On the number of equal-letter runs of the Bijective Burrows-Wheeler Transform

Elena Biagi[1], Davide Cenzato[2], Zsuzsanna Lipták[3], and Giuseppe Romana[4]

[1] University of Helsinki, Finland, elena.biagi@helsinki.fi
[2] University of Venice Ca' Foscari, Italy, davide.cenzato@unive.it
[3] University of Verona, Italy, zsuzsanna.liptak@univr.it
[4] University of Palermo, Italy, giuseppe.romana01@unipa.it

Abstract. The Bijective Burrows-Wheeler Transform (BBWT) is a variant of the famous BWT [Burrows and Wheeler, 1994]. The BBWT was introduced by Gil and Scott in 2012, and is based on the extended BWT of Mantaci et al. [TCS 2007] and on the Lyndon factorization of the input string. In the original paper, the compression achieved with the BBWT was shown to be competitive with that of the BWT, and it has been gaining interest in recent years. In this work, we present the first study of the number of runs r_B of the BBWT, which is a measure of its compression power. We exhibit an infinite family of strings on which r_B of the string and of its reverse differ by a multiplicative factor of $\Theta(\log n)$, where n is the length of the string. We also present experimental results and statistics on $r_B(s)$ and $r_B(s^{\text{rev}})$, as well as on the number of Lyndon factors in the Lyndon factorization of s and s^{rev}.

Keywords: Bijective Burrows-Wheeler Transform · BWT · eBWT · combinatorics on words · Lyndon factorization · compression.

1 Introduction

The Burrows-Wheeler-Transform (BWT) [7] is a fundamental invertible string transform originally introduced in 1994 by Michael Burrows and David J. Wheeler as a preprocessing step for string compression. The BWT tends to be easier to compress than the original input and supports efficient pattern matching tasks while keeping the transform in compressed form. Due to this, this transform has become the cornerstone of several string compressors and compressed data structures [14, 23] and several commonly used bioinformatics tools such as Bowtie [21, 20] and BWA [22].

These properties are due to the so-called clustering effect of the BWT. In fact, if the input text is highly repetitive, the final transform will tend to include few long runs of the same character, the number of which is usually denoted r. This motivated the interest for r as a parameter to measure text repetitiveness, with several recent papers studying the suitability of r as a repetitiveness measure as well as how r is related to other such measures [1, 17, 26]. Since a string and its reverse are repetitive in the same way, we would expect the number of runs of

their transformations to be the same. However, Giuliani et al. [16] showed that the parameter r is not invariant w.r.t. string reversal; in fact, there are strings s for which $r(s)$ and $r(s^{\text{rev}})$ differ by a multiplicative factor of $\Theta(\log n)$, where n is the length of the string.

The Bijective Burrows-Wheeler Transform (BBWT) [15, 19] is another invertible transformation, which is a variation of the BWT. It is defined as the extended BWT (eBWT) [24] of the factors of the Lyndon factorization of the input string. As opposed to the BWT, the BBWT is bijective: no two different words have the same BBWT, and every word is the BBWT of some word.

This transformation has met increasing interest in the last decade, with several papers published on this topic. Recently Bannai et al. in [3] presented the first linear time algorithm for constructing the Bijective BWT, thus unlocking the efficient computation of this transform for large inputs. In [18], it was further shown how to use in-place algorithms for constructing the BBWT and converting the BWT to the BBWT in quadratic time, thus highlighting a strong connection between these two transforms. Finally, in [2], Bannai et al. presented the first self-index based on the BBWT supporting efficient pattern-matching queries on the original input, similar to the original BWT. This comes with a small additional $\log(|P|)$ factor in the backward search algorithm, where P is the pattern.

In the original BBWT paper [15], the authors studied the suitability of this transform for text compression. Their experimental results show that on the Calgary corpus, the compression guaranteed by the BBWT is competitive. In particular, on average, the BBWT was about 1% more compressible than the BWT. The BBWT properties were further studied in subsequent papers, and it has inspired the definition of other bijective BWT variants [19, 10, 9]. However, the effectiveness of the number r_B of runs of the BBWT as a repetitiveness measure has not been studied before.

In this paper, we present the first results on r_B as a repetitiveness measure, comparing the behaviour of r_B of a string and of its reverse. We define an infinite family of words for which r_B of the string and its reverse differ by a multiplicative factor of $\Theta(\log n)$, where n is the length of the string, thus proving a parallel result on the BBWT to that of Giuliani et al. [16] on the BWT. This result shows the BBWT, as a measure of repetitiveness, exhibits the same defect as the BWT, namely that reversing the string may change it significantly, while repetitiveness is, of course, preserved. The family of strings used in this paper derive from finite Fibonacci words, as do those of [16], but the similarities end there; both the strings themselves and the proof techniques employed are different.

In the final part of the paper, we present experimental results on r_B , studying the multiplicative and additive difference between a string and its reverse, as well as the relationship to the number of factors of its Lyndon factorization.

Due to space constraints, some proofs have been omitted and will be included in the full version of this paper. A preliminary version of some of the results in this paper was contained in the first author's master thesis [5].

2 Basics

Let Σ be a finite ordered *alphabet* of size σ. A *string* (or *word*) over Σ is a finite sequence $w = w_1 \cdots w_n$ of *characters* w_i from Σ. We denote the length of string w as $|w|$. The *empty string* is the only string of length 0 and is denoted ε. For $n \geq 0$, Σ^n denotes the set of all words of length n, and $\Sigma^* = \cup_{n \geq 0} \Sigma^n$ the set of all finite words over Σ.

