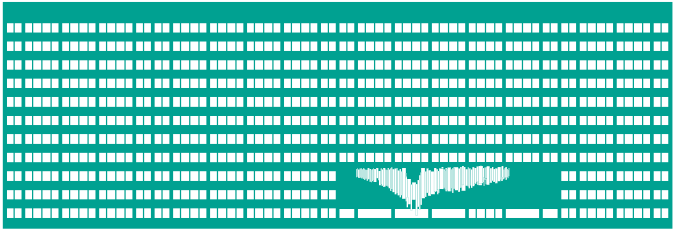
VŠB TECHNICKÁ UNIVERZITA OSTRAVA

VSB TECHNICAL UNIVERSITY OF OSTRAVA



www.vsb.cz

Transformations for data compression

LZ algorithms

Michal Vasinek

VSB – Technical University of Ostrava

name.surname@vsb.cz

May 23, 2019

Overview

- Burrows-Wheeler Transformation
- Move-To-Front
- Run-Lengh Encoding
- FM-Index

Transformation

- We use transformations to modify input sequence so that it is more suitable for compression (or easily compressed).
- Transformation is usually followed by statistical coding methods.
- If the transformation is reversible then we use it in lossless compression (BWT, MTF, RLE).
- Non-reversible transformations used in multimedia compression (JPEG, MPEG).

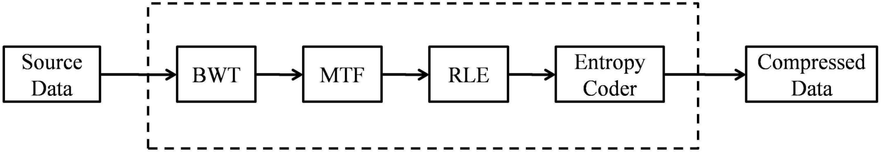


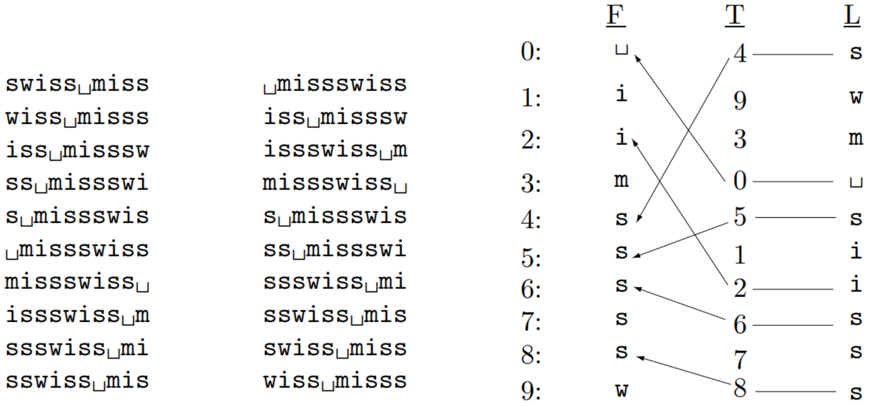
Figure: BWT based compression schema - from MDPI

Burrows-Wheeler Transformation - BWT

- Developed in 1994 by Michael Burrows and David Wheeler.
- The main idea of the BWT method is to start with a string S of n symbols and to permute them into another string L that satisfies two conditions:
    1. Any region of L will tend to have a concentration of just a few symbols.
    2. It is possible to reconstruct the original string S from L.

Burrows-Wheeler Transformation - Example

- BWT is based on sorting of input sequence rotations.



swiss␣miss	␣missswiss
wiss␣misss	iss␣missw
iss␣missw	issswiss␣m
ss␣missswi	missswiss␣
s␣missswis	s␣missswis
␣missswiss	ss␣missswi
missswiss␣	ssswiss␣mi
issswiss␣m	sswiss␣mis
ssswiss␣mi	swiss␣miss
sswiss␣mis	wiss␣misss

	F	T	L
0:	␣	4	s
1:	i	9	w
2:	i	3	m
3:	m	0	␣
4:	s	5	s
5:	s	1	i
6:	s	2	i
7:	s	6	s
8:	s	7	s
9:	w	8	s

Figure: Handbook of Data Compression, page 1090.

