Two Combinatorial Criteria for BWT Images

Konstantin M. Likhomanov   Arseny M. Shur

Ural State University, Yekaterinburg, Russia

Burrows-Wheeler Transform (BWT)

Introduced by M. Burrows, D. J. Wheeler, 1994

Burrows-Wheeler Transform (BWT)

Introduced by M. Burrows, D. J. Wheeler, 1994

An illustration:

B A N A N A

Burrows-Wheeler Transform (BWT)

Introduced by M. Burrows, D. J. Wheeler, 1994

An illustration:

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

Burrows-Wheeler Transform (BWT)

Introduced by M. Burrows, D. J. Wheeler, 1994

An illustration:

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

$$\mathrm{bwt}\,(\mathrm{BANANA}) = \mathrm{NNBAAA}$$

BWT always performs a permutation of the original word.

Burrows-Wheeler Transform (BWT)

Introduced by M. Burrows, D. J. Wheeler, 1994

An illustration:

$$\begin{array}{cccccc} A & B & A & N & A & \color{red}{N} \\ A & N & A & B & A & \color{red}{N} \\ A & N & A & N & A & \color{red}{B} \\ B & A & N & A & N & \color{red}{A} \\ N & A & B & A & N & \color{red}{A} \\ N & A & N & A & B & \color{red}{A} \end{array}$$

$$\mathrm{bwt}\,(\mathrm{BANANA}) = \mathrm{NNBAAA}$$

BWT always performs a permutation of the original word.
It is not a bijection because it doesn't distinguish between cyclic shifts.

BWT and context dependences

```
inimal length ......m
inimal letter ......m
inimal letter ......m
inimal letter ......m
inimal letter ......m
inimum letters......m
```

BWT and context dependences

$$\text{Right contexts} \left\{ \begin{array}{l} \texttt{inimal length ......m} \\ \texttt{inimal letter ......m} \\ \texttt{inimal letter ......m} \\ \texttt{inimal letter ......m} \\ \texttt{inimal letter ......m} \\ \texttt{inimum letters......m} \end{array} \right\} \text{of these letters}$$

BWT and context dependences

$$\text{Right contexts} \left\{ \begin{array}{l} \texttt{inimal length ......m} \\ \texttt{inimal letter ......m} \\ \texttt{inimal letter ......m} \\ \texttt{inimal letter ......m} \\ \texttt{inimal letter ......m} \\ \texttt{inimum letters......m} \end{array} \right\} \text{of these letters}$$

Usage of BWT:

	reversibility
+	clustering effect
+	linear-time algorithms for both bwt and bwt^{-1}
=	good preprocessing for lossless data compressors (bzip2, 7-Zip,. . .)

Combinatorial properties of BWT

- Earlier results: words with "simple" BWT images
  - Description of the words having BWT images of the form $b^i a^j$ (S. Mantaci, A. Restivo, S. Sciortino, 2003)
  - Description of the words having BWT images of the form $c^i b^j a^k$ (J. Simpson, S. J. Puglisi, 2008)
  - Partial description of the words having BWT images of the form $a_n^{i_n} a_{n-1}^{i_{n-1}} \dots a_1^{i_1}$ (A. Restivo, G. Rosone, 2009)

Combinatorial properties of BWT

- Earlier results: words with "simple" BWT images
  - Description of the words having BWT images of the form $b^i a^j$ (S. Mantaci, A. Restivo, S. Sciortino, 2003)
  - Description of the words having BWT images of the form $c^i b^j a^k$ (J. Simpson, S. J. Puglisi, 2008)
  - Partial description of the words having BWT images of the form $a_n^{i_n} a_{n-1}^{i_{n-1}} \ldots a_1^{i_1}$ (A. Restivo, G. Rosone, 2009)
- Two general decision problems (over any fixed ordered alphabet):
  - Given a word w, is w a BWT image?
  - Given a word $w = c_1 c_2 \ldots c_l$, is there a BWT image of the form $c_1^{p_1} \ldots c_l^{p_l}$ for some positive $p_1, \ldots, p_l$?