Let $w, u, x, v \in \Sigma^*$ such that $w = uxv$. Then u is called a *prefix*, x a *substring*, and v a *suffix* of w. A string t is a *subsequence* of w if t can be obtained from w by deleting some (possibly 0, possibly all) characters from w. A prefix (suffix, substring) u of a word w is a called *proper* if $u \neq w$. For a string u and an integer $k \geq 1$, $u^k = u \cdot u \cdots u$ denotes the k-fold concatenation of u. A string w is called *primitive* if $w = u^k$ implies $u = w$ and $k = 1$. If w is not primitive then it is called a *power*. Every word w can be uniquely written as $w = u^k$ for a primitive string u, called *root of* w. Given a string u, we also define the *infinite* word $u^\omega = u \cdot u \cdot u \cdots$ We denote the number of maximal equal-letter runs of a string w by $runs(w)$.

The *lexicographic order* on Σ^* is defined as follows: $u <_{\text{lex}} v$ if either u is a proper prefix of v, or if there exists $x \in \Sigma^*$ and $b, c \in \Sigma, b < c$ such that xb is a prefix of u and xc is a prefix of v. Another order relation on Σ^* is the *omega-order* defined as follows: Let $u = s^k$ and $v = t^\ell$, s, t primitive. Then $u <_\omega v$ if $s = t$ and $k < \ell$, or else if $s^\omega <_{\text{lex}} t^\omega$. Note that the lexicographic and the omega-order coincide on strings of the same length, but they can differ if one is a proper prefix of the other, e.g. $\texttt{ab} <_{\text{lex}} \texttt{aba}$ but $\texttt{aba} <_\omega \texttt{ab}$.

Two words w and w' are called *conjugates* (or *rotations*) if there exists u, v, possibly empty, s.t. $w = uv$ and $w' = vu$. Conjugacy is an equivalence relation. We denote the conjugacy class of a word $w \in \Sigma$, as $[w] = \{v \mid w \text{ and } v \text{ are conjugates}\}$. A word w is primitive if and only if its conjugacy class has cardinality $|w|$.

A primitive word is called a *Lyndon word* if it is lexicographically strictly smaller than all of its rotations. A *necklace* is a Lyndon word or a power of a Lyndon word. For a primitive word w, we denote by $L(w)$ the unique conjugate which is a Lyndon word (its *Lyndon rotation*). Every string w can be uniquely written as $w = f_1 \cdot f_2 \cdots f_m$ such that f_i are Lyndon words for $i = 1, \ldots, m$, and $f_1 \geq f_2 \geq \ldots \geq f_m$ [8]. This is called w's *Lyndon factorization*. A string s is Lyndon if and only if its Lyndon factorization consists of one factor only. The multiset of Lyndon factors in the Lyndon factorization of w is denoted $\mathcal{L}yn(w)$.

The *Burrows-Wheeler Transform* (BWT) [7] of a string s is a permutation of the characters of s, whose ith character is the last character of the ith rotation of s, where the rotations are taken w.r.t. lexicographic order. For example, $\text{BWT}(\texttt{banana}) = \texttt{nnbaaa}$, see Fig. 1. The number of runs of the BWT is denoted $r(s) = runs(\text{BWT}(s))$, e.g. $r(\texttt{banana}) = 3$.

The *extended BWT* (eBWT) [24] is a generalization of the BWT to a multiset of primitive strings $\mathcal{M}$: $\text{eBWT}(\mathcal{M})$ is a permutation of the characters of the strings in $\mathcal{M}$, whose ith character is the last character of the ith rotation, where the rotations of all strings in $\mathcal{M}$ are listed w.r.t. omega-order. For an example,

see Fig. 2. Note that for all strings s, $\text{eBWT}(\{s\}) = \text{BWT}(s)$. Next we list some known properties of the eBWT.

Lemma 1 (Properties of the eBWT [24]).

1. *Let $s \in \mathcal{S}$. Then $BWT(s)$ is a subsequence of $eBWT(\mathcal{S})$.*
2. *Let $\mathcal{S}$ be a multiset of primitive strings, and s' a conjugate of some $s \in \mathcal{S}$. Then the number of runs of $eBWT(\mathcal{S} \cup \{s'\})$ equals the number of runs of $eBWT(\mathcal{S})$.*
3. *Given an integer $m > 0$ and a primitive word s, let $\mathcal{S}$ be the multiset consisting of m copies of s. Then $BWT(s^m) = eBWT(\mathcal{S})$.*

3 The bijective BWT

Let $s = f_1 \cdot f_2 \cdots f_m$ be the Lyndon factorization of string s. Then $\text{BBWT}(s) = \text{eBWT}(\mathcal{M})$, where $\mathcal{M} = \mathcal{L}yn(s) = \{f_1, f_2, \ldots, f_m\}$ is the multiset of Lyndon factors of s. As an example, $\text{BWT}(\texttt{banana}) = \texttt{nnbaaa}$, while $\text{BBWT}(\texttt{banana}) = \texttt{annbaa}$, since the Lyndon factorization of `banana` is $\texttt{b} \cdot \texttt{an} \cdot \texttt{an} \cdot \texttt{a}$, and thus $\mathcal{L}yn(\texttt{banana}) = \{\texttt{a}, \texttt{an}, \texttt{an}, \texttt{b}\}$, see Fig. 1.

sorted rotations	BWT
abanan	n
anaban	n
ananab	b
banana	a
nabana	a
nanaba	a

omega-rotations	sorted rotations	BBWT
aaaa...	a	a
anan...	an	n
anan...	an	n
bbbb...	b	b
nana...	na	a
nana...	na	a

Fig. 1: BWT and BBWT of the word `banana`

First we look at under what circumstances the two transforms coincide:

Lemma 2. *$BWT(s) = BBWT(s)$ if and only if s is a necklace.*

Proof. First assume that $\text{BWT}(s) = \text{BBWT}(s)$ holds. Let $s = u^m$, where u is primitive and $m \geq 1$, and let t be the conjugate of s which is a necklace. Then clearly, $t = v^m$ with $v = L(u)$, and $\mathcal{L}yn(t)$ consists of m copies of v. Thus:

$$\text{BBWT}(t) \overset{\text{def.}}{=} \text{eBWT}(\mathcal{L}yn(t)) \overset{\text{Lemma 1}}{=} \text{BWT}(t) \overset{s,\,t \text{ conj.}}{=} \text{BWT}(s) = \text{BBWT}(s).$$

By bijectivity of the BBWT, this implies that s equals its own necklace rotation t, and is thus a necklace.

