Bijective BWT based Compression Schemes

Golnaz Badkobeh[1][0000−0001−5550−7149], Hideo Bannai[2][0000−0002−6856−5185], and Dominik Köppl[3][0000−0002−8721−4444]

[1] City, University of London, London, UK,
Golnaz.Badkobeh@city.ac.uk
[2] Tokyo Medical and Dental University, Tokyo, Japan,
hdbn.dsc@tmd.ac.jp
[3] University of Yamanashi, Kofu, Japan,
dkppl@yamanashi.ac.jp

Abstract. We investigate properties of the bijective Burrows-Wheeler transform (BBWT). We show that for any string w, a bidirectional macro scheme of size $O(r_B)$ can be induced from the BBWT of w, where r_B is the number of maximal character runs in the BBWT. We also show that $r_B = O(z \log^2 n)$, where n is the length of w and z is the number of Lempel-Ziv 77 factors of w. Then, we show a separation between BBWT and BWT by a family of strings with $r_B = \Omega(\log n)$ but having only $r = 2$ maximal character runs in the standard Burrows–Wheeler transform (BWT). However, we observe that the smallest r_B among all cyclic rotations of w is always at most r. While an $o(n^2)$ algorithm for computing an optimal rotation giving the smallest r_B is still open, we show how to compute the Lyndon factorizations – a component for computing BBWT – of all cyclic rotations in $O(n)$ time. Furthermore, we conjecture that we can transform two strings having the same Parikh vector to each other by BBWT and rotation operations, and prove this conjecture for the case of binary alphabets and permutations.

Keywords: Repetitiveness measure · Burrows–Wheeler Transform

1 Introduction

The Burrows–Wheeler transform (BWT) [11] has seen numerous applications in data compression and text indexing, and is used heavily by various tools in the field of bioinformatics. For any string w, $\mathsf{BWT}(w)$ is defined as the string obtained by concatenating the last characters of all cyclic rotations of w, in the lexicographic order of the cyclic rotations. BWT is not injective, as two strings are transformed to the same string if they are cyclic rotations of each other. Also, BWT is not surjective since BWT preserves the string length and by the pigeonhole principle, there exist strings that are not in the image of BWT.

For a string x, the inverse BWT transform is induced from the LF-mapping function $\psi_x(i)$ which maps position i in x to its rank among all positions ordered by $(x[i], i)$, i.e. $\psi_x(i) = |\{j \in [1, |x|] \mid x[j] < x[i]| + |\{j \in [1, i] \mid x[j] = x[i]\}|$. For a primitive string w, the standard BWT always constructs a string $x =$

$\mathsf{BWT}(w)$ such that ψ_x forms a single cycle (i.e., $\forall i, \exists j$ s.t. $\psi_x^j(i) = 1$), and $x[\psi_x^{|w|-1}(i)] \cdots x[\psi_x^0(i)]$ is a cyclic rotation of w. As an example, for $w[1..6] = \texttt{banana}$ and $x = \mathsf{BWT}(w) = \texttt{nnbaaa}$, $\psi_x(1) = 5, \psi_x(2) = 6, \psi_x(3) = 4, \psi_x(4) = 1, \psi_x(5) = 2, \psi_x(6) = 3$, and thus $\psi_x^6(i) = i$ for all $i \in [1, 6]$.

In general, ψ_x can form several cycles (e.g., when x is not in the image of BWT), and it is more natural to view the inverse BWT transform as a mapping from a string to a multiset of primitive cyclic strings. The bijective BWT (BBWT) [15,13,17] exploits this to define a bijection on strings. By selecting the lexicographically smallest rotation of each cyclic string and concatenating them in non-increasing lexicographic order, this mapping becomes a bijection that maps a string to another string. The forward transform $\mathsf{BBWT}(w)$ can then be defined as a transform that first computes the Lyndon factorization [12] of w, and then taking the last symbol of all cyclic rotations of all the Lyndon factors, sorted in ω-order $\prec_\omega$, which is an order defined, when x, y are primitive, as $x \prec_\omega y \iff x^\infty \prec y^\infty$, and $\prec$ denotes the standard lexicographic order.