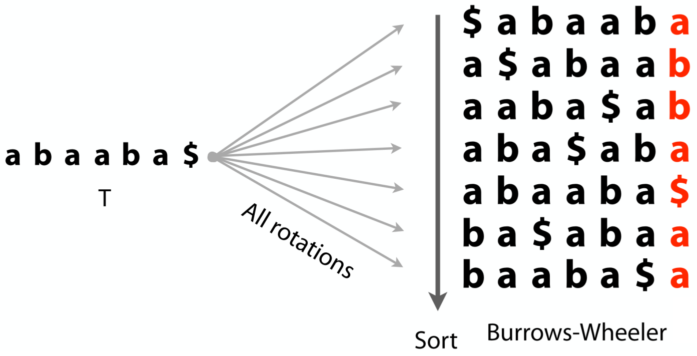
Burrows-Wheeler Transformation - Why it works?

- Consider that somewhere in the text there are these words: *bail*, *fail*, *hail*, *jail*, *mail*, *nail*, *pail*, *rail*, *sail*, *tail* and *wail*.
- Once the rotations are sorted there will be rotations beggining with **il**

F				L
…	…	…	…	…
i	l	…	b	a
i	l	…	f	a
i	l	…	h	a
i	l	…	j	a
i	l	…	m	a
i	l	…	n	a
…	…	…	…	…

BWT - Last To First Mapping

Consider another example:



Index all symbols by their order of occurence.

$a_0\ b_0\ a_1\ a_2\ b_1\ a_3\ \$$

BWT - Last To First Mapping

LF mapping

The order of occurences of a symbol in F and L are equal.

F						L
\$	a_0	b_0	a_1	a_2	b_1	a_3
a_3	\$	a_0	b_0	a_1	a_2	b_1
a_1	a_2	b_1	a_3	\$	a_0	b_0
a_2	b_1	a_3	\$	a_0	b_0	a_1
a_0	b_0	a_1	a_2	b_1	a_3	\$
b_1	a_3	\$	a_0	b_0	a_1	a_2
b_0	a_1	a_2	b_1	a_3	\$	a_0

BWT - Last To First Mapping

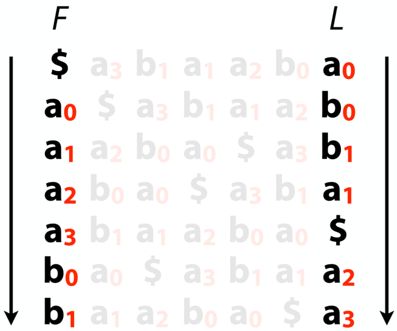
LF mapping

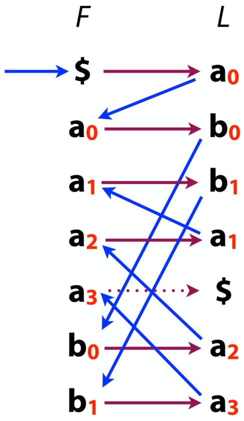
The order of occurences of a symbol in F and L are equal.

$$
\begin{array}{ccccccc}
\$ & a_0 & b_0 & a_1 & a_2 & b_1 & a_3 \\
a_3 & \$ & a_0 & b_0 & a_1 & a_2 & \mathbf{b_1} \\
a_1 & a_2 & b_1 & a_3 & \$ & a_0 & \mathbf{b_0} \\
a_2 & b_1 & a_3 & \$ & a_0 & b_0 & a_1 \\
a_0 & b_0 & a_1 & a_2 & b_1 & a_3 & \$ \\
\mathbf{b_1} & a_3 & \$ & a_0 & b_0 & a_1 & a_2 \\
\mathbf{b_0} & a_1 & a_2 & b_1 & a_3 & \$ & a_0
\end{array}
$$

BWT - reversing

- \$ denotes the special symbol for the end of sequence.
- The symbol in F column is preceeded by the symbol in L column in original message.