Conversely, if $s = u^m$ is a necklace, then $\mathcal{L}yn(s)$ consists of m copies of u, and $\text{BBWT}(s) = \text{eBWT}(\mathcal{L}yn(s)) = \text{BWT}(s)$, again by Lemma 1. □

Let us denote by $r_B(s) = runs(\mathrm{BBWT}(s))$, the number of runs of the BBWT of s. It is known that the number of runs $r(s)$ of the BWT of a binary string s is always even. This is because necessarily the first character must be $\mathtt{b}$, and the last character must be $\mathtt{a}$. This is not the case of the BBWT since $\mathtt{a}$ and $\mathtt{b}$ are the smallest and the greatest Lyndon factor, respectively, that a binary word can have, and therefore, BBWT may start and end with either $\mathtt{a}$ or $\mathtt{b}$. In the next lemma, we give the conditions that $\mathtt{a}$ or $\mathtt{b}$ appear as a Lyndon factor.

Lemma 3. *Let $s \in \{\mathtt{a},\mathtt{b}\}^*$ be a binary string, with $\mathtt{a} < \mathtt{b}$. Then, $\mathtt{a} \in \mathcal{L}yn(s)$ if and only if $\mathtt{a}$ is the last letter of s. Symmetrically, $\mathtt{b} \in \mathcal{L}yn(s)$ if and only if $\mathtt{b}$ is the first letter of s.*

Proof. We prove just the first statement on $\mathtt{a}$, and the other case is treated symmetrically. For the first direction, observe that $\mathtt{a}$ is the smallest Lyndon factor (both in lexicographical and ω order) that any binary word can have. Since the Lyndon factorization requires that any Lyndon factor must be greater or equal than the following in s (if any), $\mathtt{a} \in \mathcal{L}yn(s)$ implies that either the next letter in s is another $\mathtt{a}$, or there are no more letters in s. For the other direction, suppose by contradiction that s ends with an $\mathtt{a}$ and $\mathtt{a} \notin \mathcal{L}yn(s)$. Then, there exist $\ell \geq 0$, $m \geq 1$, $u \in \{\mathtt{a},\mathtt{b}\}^*$ such that the word $w = \mathtt{a}^\ell\mathtt{b}^m u\mathtt{a} \in \mathcal{L}yn(s)$, that is the Lyndon factor containing the last $\mathtt{a}$ of s. However, one can verify that $\mathtt{a}^{\ell+1}\mathtt{b}^m u < \mathtt{a}^\ell\mathtt{b}^m u\mathtt{a}$, which contradicts that w is a Lyndon word. □

We next characterize when $r_B(s)$ is odd.

Lemma 4. *Let $s \in \{\mathtt{a},\mathtt{b}\}^*$ be a binary word. It holds that $r_B(s)$ is odd if and only if s starts and ends with the same letter.*

However, it turns out that there is a simple connection between $r_B(s)$ and $r_B(s^{\mathrm{rev}})$, namely that they must have the same parity:

Lemma 5. *For every binary string s, the difference between the number of runs of the BBWT of s and the BBWT of its reverse is even.*

Proof. If s starts and ends with the same character, then so does s^{rev}, and by Lemma 4, both have an odd number of runs. If s starts and ends with different characters, then so does s^{rev}, and by Lemma 4, both have an even number of runs. □

The BWT of a word achieves maximal compression when it has as many runs as the size of the alphabet. In case of binary alphabets, it was shown in [25] that the perfect clustering effect of the BWT is a characterization for the family of *standard Sturmian words*. These words can be constructed through a *directive sequence*, an infinite sequence of integers $\{d_i\}_{i\geq 0}$ such that $d_0 \geq 0$ and $d_i > 0$ for all $i > 0$. The standard Sturmian words generated by this sequence are the words of the sequence $\{s_i\}_{i\geq 0}$ such that $s_0 = \mathtt{b}$, $s_1 = \mathtt{a}$, and $s_{i+1} = s_i^{d_{i-1}} s_{i-1}$ for $i > 1$. For instance, the directive sequence $1, 1, 1, 1, \ldots$ generates the well-known sequence of *finite Fibonacci words*: $s_0 = \mathtt{b}$, $s_1 = \mathtt{a}$, $s_2 = \mathtt{ab}$, $s_3 = \mathtt{aba}$, $s_4 = \mathtt{abaab}$, $s_5 = \mathtt{abaababa}$, $s_6 = \mathtt{abaababaabaab}$, $s_7 = \mathtt{abaababaabaababaababa}$, ...

Theorem 1 ([25]). *$BWT(s) = \mathtt{b}^k\mathtt{a}^\ell$, for some $k, \ell \geq 1$, if and only if s is the power of a rotation of a standard Sturmian word.*

We give a similar characterization for the BBWT. As opposed to the BWT, here it is not necessary that the transform begins with `b` and ends with `a`.

Lemma 6. *The number of runs of the BBWT of a string s is 2 if and only if (i) s has the form $\mathtt{b}^k\mathtt{a}^\ell$ for some $k, \ell \geq 1$, or (ii) s is the Lyndon rotation of a standard Sturmian word or a power of the Lyndon rotation of a standard Sturmian word.*

In the rest of the paper, we will look at the relationship between $r_B(s)$ and $r_B(s^{\text{rev}})$. To this end, we define the *runs-ratio* of string s as $\rho_B(s) = \max(\frac{r_B(s)}{r_B(s^{\text{rev}})}, \frac{r_B(s^{\text{rev}})}{r_B(s)})$, and the *runs-difference* of s as $\delta_B(s) = r_B(s) - r_B(s^{\text{rev}})$.

4 Fibonacci words

In this section, we will show that the Lyndon rotation of a Fibonacci word of order k has the following interesting property: the number of runs of the BBWT of its reverse has $2(k-2)$ runs, while the BBWT of the word itself has only 2 runs. Thus, ρ_B of these words is $\Theta(\log n)$, where n is the length of the word.