It is known that the BBWT can be computed in linear time [6,8]. It can also be used as an index similar to the BWT [5,6], or as an index for a set of circular strings [10]. While the size r of the run-length compressed BWT (RLBWT) has been a focus of study in various contexts and is known to be small for highly-repetitive texts [20], the size r_B of the run-length compressed BBWT (RLBBWT) has not yet been studied rigorously. Biagi et al. [7] study the sensitivity [1] of r_B with respect to the *reverse* operation, and present an infinite family of strings such that r_B of a string and its reverse can differ by a factor of $\Omega(\log n)$.

In this paper, we investigate properties of BBWT and r_B. In detail, we show that we can induce a bidirectional macro scheme (BMS) [23] of size $O(r_B)$ for w, from the RLBBWT of w (Lemma 1). We further show that $r_B = O(z \log^2 n)$ where z is the number of Lempel-Ziv 77 (LZ77) factors of w (Theorem 1). Then, we show a separation between r_B and r, by a family of strings with $r_B = \Omega(\log n)$ but $r = 2$ (Theorem 2).

Noticing that $r_B = r$ for Lyndon words, the smallest r_B among all cyclic rotations of w is always at most r. While we do not yet know how to compute, in subquadratic time, such an optimal rotation that gives the smallest r_B, we show that we can compute the Lyndon factorizations of all rotations of w in linear time (Theorem 3).

Finally, we conjecture that two strings having the same Parikh vector can be transformed to each other by BBWT and rotation operations (Conjecture 1); we prove this conjecture for special cases (Theorem 4).

2 Preliminaries

Let Σ be a set of symbols referred to a the *alphabet*, and Σ^* the set of strings over Σ. For a string $x \in \Sigma^*$, $|x|$ denotes x's length. The empty string (the string of length 0) is denoted by ε. For integer $i \in [1, |x|]$, $x[i]$ is the ith symbol of x, and for integer $j \in [i, |x|]$, $x[i..j] = x[i] \cdots x[j]$. For convenience, let $x[i..j] = \varepsilon$ if

$i > j$. Let $x = x^1$, and for integer $k \geq 2$, $x^k = xx^{k-1}$ A string is *primitive*, if it cannot be represented as x^k for some string x and integer $k \geq 2$.

Let $rot(x) = x[|x|]x[1..|x|-1]$. A string y is a *cyclic rotation* (or simply a *rotation*) of x if there exists i such that $y = rot^i(x)$, where $rot^1(x) = rot(x)$, and for integer $k \geq 2$, $rot^k(x) = rot^{k-1}(rot(x))$.

Given a total order $\prec$ on Σ, the lexicographic order (also denoted by $\prec$) induced by $\prec$ is a total order on Σ^* such that $x \prec y$ if and only if x is a prefix of y, or, $x[i] \prec y[i]$ where $i = \min\{k \geq 1 \mid x[k] \neq y[k]\}$. A string w is a *Lyndon word*, if it is lexicographically smaller than all of its proper suffixes [16]. Lyndon words must therefore be primitive. Also, any string w can be partitioned into a unique sequence of lexicographically non-increasing Lyndon words, called the *Lyndon factorization* [12] of w, i.e., $w = f_1^{k_1} \cdots f_{\ell(w)}^{k_{\ell(w)}}$ where each f_i $(1 \leq i \leq \ell(w))$ is a Lyndon word, and $f_i \succ f_{i+1}$ for all $1 \leq i < \ell(w)$. We call $f_i^{k_i}$ the i-th *Lyndon necklace* of w. The ω-order $\prec_\omega$ is a total order over primitive strings, defined as: $x \prec_\omega y$ if and only if $x^\infty \prec y^\infty$. [4]

Given a string w, the Burrows–Wheeler transform $\mathsf{BWT}(w)$ is a string obtained by concatenating the last symbol of all cyclic rotations of w, in lexicographic order. The bijective BWT $\mathsf{BBWT}(w)$ is a string obtained by concatenating the last symbol of all cyclic rotations of all Lyndon factors in the Lyndon factorization of w, in ω-order. The number of maximal same-character runs in $\mathsf{BWT}(w)$ and $\mathsf{BBWT}(w)$ will be denoted by $r(w)$, and $r_B(w)$ respectively. Although $r(w)$, $r_B(w)$ and $\ell(w)$ are functions on strings to non-negative integers, we will omit writing the considered string and just write r, r_B, ℓ, if the context is clear.

