 24, 5, 9, 13, 17, 22, \textcolor{red}{1}, 2, 4, 8, 12, 16, 21].$$

Since there exist linear algorithms for the construction of SA, this give an linear alternative to Duval's algorithm for finding Lyndon Factorization.

Thanks for your attention!