Combinatorial properties of BWT

- Earlier results: words with "simple" BWT images
  - Description of the words having BWT images of the form $b^i a^j$ (S. Mantaci, A. Restivo, S. Sciortino, 2003)
  - Description of the words having BWT images of the form $c^i b^j a^k$ (J. Simpson, S. J. Puglisi, 2008)
  - Partial description of the words having BWT images of the form $a_n^{i_n} a_{n-1}^{i_{n-1}} \dots a_1^{i_1}$ (A. Restivo, G. Rosone, 2009)
- Two general decision problems (over any fixed ordered alphabet):
  - Given a word w, is w a BWT image?
  - Given a word $w = c_1 c_2 \dots c_l$, is there a BWT image of the form $c_1^{p_1} \dots c_l^{p_l}$ for some positive $p_1, \dots, p_l$?

We present efficient solutions for both problems

Checking whether a word is a BWT image

The stable sorting of a word $w = b_1 \dots b_k$ is the permutation σ of the set $\{1, \dots, k\}$ such that the string $\sigma(w) = b_{\sigma(1)} \dots b_{\sigma(k)}$ is lexicographically sorted and the relative order of equal letters in $\sigma(w)$ is the same as in w.

Checking whether a word is a BWT image

The stable sorting of a word $w = b_1 \dots b_k$ is the permutation σ of the set $\{1, \dots, k\}$ such that the string $\sigma(w) = b_{\sigma(1)} \dots b_{\sigma(k)}$ is lexicographically sorted and the relative order of equal letters in $\sigma(w)$ is the same as in w.

Theorem 1

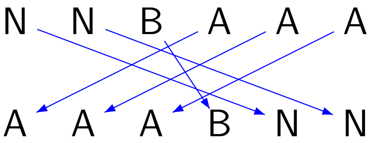
A word $w = c_1^{p_1} \dots c_k^{p_k}$, where $p_i > 0$ and $c_i \neq c_{i+1}$, is a BWT image if and only if the number of orbits of its stable sorting is equal to $\gcd(p_1, \dots, p_k)$.

Checking whether a word is a BWT image

The stable sorting of a word $w = b_1 \dots b_k$ is the permutation σ of the set $\{1, \dots, k\}$ such that the string $\sigma(w) = b_{\sigma(1)} \dots b_{\sigma(k)}$ is lexicographically sorted and the relative order of equal letters in $\sigma(w)$ is the same as in w.

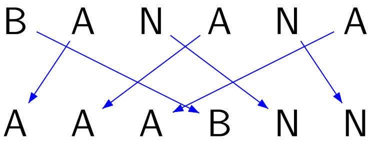
Theorem 1

A word $w = c_1^{p_1} \dots c_k^{p_k}$, where $p_i > 0$ and $c_i \neq c_{i+1}$, is a BWT image if and only if the number of orbits of its stable sorting is equal to $\gcd(p_1, \dots, p_k)$.



$(1\,4\,3\,6\,2\,5)$

✓



$(1\,2\,4)\,(3\,5\,6)$

✗

Checking whether a word is a BWT image

The stable sorting of a word $w = b_1 \dots b_k$ is the permutation σ of the set $\{1, \dots, k\}$ such that the string $\sigma(w) = b_{\sigma(1)} \dots b_{\sigma(k)}$ is lexicographically sorted and the relative order of equal letters in $\sigma(w)$ is the same as in w.

Theorem 1

A word $w = c_1^{p_1} \dots c_k^{p_k}$, where $p_i > 0$ and $c_i \neq c_{i+1}$, is a BWT image if and only if the number of orbits of its stable sorting is equal to $\gcd(p_1, \dots, p_k)$.

- A linear-time algorithm checks whether w is a BWT image, and returns "for free" any given preimage of w;
- so, this algorithm can be used in BWT-based compression schemes to check the correctness of decompression when calculating the inverse BWT

"Pumping" words to BWT images

A word w has a global ascent if $w = uv$ and each letter of u is less than or equal to each letter of v.

“Pumping” words to BWT images

A word w has a global ascent if $w = uv$ and each letter of u is less than or equal to each letter of v.

Theorem 2

Let $w = c_1 \ldots c_l$. A BWT image of the form $c_1^{p_1} \ldots c_l^{p_l}$, $p_i > 0$, exists if and only if w has no global ascents.

"Pumping" words to BWT images

A word w has a global ascent if $w = uv$ and each letter of u is less than or equal to each letter of v.