3 Properties of r_B

We here analyze r_B as a repetitiveness measure for a string w. We first confirm that r_B corresponds to the size of a *bidirectional macro scheme (BMS)*, and is a repetitiveness measure for a form of dictionary compression, as is RLBWT. A BMS [23], the most expressive form of dictionary compression, partitions w into phrases, such that each phrase of length at least 2 can be represented as a reference to another substring of w. The referencing of the phrases induces a referencing forest over the positions: any position in a phrase of length at least 2 references another position in w, such that all positions in the same phrase have the same offset and thus adjacent positions point to adjacent positions, and there are no cycles.

Lemma 1. *There exists a BMS of size $O(r_B(w))$ that represents the string w.*

Proof. We follow the existence proof for a BMS of size $O(r(w))$ by Navarro et al. [21]. We consider a BMS such that each text position that does not correspond

[4] Mantaci et al. [17] define the ω-order as a total order over arbitrary strings (including non-primitive strings), but as it is not relevant in our presentation, we omit this for simplicity.

to a beginning of a same-character run in $\mathsf{BBWT}(w)$ will reference the text position corresponding to the preceding character in the $\mathsf{BBWT}(w)$. It is clear that there are no cycles in such a referencing, and we claim that this allows the string to be partitioned into $O(r_B)$ phrases, such that references of adjacent positions in a given phrase of length at least two point to adjacent positions.

Focus on the i-th Lyndon necklace $f_i^{k_i}$ of w. For any cyclic rotation of f_i, its k_i copies occur adjacently when writing all cyclic conjugates of all Lyndon factors in the ω-order, and thus correspond to adjacent characters in a run in $\mathsf{BBWT}(w)$. It follows that for any position in the last $k-1$ copies of f_i, the reference points to the corresponding position in the preceding copy. Thus, adjacent positions reference adjacent positions, and can be contained in the same phrase.

Next, consider a position in the first copy of f_i that does not correspond to a beginning of a same-character run in $\mathsf{BBWT}(w)$. Then, since the character at this position and the preceding (in ω-order) cyclic string is the same, their preceding positions must also correspond to adjacent positions in $\mathsf{BBWT}(w)$. This implies that, as long as the corresponding position is again not a beginning of a same-character run, adjacent positions will refer to adjacent positions, albeit, in the cyclic sense. Being the beginning of a same-character run in $\mathsf{BBWT}(w)$ can happen at most r_B times. Being adjacent in the cyclic sense, but not being adjacent in text-order can happen at most once per referenced Lyndon necklace. Thus, the number of times adjacent text positions can be in a different phrase is bounded by $O(r_B + \ell(w))$. Since $\ell(w) \leq r_B$ can be shown from Corollary 2. of [9], this concludes the proof. □

Example 1. For the string $w =$ `abbbabbababab`, we give below an example for the BMS computed from its BBWT. The Lyndon factorization of w is `abbb`, `abb`, `ab`, `ab`, `ab`. The number of runs r_B is 6. BBWT positions belonging to a referencing phrase are marked with ✓ in the last row of the table below. We therefore have 6 non-referencing phrases.

i	1	2	3	4	5	6	7	8	9	10	11	12	13
$w[i]$	a	b	b	b	a	b	b	a	b	a	b	a	b
$\mathsf{BBWT}[i]$	b	b	b	b	b	a	a	a	b	b	a	b	a
$\mathsf{CSA}[i]$	8	10	12	5	1	9	11	13	7	4	6	3	2
$\mathsf{CSA}[i]-1$	9	11	13	7	4	8	10	12	6	3	5	2	1
ref?		✓	✓	✓	✓		✓	✓		✓			