Fibonacci words were defined in the previous section. It follows directly from the definition that for all $k \geq 0$, $|s_k| = F_k$, where $\{F_k\}_{k\geq 0}$ is the well-known Fibonacci sequence $1, 1, 2, 3, 5, 8, 13, \ldots$ Fibonacci words have been studied extensively, see [11] for an overview. Some of the properties of Fibonacci words also follow from properties that have been shown for all standard words, see e.g. [13, 12, 4, 6, 25]. We next list some of these properties:

Proposition 1 (Some known properties of the Fibonacci words). *Let s_k be the Fibonacci word of order $k \geq 0$. Then there exists a palindrome x_k with the following properties:*

1. *for all $k \geq 2$: if k is even, then $s_k = x_k\mathtt{ab}$, and if k is odd, then $s_k = x_k\mathtt{ba}$, where x_k is a palindrome (in particular, $x_2 = \varepsilon$).*
2. *for all $k \geq 4$,*
   - *if k is even, then $s_k = x_{k-1}\mathtt{ba}x_{k-2}\mathtt{ab} = x_{k-2}\mathtt{ab}x_{k-1}\mathtt{ab}$, and*
   - *if k is odd, then $s_k = x_{k-1}\mathtt{ab}x_{k-2}\mathtt{ba} = x_{k-2}\mathtt{ba}x_{k-1}\mathtt{ba}$.*
3. *for all $k \geq 2$, $\mathtt{a}x_k\mathtt{b}$ is a Lyndon word.*
4. *for all k: $(s^k)^{rev}$ is a conjugate of s_k.*
5. *for $k \geq 2$: $BWT(s_k)$ has two runs; in particular, $BWT(s_k) = \mathtt{b}^{F_{k-2}}\mathtt{a}^{F_{k-1}}$.*
6. *for $k \geq 2$: $x_k\mathtt{ab}$ and $x_k\mathtt{ba}$, the so-called* central words*, are adjacent in the BW-matrix of s_k.*

From these we can easily deduce further properties. Recall that $L(s)$ is the Lyndon rotation of a primitive string s.

Corollary 1. *Let x_k be the palindrome from Prop. 1, i.e. $s_k = x_k\mathtt{ab}$ for k even, and $s_k = x_k\mathtt{ba}$ for k odd. Then*

1. $L(s_k) = \mathtt{a}x_k\mathtt{b}$,
2. $x_k = x_{k-1}\mathtt{ba}x_{k-2} = x_{k-2}\mathtt{ab}x_{k-1}$, *if* k *even,*
3. $x_k = x_{k-1}\mathtt{ab}x_{k-2} = x_{k-2}\mathtt{ba}x_{k-1}$, *if* k *odd.*

Since Lyndon rotations of Fibonacci words will be of central importance, we list the first few here: $L(s_0) = \mathtt{b}$, $L(s_1) = \mathtt{a}$, $L(s_2) = \mathtt{ab}$, $L(s_3) = \mathtt{aab}$, $L(s_4) = \mathtt{aabab}$, $L(s_5) = \mathtt{aabaabab}$.

Next we study the Lyndon factorization of the reverse of the Lyndon rotation of a Fibonacci word s_k. We will show that it consists of the Lyndon rotations of s_i for $i = 0, \ldots, k-2$, where the words of even order are listed increasingly (w.r.t. their order), followed by those of odd order listed decreasingly, followed by one additional factor $L(s_1) = \mathtt{a}$. Formally:

Lemma 7. *Let* $t_k = L(s_k)^{rev}$ *be the reverse of the Lyndon rotation of the Fibonacci word of order* k*. Then the Lyndon factorization of* t_k *is as follows:*

1. $t_k = L(s_0) \cdot L(s_2) \cdot L(s_4) \cdots L(s_{k-2}) \cdot L(s_{k-3}) \cdot L(s_{k-5}) \cdots L(s_3) \cdot L(s_1) \cdot L(s_1)$ *if* k *is even, and*
2. $t_k = L(s_0) \cdot L(s_2) \cdot L(s_4) \cdots L(s_{k-3}) \cdot L(s_{k-2}) \cdot L(s_{k-4}) \cdots L(s_3) \cdot L(s_1) \cdot L(s_1)$ *if* k *is odd.*

Example 1. In the following we list t_k with its Lyndon factorization, for $k = 2, \ldots, 8$:

- $t_2 = L(s_2)^{\mathrm{rev}} = \mathtt{b} \cdot \mathtt{a}$,
- $t_3 = L(s_3)^{\mathrm{rev}} = \mathtt{b} \cdot \mathtt{a} \cdot \mathtt{a}$,
- $t_4 = L(s_4)^{\mathrm{rev}} = \mathtt{b} \cdot \mathtt{ab} \cdot \mathtt{a} \cdot \mathtt{a}$,
- $t_5 = L(s_5)^{\mathrm{rev}} = \mathtt{b} \cdot \mathtt{ab} \cdot \mathtt{aab} \cdot \mathtt{a} \cdot \mathtt{a}$,
- $t_6 = L(s_6)^{\mathrm{rev}} = \mathtt{b} \cdot \mathtt{ab} \cdot \mathtt{aabab} \cdot \mathtt{aab} \cdot \mathtt{a} \cdot \mathtt{a}$,
- $t_7 = L(s_7)^{\mathrm{rev}} = \mathtt{b} \cdot \mathtt{ab} \cdot \mathtt{aabab} \cdot \mathtt{aabaabab} \cdot \mathtt{aab} \cdot \mathtt{a} \cdot \mathtt{a}$,
- $t_8 = L(s_8)^{\mathrm{rev}} = \mathtt{b} \cdot \mathtt{ab} \cdot \mathtt{aabab} \cdot \mathtt{aabaababaabab} \cdot \mathtt{aabaabab} \cdot \mathtt{aab} \cdot \mathtt{a} \cdot \mathtt{a}$.

