Organizačné poznámky

- Posledná šanca rozhodnúť sa pre projekt
- Do 15.4. si vyberte článok na prezentáciu
  - výber zapíšte v Moodli
  - vyberte zo zoznamu na stránke, alebo nejaký iný
  - prezentácie budú posledné dva týždne semestra
  - k článku sa vás môžem spýtať aj na skúške

Suffix array

Array of suffixes in lexicographic order

(assume \$ $< a \quad \forall a \in \Sigma$)

i	0	1	2	3	4	5	6
$S[i]$	b	a	n	a	n	a	\$

i	$SA[i]$	Suffix
0	6	\$
1	5	a\$
2	3	ana\$
3	1	anana\$
4	0	banana\$
5	4	na\$
6	2	nana\$

Can be computed in $O(n)$

Burrows-Wheeler transform (BWT) 1994

$S=$banana$

Sort all rotations of the word lexicographically:

b	a	n	a	n	a	$
a	n	a	n	a	$	b
n	a	n	a	$	b	a
a	n	a	$	b	a	n
n	a	$	b	a	n	a
a	$	b	a	n	a	n
$	b	a	n	a	n	a

$	b	a	n	a	n	**a**
a	$	b	a	n	a	**n**
a	n	a	$	b	a	**n**
a	n	a	n	a	$	**b**
b	a	n	a	n	a	**$**
n	a	$	b	a	n	**a**
n	a	n	a	$	b	**a**

BWT = annb$aa

Can be computed using suffix array: BWT[i] = S[SA[i]-1] (or S[n] if SA[i]=0)

Reverse transformation

Get first column (F) by sorting the last (L):

\$	...	a
a	...	n
a	...	n
a	...	b
b	...	\$
n	...	a
n	...	a

$F[i]$ follows $L[i]$ in S

Find \$ in L, get $S[0]$ in F ($S[0] = b$)

Find $S[0]$ in L, get $S[1]$ in F ($S[1] = a$)

Find $S[1]$ in L, get $S[2]$ in F ($S[2] \in \{n, \$\}$) ...

Reverse transformation

Find $S[i]$ in L, get $S[i+1]$ in F

What if multiple occurrences of $S[i]$ in L?

**$**$_6$	b	a	n	a	n	**a**$_5$
a$_5$	$	b	a	n	a	**n**$_4$
a$_3$	n	a	$	b	a	**n**$_2$
a$_1$	n	a	n	a	$	**b**$_0$
b$_0$	a	n	a	n	a	**$**$_6$
n$_4$	a	$	b	a	n	**a**$_3$
n$_2$	a	n	a	$	b	**a**$_1$

Relative ordering of a's (or n's) the same in L and F. Why?

Find $S[0]$ in L, get $S[1]$ in F ($S[1] =$ 3rd a)

Find 3rd a in L, get $S[2]$ in F ($S[2] =$ 2nd n)

Reverse transformation

0:	$\$_6$	a_5
1:	a_5	n_4
2:	a_3	n_2
3:	a_1	b_0
4:	b_0	$\$_6$
5:	n_4	a_3
6:	n_2	a_1

Sort L to get F, then build the following data structures:

Representation of F :

$: 0

a: 1

b: 4

n: 5

0: $
1: a
2: a
3: a
4: b
5: n
6: n

Representation of L:

$: 4

a: 0, 5, 6

b: 3

n: 1, 2

Use of Burrows-Wheeler transform in compression

$S =$ `ema.ma.mamu.mama.ma.emu.ema.sa.ma$`

$\mathrm{B}WT =$ `auaaaauaammsmmmmmm$....ae.e..ea.mm`

In a given region of SA common prefixes

Preceded by similar letters

In our case $'$`.`$'$preceded by u,a

Regions with the same letter repeated or few letters mixed

Use of Burrows-Wheeler transform in compression

$S =$`ema.ma.mamu.mama.ma.emu.ema.sa.ma$`

$BWT =$`auaaaauaammsmmmmmm$....ae.e..ea.mm`

Regions with the same letter repeated or few letters mixed

Move-to-front recording: replace $S[i]$ by the number of distinct letters since last occurrence of $S[i]$ in $S[0..i-1]$

`$.aemsu|auaaaauaammsmmmmmm$....ae.e..ea.mm`

4, 1, 1, 0, 0, 0, 1, 1, 0, 3, 0, 3, 1, 0, 0, 0, 0, 0, 6, 6, 0, 0, 0, 4, 6, 2, 1, 1, 0, 1, 2, 2, 4, 0

Small numbers, many zeroes, in English text 50% zeroes

Use of Burrows-Wheeler transform in compression

Encode MTF of BWT by Huffman or arithmetic encoding

$S \longrightarrow BWT \longrightarrow MTF \longrightarrow$ Huffman/arithmetic encoding $\longrightarrow$ compressed S

e.g. bzip

Entropy

Let $\Sigma = \{x_1, \ldots, x_\sigma\}$, probability of x_i is p_i

Entropy of this distribution is: $-\sum_{i=1}^{\sigma} p_i \log_2 p_i$

Entropy gives lower bound on the number of bits needed per character.

Huffman encoding assigns shorter code sequences to more frequent words, uses approximately $-\log_2 p_i$ bits to encode x_i

Arithmetic encoding has fewer rounding errors

Example: Entropy of S is 2.37, entropy of MTF of BWT is 2.18