The referencing phrases are computed as follows: In the table above, CSA is the circular suffix array whose entry $\mathsf{CSA}[i]$ denotes the text position after the one from which we took $\mathsf{BBWT}[i]$ (or, if $\mathsf{BBWT}[i]$ belongs to the last character of a Lyndon factor F, the starting position of F). The row $\mathsf{CSA}[i]-1$ denotes the text position corresponding to $\mathsf{BBWT}[i]$. By construction of our BMS, the CSA entry positions 2-5, 7-8, and 10 correspond to referencing phrases, i.e., the text positions after 10,12,5,1,11,13,4 (applying CSA on these entry positions), i.e., the text positions 11, 13, 7, 4, 10, 12, 3. These positions refer to 9, 11, 13, 7, 8, 10, 6, respectively. If we group together neighboring positions that have the same offset to its reference, we obtain 9 phrases, which are visualized in Fig. 1.

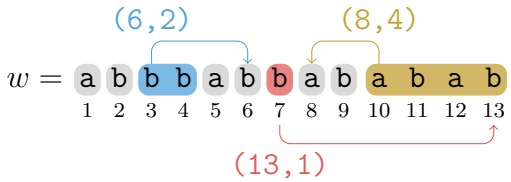

Fig. 1. The BMS factorization of Example 1

Theorem 1. *For any string,* $r_B = O(z \log^2 n)$.

Proof. (Sketch) We follow the proof of $r = O(z \log^2 n)$ by Kempa and Kociumaka [14]. The LCP array of a string w is an array of integers such that its i-th entry is the length of the longest common prefix (*lcp*) between the lexicographically $(i-1)$-th and i-th cyclic rotation of w. An *irreducible LCP position* is a position i such that $i = 1$ or $\mathsf{BWT}(w)[i-1] \neq \mathsf{BWT}(w)[i]$, and thus the number r of BWT runs is the number of irreducible LCP positions. For a multiset of primitive cyclic strings, we analogously define the ω-LCP array such that its i-th entry is the *lcp* between x^∞ and y^∞, where x and y are respectively the $(i-1)$-th and i-th string, in ω-order, among all cyclic rotations of all Lyndon factors of the Lyndon factorization of w. By construction, the number r_B of BBWT runs is the number of irreducible ω-LCP positions, i.e., $i = 1$ or $\mathsf{BBWT}(w)[i-1] \neq \mathsf{BBWT}(w)[i]$. Note that ω-LCP values can be infinite when there are Lyndon necklaces in the Lyndon factorization with exponent at least 2, but they can be safely disregarded since they are not irreducible. The theorem follows if we can show that for any value k, the number of irreducible ω-LCP values in $[k, 2k)$ is $O(z \log n)$ and considering $k = 2^i$ for $i = 0, \ldots, \lfloor \log n \rfloor$.

The arguments in the proof in [14] proceed by first asserting that for any integer k, a string contains at most $3kz$ distinct strings of length $3k$. Then, each irreducible LCP value in $[k, 2k)$ is associated with a cost of k, which are charged to positions in the at most $3kz$ strings that have an occurrence crossing the corresponding suffix array position, and it is shown that each substring can be charged at most $2 \log n$ times. The total cost is thus at most $6kz \log n$ and thus the number of irreducible LCP values is $O(z \log n)$.

For ω-LCP, the corresponding length $3k$ substring associated with the suffix array position may not occur in the original string but instead will correspond to a substring of some Lyndon necklace of the Lyndon factorization. Note that there are at most $3k$ distinct substrings of length $3k$ that are not substrings of the original string but a substring of a given Lyndon necklace. Since $\ell(w) < 4z$ [24], we have that the total number of such distinct substrings of length $3k$ that occur in this context is still bounded by $O(kz)$, and that the arguments still hold. □

Despite sharing common traits, r and r_B can be asymptotically different:

Theorem 2. *There exists a family of strings with* $r_B = \Omega(\log n)$ *and* $r = 2$.