For the proof of Lemma 7, we will first need two other lemmas. The first one gives a simple recursion for t_k:

Lemma 8. *Let* $t_k = (L(s_k))^{rev}$*. Then the following recursion holds for* $k \geq 2$:

- $t_k = t_{k-2}t_{k-1}$ *if* k *is even, and*
- $t_k = t_{k-1}t_{k-2}$ *if* k *is odd.*

Proof. First note that since $L(s_k) = \mathtt{a}x_k\mathtt{b}$, and x_k is a palindrome, therefore $t_k = \mathtt{b}x_k\mathtt{a}$. Let $k \geq 2$ be even. By Corollary 1, $t_k = \mathtt{b}x_k\mathtt{a} = \mathtt{b}x_{k-2}\mathtt{ab}x_{k-1}\mathtt{a} = t_{k-2}t_{k-1}$. Similarly, if k is odd, then $t_k = \mathtt{b}x_k\mathtt{a} = x_{k-1}\mathtt{ab}x_{k-2} = t_{k-1}t_{k-2}$. □

The second lemma gives a factorization of the Lyndon rotations of the s_k.

Lemma 9. *Let* $k \geq 4$*. Then the following holds:*

- $L(s_k) = L(s_{k-3})L(s_{k-5}) \cdots L(s_3)L(s_1)L(s_1)L(s_0)L(s_2)L(s_4) \cdots L(s_{k-2})$*, if* k *is even, and*

– $L(s_k) = L(s_{k-2})L(s_{k-4})\cdots L(s_3)L(s_1)L(s_1)L(s_0)L(s_2)L(s_4)\cdots L(s_{k-3})$, *if* k *is odd.*

Now we are ready to prove Lemma 7.

Proof (of Lemma 7). The proof is by induction on k. For the base cases, $t_2 = \mathtt{b}\cdot\mathtt{a}$, and $t_3 = \mathtt{b}\cdot\mathtt{a}\cdot\mathtt{a}$, as claimed. Now let $k > 3$. If k is even, then $t_k = t_{k-2}t_{k-1} = \mathtt{b}x_{k-2}\mathtt{ab}x_{k-1}\mathtt{a}$. Thus,

$$t_k = \underbrace{L(s_0)L(s_2)\cdots L(s_{k-4})L(s_{k-5})\cdots L(s_3)L(s_1)L(s_1)}_{t_{k-2}\text{ by the I.H.}}\cdot \underbrace{L(s_0)L(s_2)\cdots L(s_{k-4})L(s_{k-3})\cdots L(s_3)L(s_1)L(s_1)}_{t_{k-1}\text{ by the I.H.}}$$

$$= L(s_0)L(s_2)\cdots L(s_{k-4})\cdot \underbrace{L(s_{k-5})\cdots L(s_3)L(s_1)L(s_1)L(s_0)L(s_2)\cdots L(s_{k-4})}_{L(s_{k-2})\text{ by Lemma 9}}\cdot L(s_{k-3})\cdots L(s_3)L(s_1)L(s_1)$$

$$= L(s_0)\cdot L(s_2)\cdot L(s_4)\cdots L(s_{k-2})\cdot L(s_{k-3})\cdot L(s_{k-5})\cdots L(s_3)\cdot L(s_1)\cdot L(s_1).$$

The claim for k odd follows analogously. □

Example 2. For example, $t_8 = t_6\cdot t_7 = \mathtt{babaababaabaa}\cdot\mathtt{babaababaabaababababaa}$ $= \mathtt{b}\cdot\mathtt{ab}\cdot\mathtt{aabab}\cdot\underline{\mathtt{aabaababaabab}}\cdot\mathtt{aabaabab}\cdot\mathtt{aab}\cdot\mathtt{a}\cdot\mathtt{a}$, where the new Lyndon factor is underlined; its factorization from Lemma 9 is $L(s_6) = \mathtt{aabaababaabab}$ $= \mathtt{aab}\cdot\mathtt{a}\cdot\mathtt{a}\cdot\mathtt{b}\cdot\mathtt{ab}\cdot\mathtt{aabab}$.

Corollary 2 (from Lemma 7). *Let t_k be the reverse of the Lyndon rotation of the Fibonacci word of order k. Then $\mathcal{L}yn(t_k) = \{L(s_0), L(s_1), L(s_2), \ldots, L(s_{k-2})\}$ $\cup\{L(s_1)\}$, i.e. the factor $L(s_1) = \mathtt{a}$ appears with multiplicity 2 in the factorization, while all other factors appear exactly once.*

The final piece we need to prove the main theorem of this section will be Lemma 10, which gives the number of runs of the eBWT of the set of Fibonacci words $\mathcal{S}_k = \{s_0, s_1, \ldots, s_k\}$. A crucial role will be played by the central words, which we list up to order 7 in Table 1.

Table 1: Fibonacci central words of order k

k	2	3	4	5	6	7
u_k	ab	aab	abaab	abaabaab	abaababaabaab	abaababaabaababaabaab
v_k	ba	aba	ababa	abaababa	abaababaababa	abaababaabaababaababa

Lemma 10. *Let $k > 0$ and $\mathcal{S}_k$ be the set of Fibonacci words of order up to k, i.e. $\mathcal{S}_k = \{s_0, s_1, \ldots, s_k\}$. Then eBWT($\mathcal{S}_k$) has $2k$ runs.*

Proof. We prove the claim by induction on k. For $k = 1, 2$, eBWT($\{\mathtt{a}, \mathtt{b}\}$) = $\mathtt{ab}$, which has 2 runs, and eBWT($\{\mathtt{a}, \mathtt{b}, \mathtt{ab}\}$) = $\mathtt{abab}$, which has 4 runs, as claimed.