Theorem 2

Let $w = c_1 \ldots c_l$. A BWT image of the form $c_1^{p_1} \ldots c_l^{p_l}$, $p_i > 0$, exists if and only if w has no global ascents.

BANANA has no global ascents
One can check that BANNANA = bwt(AANANNB)

"Pumping" words to BWT images

A word w has a global ascent if $w = uv$ and each letter of u is less than or equal to each letter of v.

Theorem 2

Let $w = c_1 \dots c_l$. A BWT image of the form $c_1^{p_1} \dots c_l^{p_l}$, $p_i > 0$, exists if and only if w has no global ascents.

BANANA has no global ascents
One can check that BANNANA $=$ bwt(AANANNB)

In general, constructing such an image is not easy. We present a quadratic-time algorithm.

Algorithm for constructing a BWT image

Lemma

If a nonempty word has no global ascents, then we can delete one of its letters so that the new word will have no global ascents.

Algorithm for constructing a BWT image

Lemma

If a nonempty word has no global ascents, then we can delete one of its letters so that the new word will have no global ascents.

Stable sorting moves blocks of letters as a whole.

Algorithm for constructing a BWT image

Lemma

If a nonempty word has no global ascents, then we can delete one of its letters so that the new word will have no global ascents.

Stable sorting moves blocks of letters as a whole.

Lemma

If we change the length of some block of letters by its shift under a stable sorting, this doesn't change neither the number of orbits of the sorting nor the GCD of the lengths of blocks.

Algorithm for constructing a BWT image

Lemma

If a nonempty word has no global ascents, then we can delete one of its letters so that the new word will have no global ascents.

Stable sorting moves blocks of letters as a whole.

Lemma

If we change the length of some block of letters by its shift under a stable sorting, this doesn't change neither the number of orbits of the sorting nor the GCD of the lengths of blocks.

The idea of the algorithm: remove letters until we obtain a BWT image, then return them as zero-length blocks and resize those blocks by their shifts

An example

Let us construct a BWT image of the form $\mathrm{D}^{?}\mathrm{A}^{?}\mathrm{C}^{?}\mathrm{B}^{?}\mathrm{A}^{?}$

An example

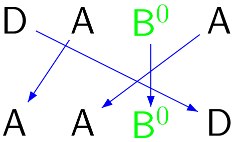
Let us construct a BWT image of the form $\mathrm{D}^?\mathrm{A}^?\mathrm{C}^?\mathrm{B}^?\mathrm{A}^?$

- Deleting letters, we get $\mathrm{DACBA} \to \mathrm{DABA} \to \mathrm{DAA}$ (a BWT image)

An example

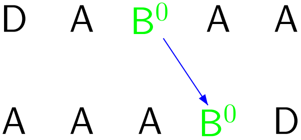
Let us construct a BWT image of the form $\mathrm{D}^?\mathrm{A}^?\mathrm{C}^?\mathrm{B}^?\mathrm{A}^?$

- Deleting letters, we get $\mathrm{DACBA} \to \mathrm{DABA} \to \mathrm{DAA}$ (a BWT image)



The block B^0 has zero shift and cannot be pumped

Auxiliary step: pump an existing block (the last A, having the shift 1)



Now the block B^0 has shift 1 and can be pumped to get DABAA

An example

Let us construct a BWT image of the form $\mathrm{D}^?\mathrm{A}^?\mathrm{C}^?\mathrm{B}^?\mathrm{A}^?$

- Deleting letters, we get $\mathrm{DACBA} \to \mathrm{DABA} \to \mathrm{DAA}$ (a BWT image)

$$
\begin{array}{cccccc}
\mathsf{D} & \mathsf{A} & \mathsf{C}^0 & \mathsf{B} & \mathsf{A} & \mathsf{A} \\
\\
\mathsf{A} & \mathsf{A} & \mathsf{A} & \mathsf{B} & \mathsf{C}^0 & \mathsf{D}
\end{array}
$$

The block C^0 has shift 2 and can be pumped to get $\mathrm{DACCBAA}$

One can check that $\mathrm{DACCBAA} = \mathsf{bwt}(\mathrm{AACBCAD})$.

Open problems

- Improve the complexity of the above algorithm
- Construct the shortest possible BWT image with given order of letters