Decoding - process outline

1. Construct the F column by sorting the L column.
2. Construct a mapping between symbols in L and F, i.e. compute the index T of each symbol in L to F.
3. Start with the first line and say it is the current row. It contains \$ in F.
4. For the current row the symbol in L column is the symbol in the original sequence. Output the symbol and locate it in F (it becomes the current line) repeat this step until the message is reconstructed.

The message decoded in this way is in reversed order since we decoded the original sequence from the last symbol.

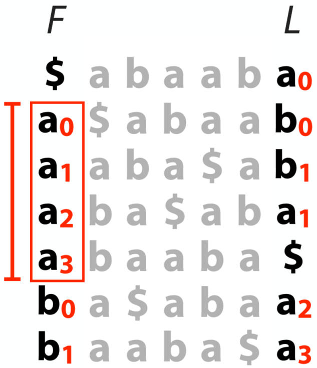
BWT - implementation notes

- Construction of all rotations $O(n^2)$ time and space operation.
- Instead sufficient to keep indexes of rotations with respect to the original string: suffix array.
- Sorting of strings is generally $O(n \log n)$ operation, SA-IS algorithm can do it in $O(n)$.
- Usually applied on fixed-length block of text.

BWT - solution to subsequence exists problem

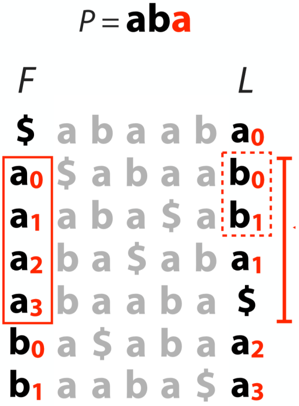
- Start matching from the last symbol of the searched pattern P.
- We know that the last symbol is located in the region that belongs to a symbol in F.

$P = \mathbf{aba}$



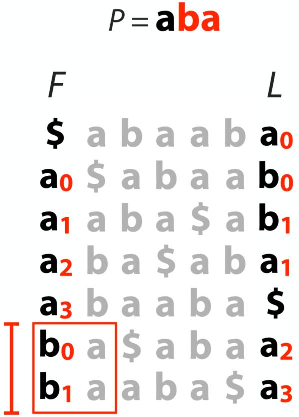
BWT - solution to subsequence exists problem

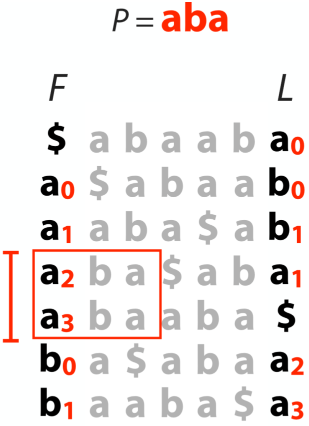
- Use LF mapping to find the indexes of b in L within the range of a.



BWT - solution to subsequence exists problem

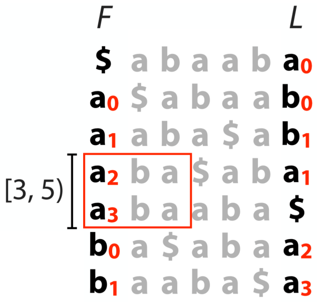
- Locate the top and bottom b in the F column => again it defines the range in which we will search.
- We immediatelly see that both these bs' were preceeded by a, so we located pattern $P = aba$.