Now let $k > 2$. We will observe what happens to eBWT($\mathcal{S}_{k-1}$) when we insert s_k into $\mathcal{S}_{k-1}$ and will show that exactly 2 new runs are created, see Fig. 2.

k	sorted rotations	eBWT
	a	a
3	aab	b
3	aba	a
2	ab	b
	baa	a
2	ba	a
	b	b

k	sorted rotations	eBWT
	a	a
3	aab	b
	aabab	b
3	aba	a
4	abaab	b
4	ababa	a
2	ab	b
	baa	a
	baaba	a
	babaa	a
2	ba	a
	b	b

Fig. 2: Tables showing the eBWT matrix for $k = 3, 4$. Central words of order k are marked in the first column. Left: eBWT($\mathcal{S}_3$) = eBWT($\{\mathtt{b}, \mathtt{a}, \mathtt{ab}, \mathtt{aba}\}$) = $\mathtt{ababaab}$, right: eBWT($\mathcal{S}_4$) = eBWT($\{\mathtt{b}, \mathtt{a}, \mathtt{ab}, \mathtt{aba}, \mathtt{abaab}\}$) = $\mathtt{abbababaaaab}$.

Let eBWT($\mathcal{S}_{k-1}$) be given, and consider what happens when we add s_k. Denote the two central words of order k as $u_k = x_k\mathtt{ab}$ and $v_k = x_k\mathtt{ba}$. Note that $s_k = u_k$ if k is even, and $s_k = v_k$ if k is odd. Clearly, $u_k <_\omega v_k$ for all k. Moreover, it can be proven that for $k > 2$, the following relationship holds between central words of subsequent order: $u_k >_\omega v_{k-1}$ if k is even, and $v_k <_\omega u_{k-1}$ if k is odd.

Now, when considering eBWT($\mathcal{S}_{k-1}$), the two words u_k and v_k are inserted together, i.e. they are adjacent in the eBWT-matrix of $\mathcal{S}_k$. This is because they have the common prefix x_k, of length $|s_k| - 2$, which, for $k > 3$, is longer than the longest word previously present (of length $|s_{k-1}| = F_{k-1} < F_k - 2$); for $k = 3$ the claim can be seen by direct inspection (see Fig. 2, left).

Moreover, the two central words will be inserted immediately adjacent to one of the two central words of order $k - 1$. This is because one of the two central words is always equal to s_k (u_k for k even, v_k for k odd), and thus, u_{k-1} is a proper prefix of v_k or v_{k-1} is a proper prefix of u_k, and no longer prefix of these words can be present. Thus we have $s_k = u_k >_\omega v_{k-1} = s_{k-1}$ if k is even, and $s_k = v_k <_\omega u_{k-1} = s_{k-1}$ if k is odd. Therefore, u_k and v_k will be inserted immediately after $s_{k-1} = v_{k-1}$ if k is even, and immediately before $s_{k-1} = u_{k-1}$

if k is odd. It follows by induction that s_{k-1} was inserted immediately after (immediately before) s_{k-2} if k is even (if k is odd). This in turn implies that the word just before (just after) u_k and v_k is s_{k-2}, for k even (for k odd).

Now consider the number of runs of eBWT($\mathcal{S}_{k-1}$). By the induction hypothesis, eBWT($\mathcal{S}_{k-1}$) has $2k-2$ runs. Inserting the two central words creates two new runs. This is because they end in `b` and `a`, in this order, and they are inserted between the two previous Fibonacci words: if k is even, then they are inserted between s_{k-1}, which ends in `a`, and s_{k-2}, which ends in `b`; if k is odd, then between s_{k-2}, which ends in `a`, and s_{k-1}, which ends in `b`.

Note that, since s_k is a standard word, BWT(s_k) has the form $\mathtt{b}^k\mathtt{a}^\ell$, and by Lemma 1, this will be a subsequence of eBWT($\mathcal{S}_k$). This implies that, if the two central words of order k are inserted between, say, position i and position $i+1$ of eBWT($\mathcal{S}_{k-1}$), then all rotations of s_k ending in `b` will be inserted before i, and all rotations ending in `a` will be inserted after $i+1$. Inserting a `b` will create a new run only if it is inserted between two `a`'s; likewise, inserting an `a` will create a new run only if it is inserted between two `b`'s. It is not difficult to prove (by induction) that all `a`-runs before the position of s_{k-1} have length 1, and that all `b`-runs after the position of s_{k-1} have length 1.

Thus, exactly two new runs are created. This completes the proof. □

Theorem 2. *Let $w_k = L(s_k)$ be the Lyndon rotation of the kth Fibonacci word s_k. Then $\rho_B(w_k) = \Theta(\log |w_k|)$.*

Proof. First note that since w_k is a Lyndon word, BBWT(w_k) = BWT(w_k) by Lemma 2. Since it is a rotation of s_k, BWT(w_k) = BWT(s_k). By Prop. 1, BWT(s_k) has two runs, so BBWT(w_k) has two runs, thus, $r_B(w_k) = 2$.

Now let $t_k = (w_k)^{\text{rev}}$. By Corollary 2, the set of Lyndon factors of t_k is $\mathcal{S}_{k-2} = \{s_0, \ldots, s_{k-2}\}$. It follows from Lemma 1 that the number of runs of a multiset depends only on the set of the elements (and not on the multiplicity of each element). By Lemma 10, the number of runs of eBWT($\mathcal{S}_{k-2}$) is $2(k-2)$, and therefore, $r_B(t_k) = 2(k-2)$.

Finally, using the fact that $|w_k| = |s_k| = F_k$ grows exponentially in k, it follows that $\rho_B(w_k) = \frac{2(k-2)}{2} = \Theta(k) = \Theta(\log |w_k|)$. □

5 Experimental results

In our experiments, we studied the r_B parameter of a string s and its reverse, looking at both multiplicative (ρ_B) and additive difference (δ_B). We also studied the number of Lyndon factors in the Lyndon factorization of s. We considered only those strings s which are lexicographically strictly smaller than their reverse. We refer to such strings also as *forward strings*, as opposed to strings which are strictly larger than their reverse: we refer to these as *reverse strings*. This is to avoid repeating the same experiment twice (as we compare r_B of a string and of its reverse). Note that this experimental setup excludes palindromes.