Proof. Define the Fibonacci words as follows: $F_0 = \mathtt{b}, F_1 = \mathtt{a}, F_i = F_{i-1}F_{i-2}$. The infinite Fibonacci word is $\lim_{k\to\infty} F_k$. Melançon [19] showed that the k-th factor (the first factor being the 0-th) of the Lyndon factorization of the infinite Fibonacci word, has length f_{2k+2}, where $f_i = |F_i|$. Now, $\sum_{k=0}^{i} f_{2k+2} = -f_1 + (\cdots(((f_1 + f_2) + f_4) + f_6) + \cdots) + f_{2i+2} = f_{2i+3} - 1$. Therefore, the word obtained by deleting the last symbol of F_{2i+3} has $i+1$ distinct Lyndon factors. Noticing that the last symbol of F_{2i+3} must be '$\mathtt{a}$' and will form a distinct Lyndon factor, we have that the size of the Lyndon factorization of F_{2i+3} is $i+2$. Since $\ell(w) \leq r_B$ [9], the BBWT of the k-th Fibonacci word F_k for odd k has $\Omega(k)$ runs, while the BWT of any Fibonacci word has $r(F_k) = 2$ runs [18]. □

We have not yet been able to find a family of strings where $r_B = o(r)$.

4 RLBBWT and Rotation

Theorem 2 may give the impression that r may be a smaller measure compared to r_B. However, if we are to incorporate a rotation operation, which can be encoded as a single $\log n$-bit integer, we could possibly obtain a representation smaller than r using BBWT. This is because we have $r(x) = r_B(x)$ for the Lyndon rotation x of any primitive word. The Lyndon rotation of a primitive word can be computed in linear time [22].

For any string w, let $\hat{w} = \arg\min_{uv=w}\{r_B(vu)\}$ be the *optimal* rotation with respect to r_B. We observe that $\hat{w}$ is not always the Lyndon rotation of w. For example, for the Lyndon word $w = \mathtt{aaabaabaaabaabb}$, we have that $\mathsf{BWT}(w) = \mathsf{BBWT}(w) = \mathtt{bbbaabaaaabaaaa}$, thus $r_B(w) = 6$. However, we have that $\hat{w} = rot(w) = \mathtt{baaabaabaaabaab}$ and $\mathsf{BBWT}(\hat{w}) = \mathtt{bbbbaaaaaaaaaab}$, thus $r_B(\hat{w}) = 3$.

Since $\mathsf{BBWT}(w)$ (and hence $r_B(w)$) can be computed in $O(n)$ time, it is straightforward to compute $\hat{w}$ (and hence $r_B(\hat{w})$) in $O(n^2)$ time. A subquadratic time algorithm for this problem would be very interesting. (For LZ77, it was recently shown that indeed subquadratic time computation is possible [3].) While we have not yet been able to achieve this, we give a partial result: a linear time algorithm for computing the Lyndon factorizations, a precursor to computing BBWT, of all cyclic rotations.

Theorem 3. *We can compute the sizes of the Lyndon factorizations of all cyclic rotations of w in time linear in the length of w.*

Proof. (Sketch) Assume that w is Lyndon, consider the string $W = ww$, and view the cyclic rotations of w as substrings of length $|w|$ of W. Any Lyndon factorization of such a substring consists of the Lyndon factorization of a suffix of w and a prefix of w, since any $x = uv$ such that u is a suffix of w and v is a prefix of w cannot be Lyndon: it would imply $uv \prec v \prec w \prec u$, a contradiction.

We observe that the factors of the Lyndon factorization for any suffix of w, are the sequence of maximal right sub-trees of the standard (right) Lyndon tree [4] of w that are contained in the suffix. Similarly, for any prefix of w, they are the

maximal left sub-trees of the left Lyndon tree [2] of w that are contained in the prefix.

The right Lyndon tree of a Lyndon word w is a binary tree defined recursively as follows: if w is a single letter, it is a leaf, otherwise, the left and right child are respectively the right Lyndon trees of u, v where $w = uv$ and v is the longest proper suffix of w that is a Lyndon word. Note that it can be shown that this choice of v implies that u is a Lyndon word. The left Lyndon tree is defined analogously, but u is the longest proper prefix of w that is a Lyndon word. Similarly, it can be shown that this choice of u implies that v is a Lyndon word.