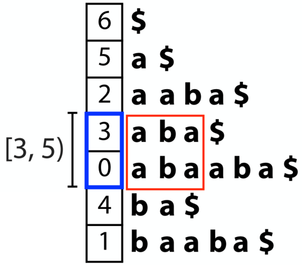




FM-Index - exists, count and locate

- The range of solutions to **exists** operation gives us also the number of occurences - **count** operation - of the pattern.
- Together with suffix array we can prepare **locate** operation.





FM-Index remarks

- Using additional structures we can implement exists, count and locate operations that works with $O(n)$ time complexity, where n is the length of the pattern.
- FM-Index is the data structured that allowed almost real time processing of the Next-Generation Sequencing of DNA.

bzip2

- bzip2 - standard data compression algorithm, characteristical with extremely good compression ratio.
- Option in 7zip software.
- Consists of several consecutive transformations eventually compressed by Huffman or Arithmetic coding.

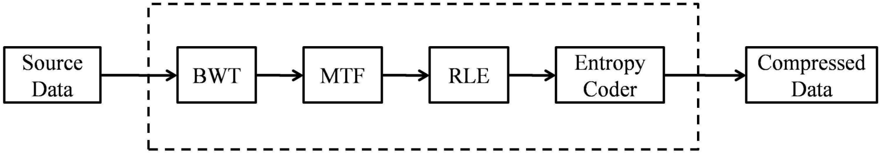


Figure: bzip2 compression schema - from MDPI

Move-To-Front coding

- First published by Ryabko in 1980 but authorship officialy belongs to Bentley in paper from 1986.
- The basic idea of this method is to maintain the alphabet Σ of symbols as a list where frequently-occurring symbols are located near the front.
- A symbol s is encoded as the number of symbols that precede it in this list.

Move-To-Front coding - Example

Encoding $s = aabcdeed$

Current alphabet	Current Symbol	Output
abcde	a	0
abcde	a	0
abcde	b	1
bacde	c	2
cbade	d	3
dcbae	e	4
edcba	e	0
edcba	d	1

- Remember - BWT groups symbols so they occur nearby or even repeat in L column.
- MTF encodes nearby symbols by small numbers.
- MTF encodes symbol repetition as a sequence of zeros.

Move-To-Front coding - Decoding

Decoding $c = 00123401$ if the alphabet is $\Sigma = \{a, b, c, d, e\}$.

Current alphabet	Current Symbol	Output
abcde	0	a
abcde	0	a
abcde	1	b
bacde	2	c
cbade	3	d
dcbae	4	e
edcba	0	e
edcba	1	d

- Each time we decode a symbol we move it to the front of the alphabet list.

Run-length encoding

- If a data item d occurs n consecutive times in the input stream, replace the n occurrences with the single pair (n, d).
- The n consecutive occurrences of a data item are called a run length of n, and this approach to data compression is called run-length encoding or RLE.
- Example: RLE(aabcdddaeeee) = 2a1b1c3d1a4e
- Not very usefull for text compression. Each additional byte used to represent run-length would effectively doubled the size of the input.

Run-length encoding - modification 1

- Use special character such as @ to denote repetition.
- Example: RLE(aabcdddaeeee) = @2abc@3da@4e
- Does it make sense to encode symbol repeating twice or three times? No, apply the rule only if more than three repetitions are present.
- Example: RLE(aabcdddaeee) = aabcddda@4e -> saved one byte

Run-length encoding - modification 2

- We usually view symbols and bytes as the same quantity.
- What if all symbols of the alphabet (i.e. all byte values) are used, so that we cannot use special symbol?
- Say we have n consecutive occurences of some symbol x, we send first k repetitions of x uncompressed and the rest of repetitions encoded by RLE.
- If decoder decodes k repetitions then it will know that RLE compressed pair will follow.
- RLE(aaaaaab) = aaa3ab

Thank you for your attention

Michal Vasinek

VSB – Technical University of Ostrava

name.surname@vsb.cz

May 23, 2019

VSB TECHNICAL UNIVERSITY OF OSTRAVA