We computed the BBWT of all forward strings for lengths between 3 and 25, over a binary alphabet. We also ran the same experiment over a ternary alphabet, for strings of length up to 15 (data not shown).

5.1 Multiplicative difference in r_B of a string and its reverse

In the first step of our analysis, $\rho_B(s)$ was calculated for each forward string s. We report the maximum ρ_B for each length over a binary alphabet in Table 2[1]. Our results show that increasing the sequence length n, the average runs-ratio as the proportion of sequence pairs having $\rho_B(s) = 1$ decreases. On the other hand, large sequence lengths generate large maximum $\rho_B(s)$ values. This suggests that by increasing n, we observe more sequences for which the $\rho_B(s)$ is very close to 1 and only a small subset for which $r_B(s)$ is very different from $r_B(s^{\text{rev}})$, i.e., we observe extremal words with larger $\rho_B(s)$ values. In addition, for every n up to 21, the most frequent value of $\rho_B(s)$ is 1, i.e. probability that $r_B(s) = r_B(s^{\text{rev}})$ is high (see Fig. 3 for $n = 21$).

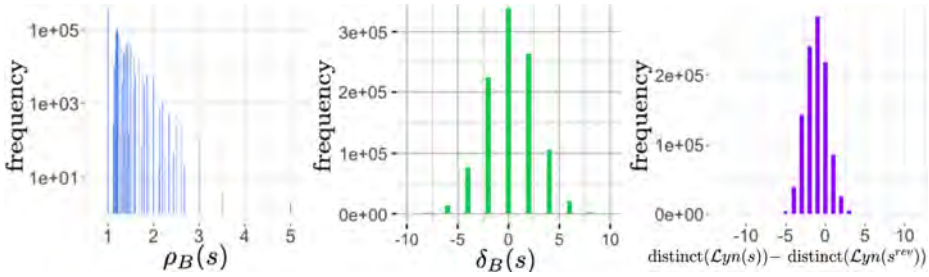

Fig. 3: Results for all 1047522 forward strings $s \in \Sigma^*$ where $\Sigma = \{\mathtt{a}, \mathtt{b}\}$ and $|s| = 21$. Left: histogram of $\rho_B(s)$; center: histogram of $\delta(s)$; right: histogram of the difference in the number of distinct Lyndon factors. Note that the left plot is in log-linear scale.

As for the extremal words, we noticed that several of them are Lyndon rotations of standard words (data not shown). In fact, these strings always have $r_B(s) = 2$, and thus they tend to generate large $\rho_B(s)$. In particular, for $\rho_B(n) \geq 4$ and $n \leq 23$, all extremal cases are Lyndon rotations of standard words. However, no precise pattern is visible since also non-standard words can be extremal words. On the other hand, if we consider the smallest n values, which present an increase in $\rho_B(n)$, for $\rho_B(n) \geq 3$, the extremal words are the Lyndon rotation of the Fibonacci word of length n and its reverse complement. For instance, $n = 8$ is the smallest n which allows to obtain $\rho_B(n) = 3$, and the two extremal cases are `aabaabab` and `ababbabb`. As for $\rho_B(n) \leq 2$, the situation appears more complex, since there are several extremal cases that reach the same runs-ratio.

[1] The statistics reported in the summary do not include palindrome sequences since their $\rho_B(s)$ is always 1. The number of palindromes of length n is $2^{\lceil n/2 \rceil}$.

Table 2: The maximum runs-ratio of binary strings of length n, for $n = 3, \ldots, 25$.

n	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
$\rho_B(n)$	1	2	2	2	2	3	2.33	3	3	3	4	3	3	4	3	4	4	3.5	5	4	4	4	4

5.2 Number of Lyndon factors and r_B

To conclude our experiment, we computed the number of Lyndon factors for all s and put that in relation to $r_B(s)$. In Fig. 4 we note that there are no strings with both a high number of Lyndon factors and a high number of runs as the top right corners of both graphs for forward (a) and reverse (b) are empty. The two plots are quite similar, with the exception of the number of Lyndon factors that appear to be higher in reverse strings (see also Fig. 3). In addition, the x-axis for forward and reverse is the same, indicating that values of r span the same range on both forward and reverse strings.

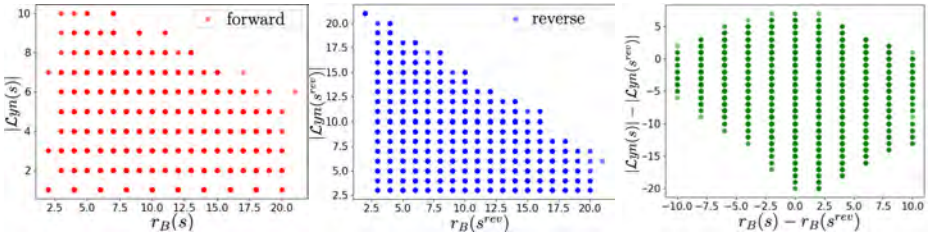

Fig. 4: Results for $s \in \Sigma^*$ where $\Sigma = \{\mathtt{a}, \mathtt{b}\}$ and $|s| = 21$. Left and center: scatter plots of $r_B(s)$ and number of Lyndon factors of s (left) and s^{rev} (center). Right: scatter plot of $\delta_B(s)$ and difference in number of Lyndon factors of s and s^{rev}.

We believe that the behavior described above might be related to the way we define s; since s is always lexicographically strictly smaller than its reverse, if s^{rev} has a run of $\mathtt{b}$ at the beginning and a run of $\mathtt{a}$ at the end the resulting Lyndon factorization will contain several length-one Lyndon factors. For instance, the Lyndon factorization of $\mathtt{bbbbababaababbaaaa}$ results in eight length-one Lyndon factors: $\mathtt{b} \cdot \mathtt{b} \cdot \mathtt{b} \cdot \mathtt{b} \cdot \mathtt{ab} \cdot \mathtt{ab} \cdot \mathtt{aababb} \cdot \mathtt{a} \cdot \mathtt{a} \cdot \mathtt{a} \cdot \mathtt{a}$. However, also in this case, no clear pattern arose from our experiments. In fact, there are cases where the Lyndon factorization of s leads to $r_B(s)$ which is much smaller than $r_B(s^{\text{rev}})$.