By definition of the Lyndon trees, and the property of the Lyndon factorization which states that the first (resp. last) factor is the longest prefix (resp. suffix) that is a Lyndon word, it is a simple observation that the Lyndon factorization of a suffix of w is exactly the sequence of maximal right nodes of the right Lyndon tree that are contained in the suffix, and the Lyndon factorization of a prefix of w is exactly the sequence of maximal left nodes of the left Lyndon tree that are contained in the prefix.

Both trees can be computed in linear time [4,2]. It is not difficult to see that the changes in the sequences, and thus the sizes of the Lyndon factorizations can be computed in total linear time for each of the suffixes and prefixes, by a left-to-right traversal on the trees. □

5 BBWT Reachability

Further developing the idea of combining rotation and BBWT in order to obtain a smaller representation of a string, we give the following conjecture.

Conjecture 1. Given two words with the same Parikh vector, we can transform one to the other by using only rotation and BBWT operations.

If true, this would suggest that it is possible, for example, to represent a string w based on w's Parikh vector, and a sequence of integers where each integer alternately represents the offset of the rotation or the number of times BBWT is applied, to reach w from the lexicographically smallest string with the same Parikh vector.

We have computationally confirmed the conjecture for ternary strings of up to length 17, with code available at https://github.com/koeppl/bbwtreachability, and have proved it for the specific cases where the alphabet is binary, or, when all symbols are distinct. We first give the following lemma which shows that, under the condition of the lemma, we can obtain a lexicographically smaller string using rotation operations and an inverse BBWT operation.

Lemma 2. *Let x be a word of length n whose smallest rotation is itself (i.e., a necklace), over the alphabet $\{c_1, \ldots, c_\sigma\}$ where $c_1 \prec \cdots \prec c_\sigma$, and let $(e_1, \ldots, e_\sigma)$ be the Parikh vector of x. Let $y = c_1^{e_1} \cdots c_\sigma^{e_\sigma}$, i.e., the lexicographically smallest string with the same Parikh vector as x, and $i = lcp(x, y)$. If $x[i] \neq x[n]$, then, there exists k such that $rot^k(\mathsf{BBWT}^{-1}(rot(x))) \prec x$.*

Proof. Note that $x[i] \neq x[n]$ implies $y \prec x$ since $y = x$ implies $i = n$. Let $x[1..i] = c_1^{e_1} \cdots c_k^{e'}$, where $e' \leq e_k$. This implies that symbols smaller than c_k are all used up in $x[1..i] = y[1..i]$, and cannot occur in $x[i+1..n]$ nor $y[i+1..n]$. Thus, for all $j \in [i+1, n]$, it holds that $x[j] \succeq x[i] = c_k$, in particular, for all $j \in [1, i]$, it holds that $x[n] \succ x[i] \succeq x[j]$ since $x[i] \neq x[n]$ is assumed.

Next, consider traversing the symbols of $\hat{x} = rot(x) = x[n]x[1..n-1]$ using the LF mapping $\psi_{\hat{x}}$ starting from position i' such that $\psi_{\hat{x}}(i') = i+1$ to recover a cyclic substring of $\mathsf{BBWT}^{-1}(\hat{x})$, whose smallest rotation (Lyndon rotation) will be a substring of $\mathsf{BBWT}^{-1}(\hat{x})$. Thus, we start from the symbol $\hat{x}[i'] = y[i+1]$. Since $\hat{x}[1] = x[n] \neq x[j] = y[j]$ for any $j \in [1..i]$ and $\hat{x}[2..i+1] = x[1..i] = y[1..i]$, it follows that $\psi_{\hat{x}}(j) = j - 1$ for any $j \in [2..i+1]$. Therefore, we have that $y[i+1]$ is prefixed by $x[1..i]$, i.e., $x[1..i]y[i+1] \prec x[1..i+1]$ is a cyclic substring of $\mathsf{BBWT}^{-1}(\hat{x})$, where the inequality follows from the definition of y and i. Since the length $i+1$ prefix of the Lyndon rotation of the whole cyclic substring that is retrieved by $\psi_{\hat{x}}$ starting from $i+1$ cannot be larger than $x[1..i]y[i+1]$, it follows that $\mathsf{BBWT}^{-1}(\hat{x})$ contains a length $i+1$ substring that is smaller than $x[1..i+1]$, and the lemma holds. □

Theorem 4. *Given two words of the same length with the same Parikh vector, it is possible to transform one to the other by using only rotations and* BBWT *transformations if all symbols are distinct, or if the alphabet is binary.*

Proof. Given any word, consider its smallest rotation x, and let y be the smallest word with the same Parikh vector. Since BBWT and rotations are bijections, it is easy to see that $\mathsf{BBWT}^{-1}(x)$ (resp. $rot^{-1}(x)$) can be represented by a sequence of $\mathsf{BBWT}(x)$ (resp. $rot(x)$) operations. Therefore, it suffices to show that we can reach y from x using any of these operations. If $y \prec x$, using Lemma 2, we can always obtain a strictly lexicographically smaller string using rotations and BBWT^{-1} and thus eventually reach y: when all symbols are distinct, it is easy to see that the condition of Lemma 2 holds. If the alphabet is binary, i.e., $\{\mathtt{a}, \mathtt{b}\}$, we have that $x[n] = \mathtt{b}$ since x is a smallest rotation. Furthermore, if $i = lcp(x, y)$, then, since $x[i] = \mathtt{b}$ would imply $x = y$, we have $x[i] = \mathtt{a} \neq \mathtt{b} = x[n]$. □

Acknowledgments

This work was supported by JSPS KAKENHI Grant Numbers JP24K02899 (HB) and JP23H04378 (DK).

References

1. Akagi, T., Funakoshi, M., Inenaga, S.: Sensitivity of string compressors and repetitiveness measures. Inf. Comput. **291**, 104999 (2023). https://doi.org/10.1016/J.IC.2022.104999, https://doi.org/10.1016/j.ic.2022.104999
2. Badkobeh, G., Crochemore, M.: Linear construction of a left Lyndon tree. Inf. Comput. **285**(Part), 104884 (2022). https://doi.org/10.1016/J.IC.2022.104884, https://doi.org/10.1016/j.ic.2022.104884
3. Bannai, H., Charalampopoulos, P., Radoszewski, J.: Maintaining the size of LZ77 on semi-dynamic strings. In: Inenaga, S., Puglisi, S.J. (eds.) 35th Annual Symposium on Combinatorial Pattern Matching, CPM 2024, June 25-27, 2024, Fukuoka, Japan. LIPIcs, vol. 296, pp. 3:1–3:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2024). https://doi.org/10.4230/LIPICS.CPM.2024.3, https://doi.org/10.4230/LIPIcs.CPM.2024.3
4. Bannai, H., I, T., Inenaga, S., Nakashima, Y., Takeda, M., Tsuruta, K.: The "runs" theorem. SIAM J. Comput. **46**(5), 1501–1514 (2017)
5. Bannai, H., Kärkkäinen, J., Köppl, D., Piątkowski, M.: Indexing the bijective BWT. In: Pisanti, N., Pissis, S.P. (eds.) 30th Annual Symposium on Combinatorial Pattern Matching, CPM 2019, June 18-20, 2019, Pisa, Italy. LIPIcs, vol. 128, pp. 17:1–17:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2019). https://doi.org/10.4230/LIPICS.CPM.2019.17, https://doi.org/10.4230/LIPIcs.CPM.2019.17
6. Bannai, H., Kärkkäinen, J., Köppl, D., Piątkowski, M.: Constructing and indexing the bijective and extended Burrows–Wheeler transform. Information and Computation **297**, 105153 (2024). https://doi.org/https://doi.org/10.1016/j.ic.2024.105153, https://www.sciencedirect.com/science/article/pii/S089054012400018X
7. Biagi, E., Cenzato, D., Lipták, Zs., Romana, G.: On the number of equal-letter runs of the bijective Burrows-Wheeler transform. In: Castiglione, G., Sciortino, M. (eds.) Proceedings of the 24th Italian Conference on Theoretical Computer Science, Palermo, Italy, September 13-15, 2023. CEUR Workshop Proceedings, vol. 3587, pp. 129–142. CEUR-WS.org (2023), https://ceur-ws.org/Vol-3587/4564.pdf
8. Boucher, C., Cenzato, D., Lipták, Zs., Rossi, M., Sciortino, M.: Computing the original eBWT faster, simpler, and with less memory. In: Lecroq, T., Touzet, H. (eds.) String Processing and Information Retrieval - 28th International Symposium, SPIRE 2021, Lille, France, October 4-6, 2021, Proceedings. Lecture Notes in Computer Science, vol. 12944, pp. 129–142. Springer (2021). https://doi.org/10.1007/978-3-030-86692-1_11, https://doi.org/10.1007/978-3-030-86692-1_11
9. Boucher, C., Cenzato, D., Lipták, Zs., Rossi, M., Sciortino, M.: r-indexing the eBWT. In: Proc. SPIRE. LNCS, vol. 12944, pp. 3–12 (2021)
10. Boucher, C., Cenzato, D., Lipták, Zs., Rossi, M., Sciortino, M.: r-indexing the eBWT. Information and Computation **298**, 105155 (2024). https://doi.org/https://doi.org/10.1016/j.ic.2024.105155, https://www.sciencedirect.com/science/article/pii/S0890540124000208
11. Burrows, M., Wheeler, D.J.: A block sorting lossless data compression algorithm. Tech. Rep. 124, Digital Equipment Corporation, Palo Alto, California (1994)
12. Chen, K.T., Fox, R.H., Lyndon, R.C.: Free differential calculus, IV. The quotient groups of the lower central series. Annals of Mathematics **68**(1), 81–95 (1958)
13. Gil, J.Y., Scott, D.A.: A bijective string sorting transform. CoRR **abs/1201.3077** (2012), http://arxiv.org/abs/1201.3077
14. Kempa, D., Kociumaka, T.: Resolution of the Burrows-Wheeler transform conjecture. Commun. ACM **65**(6), 91–98 (2022). https://doi.org/10.1145/3531445, https://doi.org/10.1145/3531445