On strings over a binary alphabet, we can observe in the right plot in Fig. 4 that many strings are found on the vertical line indicating 0 as the difference in the number of runs, and the two measures analysed do not seem to strongly correlate. Results were inconclusive also when performing the same analysis counting only *distinct* Lyndon factors. Further details on the experimental results will be given in the full version of the paper.

References

1. T. Akagi, M. Funakoshi, and S. Inenaga. Sensitivity of string compressors and repetitiveness measures. *Inf. Comput.*, 291:104999, 2023.
2. H. Bannai, J. Kärkkäinen, D. Köppl, and M. Piatkowski. Indexing the Bijective BWT. In *Proc. of 30th Annual Symposium on Combinatorial Pattern Matching, CPM 2019*, volume 128 of *LIPIcs*, pages 17:1–17:14, 2019.
3. H. Bannai, J. Kärkkäinen, D. Köppl, and M. Piatkowski. Constructing the Bijective and the Extended Burrows-Wheeler Transform in linear time. In *Proc. of 32nd Annual Symposium on Combinatorial Pattern Matching, CPM 2021*, volume 191 of *LIPIcs*, pages 7:1–7:16, 2021.
4. J. Berstel and A. de Luca. Sturmian words, Lyndon words and trees. *Theor. Comp. Sci.*, 178(1-2):171–203, 1997.
5. E. Biagi. On comparing the Bijective Burrows-Wheeler-Transform of a word and its reverse, 2022.
6. J. Borel and C. Reutenauer. On Christoffel classes. *RAIRO Theor. Inf. Appl.*, 40(1):15–27, 2006.
7. M. Burrows and D. J. Wheeler. A block-sorting lossless data compression algorithm. Technical report, DIGITAL System Research Center, 1994.
8. K. T. Chen, R. H. Fox, and R. Lyndon. Free differential calculus, iv. the quotient groups of the lower central series. *Annals of Mathematics*, 68:81, 1958.
9. J. W. Daykin, R. Groult, Y. Guesnet, T. Lecroq, A. Lefebvre, M. Léonard, and É. Prieur-Gaston. A survey of string orderings and their application to the Burrows-Wheeler transform. *Theor. Comput. Sci.*, 710:52–65, 2018.
10. J. W. Daykin and W. F. Smyth. A bijective variant of the Burrows-Wheeler Transform using V-order. *Theor. Comput. Sci.*, 531:77–89, 2014.
11. A. de Luca. A combinatorial property of the Fibonacci words. *Inf. Process. Lett.*, 12(4):193–195, 1981.
12. A. de Luca. Sturmian words: Structure, combinatorics, and their arithmetics. *Theor. Comput. Sci.*, 183(1):45–82, 1997.
13. A. de Luca and F. Mignosi. Some combinatorial properties of Sturmian words. *Theor. Comput. Sci.*, 136(2):361–285, 1994.
14. T. Gagie, G. Navarro, and N. Prezza. Fully functional suffix trees and optimal text searching in BWT-runs bounded space. *ACM*, 67(1):2:1–2:54, 2020.
15. J. Y. Gil and D. A. Scott. A bijective string sorting transform. *CoRR*, abs/1201.3077, 2012.
16. S. Giuliani, S. Inenaga, Zs. Lipták, N. Prezza, M. Sciortino, and A. Toffanello. Novel results on the number of runs of the Burrows-Wheeler-Transform. In *Proc. of 47th International Conference on Current Trends in Theory and Practice of Computer Science, SOFSEM 2021*, volume 12607 of *Lecture Notes in Computer Science, LNCS*, pages 249–262, 2021.
17. S. Giuliani, S. Inenaga, Zs. Lipták, G. Romana, M. Sciortino, and C. Urbina. Bit catastrophes for the Burrows-Wheeler Transform. In *Proc. of Developments in Language Theory - 27th International Conference, DLT 2023*, volume 13911 of *Lecture Notes in Computer Science, LNCS*, pages 86–99, 2023.
18. D. Köppl, D. Hashimoto, D. Hendrian, and A. Shinohara. In-place Bijective Burrows-Wheeler Transforms. In *Proc. of 31st Annual Symposium on Combinatorial Pattern Matching, CPM 2020*, volume 161 of *LIPIcs*, pages 21:1–21:15, 2020.

19. M. Kufleitner. On bijective variants of the Burrows-Wheeler Transform. In *Proc. of the Prague Stringology Conference 2009*, pages 65–79, 2009.
20. B. Langmead and S. L. Salzberg. Fast gapped-read alignment with Bowtie 2. *Nature Methods*, 9(4):357–359, 2012.
21. B. Langmead, C. Trapnell, M. Pop, and S. L. Salzberg. Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. *Genome Biology*, 10:R25, 2009.
22. H. Li and R. Durbin. Fast and accurate long-read alignment with Burrows-Wheeler transform. *Bioinform.*, 26(5):589–595, 2010.
23. V. Mäkinen and G. Navarro. Succinct suffix arrays based on run-length encoding. *Nord. J. Comput.*, 12(1):40–66, 2005.
24. S. Mantaci, A. Restivo, G. Rosone, and M. Sciortino. An extension of the Burrows-Wheeler Transform. *Theor. Comput. Sci.*, 387(3):298–312, 2007.
25. S. Mantaci, A. Restivo, and M. Sciortino. Burrows-Wheeler transform and Sturmian words. *Inf. Process. Lett.*, 86(5):241–246, 2003.
26. G. Navarro. Indexing highly repetitive string collections, part I: repetitiveness measures. *ACM Comput. Surv.*, 54(2):29:1–29:31, 2022.