15. Kufleitner, M.: On bijective variants of the Burrows–Wheeler transform. In: Proc. PSC. pp. 65–79 (2009)
16. Lyndon, R.C.: On Burnside's problem. Transactions of the American Mathematical Society **77**(2), 202–215 (1954)
17. Mantaci, S., Restivo, A., Rosone, G., Sciortino, M.: An extension of the Burrows–Wheeler transform. Theor. Comput. Sci. **387**(3), 298–312 (2007)
18. Mantaci, S., Restivo, A., Sciortino, M.: Burrows–Wheeler transform and Sturmian words. Inf. Process. Lett. **86**(5), 241–246 (2003)
19. Melançon, G.: Lyndon words and singular factors of Sturmian words. Theor. Comput. Sci. **218**(1), 41–59 (1999)
20. Navarro, G.: Indexing highly repetitive string collections, part I: repetitiveness measures. ACM Comput. Surv. **54**(2), 29:1–29:31 (2021)
21. Navarro, G., Ochoa, C., Prezza, N.: On the approximation ratio of ordered parsings. IEEE Trans. Inf. Theory **67**(2), 1008–1026 (2021)
22. Shiloach, Y.: Fast canonization of circular strings. J. Algorithms **2**(2), 107–121 (1981)
23. Storer, J.A., Szymanski, T.G.: Data compression via textual substitution. J. ACM **29**(4), 928–951 (1982). https://doi.org/10.1145/322344.322346, https://doi.org/10.1145/322344.322346
24. Urabe, Y., Nakashima, Y., Inenaga, S., Bannai, H., Takeda, M.: On the size of overlapping Lempel-Ziv and Lyndon factorizations. In: Pisanti, N., Pissis, S.P. (eds.) 30th Annual Symposium on Combinatorial Pattern Matching, CPM 2019, June 18-20, 2019, Pisa, Italy. LIPIcs, vol. 128, pp. 29:1–29:11. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2019). https://doi.org/10.4230/LIPICS.CPM.2019.29, https://doi.org/10.4230/LIPIcs.CPM.2019.29