Discrete Applied Mathematics 355 (2024) 180–199

Contents lists available at ScienceDirect

Discrete Applied Mathematics

journal homepage: www.elsevier.com/locate/dam

On arithmetically progressed suffix arrays and related Burrows–Wheeler transforms ☆

Jacqueline W. Daykin [a,b,c], Dominik Köppl [d,*], David Kübel [e], Florian Stober [f]

[a] Department of Computer Science, Aberystwyth University, UK
[b] Univ Rouen Normandie, INSA Rouen Normandie, Université Le Havre Normandie, Normandie Univ, LITIS, 76000 Rouen, France
[c] Department of Information Science, Stellenbosch University, South Africa
[d] Department of Computer Science and Engineering, University of Yamanashi, Kōfu, Japan
[e] University of Bonn, Institute of Computer Science, Germany
[f] University of Stuttgart, Institute for Formal Methods of Computer Science (FMI), Germany

ARTICLE INFO

Article history:
Received 30 October 2021
Received in revised form 19 October 2023
Accepted 16 April 2024
Available online xxxx

Keywords:
Arithmetic progression
Burrows–Wheeler transform
Christoffel words
Suffix array
String combinatorics

ABSTRACT

We characterize those strings whose suffix arrays are based on arithmetic progressions, in particular, arithmetically progressed permutations where all pairs of successive entries of the permutation have the same difference modulo the respective string length. We show that an arithmetically progressed permutation P coincides with the suffix array of a unary, binary, or ternary string. We further analyze the conditions of a given P under which we can find a uniquely defined string over either a binary or ternary alphabet having P as its suffix array. For the binary case, we show its connection to lower Christoffel words, balanced words, and Fibonacci words. In addition to solving the arithmetically progressed suffix array problem, we give the shape of the Burrows–Wheeler transform of those strings solving this problem. These results give rise to numerous future research directions.

1. Introduction

The integral relationship between the suffix array [1] (SA) and Burrows–Wheeler transform [2] (BWT) is explored by Adjeroh et al. [3], who also illustrate the versatility of the BWT beyond its original motivation in lossless block compression [2]. The BWT has an injective property, that is, it computes a One-to-One mapping from the input to the transform, which enables recovery of input data in lossless compression scenarios. BWT applications include compressed index structures using backward search pattern matching, multimedia information retrieval, bioinformatics sequence processing, and it is at the heart of the bzip2 suite of text compressors. By its association with the BWT, this also indicates the importance of the SA data structure and hence our interest in exploring its combinatorial properties.

These combinatorial properties can be useful when checking the performance or integrity of string algorithms or data structures on string sequences in testbeds when properties of the employed string sequences are well understood. In particular, due to current trends involving massive data sets, indexing data structures need to work in external memory (e.g., [4]), or on distributed systems (e.g. [5]). For devising a solution adaptable to these scenarios, it is crucial to test whether the computed index (consisting of the suffix array or the BWT, for instance) is correctly stored on the hard disk or on the computing nodes, respectively. This test is more cumbersome than in the case of a single machine working only

☆ This paper is an extension of a contribution to the Prague Stringology Conference 2020, published in Daykin et al. (2020).

* Corresponding author.
E-mail addresses: jwd6@aber.ac.uk (J.W. Daykin), dkppl@yamanashi.ac.jp (D. Köppl), florian.stober@fmi.uni-stuttgart.de (F. Stober).

https://doi.org/10.1016/j.dam.2024.04.009

with its own RAM. One way to test is to compute the index for an instance, whose index shape can be easily verified. For example, one could check the validity of the computed BWT on a Fibonacci word since the shape of its BWT is known [6–8].

Other studies based on Fibonacci words are the suffix tree [9] or the Lempel–Ziv 77 (LZ77) factorization [10]. In [11], the suffix array and its inverse of each even Fibonacci word is studied as an arithmetic progression. In this study, the authors, like many at that time, did not append the artificial $ delimiter (also known as a sentinel) to the input string, thus allowing suffixes to be prefixes of other suffixes. This small fact makes the definition of $\mathsf{BWT}_\mathsf{T}[i] = \mathsf{T}[\mathsf{SA}_\mathsf{T}[i]-1]$ for a string T with suffix array SA_T incompatible with the traditional BWT defined on the BWT matrix, namely the lexicographic sorting of all cyclic rotations of the string T. Note that throughout this paper we use indices starting from 1. Consequently, we expect $i \in [1..n]$ in the preceding definition and for convenience we define $\mathsf{T}[0] := \mathsf{T}[n]$.

For instance, $\mathsf{SA}_{\mathtt{bab}} = [2,3,1]$ with $\mathsf{BWT}_{\mathtt{bab}} = \mathtt{bab}$, while $\mathsf{SA}_{\mathtt{bab\$}} = [4,2,3,1]$ and $\mathsf{BWT}_{\mathtt{bab\$}} = \mathtt{bba\$}$ with $\$ < \mathtt{a} < \mathtt{b}$. However, the traditional BWT constructed by reading the last characters of the lexicographically sorted cyclic rotations [ab*b*, ba*b*, bb*a*] of bab yields bba, which is equal to $\mathsf{BWT}_{\mathtt{bab\$}} = \mathtt{bba\$}$ after removing the $ character.

Note that not all strings are in the BWT image. An $O(n \log n)$-time algorithm is given by Giuliani et al. [12] for identifying all the positions in a string S where a $ can be inserted into so that S becomes the BWT image of a string ending with $.

Despite this incompatibility in the suffix array based definition of the BWT, we can still observe a regularity for even Fibonacci words [11, Sect. 5]. Similarly, both methods for constructing the BWT are compatible when the string T consists of a Lyndon word. The authors of [11, Remark 1] also observed similar characteristics for other, more peculiar string sequences. For the general case, the $ delimiter makes both methods equivalent, however the suffix array approach is typically preferred as it requires $O(n)$ time [13] compared to $O(n^2)$ with the BWT matrix method [2]. By utilizing combinatorial properties of the BWT, an in-place algorithm is given by Crochemore et al. [14], which avoids the need for explicit storage for the suffix sort and output arrays, and runs in $O(n^2)$ time using $O(1)$ extra memory (apart from storing the input text). Köppl et al. [15, Sect. 5.1] adapted this algorithm to compute the traditional BWT within the same space and time bounds.

Up to now, it has remained unknown whether we can formulate a class of string sequences for which we can give the shape of the suffix array as an arithmetic progression (independent of the $ delimiter). With this article, we catch up on this question, and establish a correspondence between strings and suffix arrays generated by arithmetic progressions. Calling a permutation of integers $[1..n]$ *arithmetically progressed* if all pairs of successive entries of the permutation have the same difference modulo n, we show that an arithmetically progressed permutation coincides with the suffix array of a unary, binary or ternary string. We analyze the conditions of a given arithmetically progressed permutation P under which we can find a uniquely defined string T over either a unary, a binary, or ternary alphabet having P as its suffix array.

The simplest case is for unary alphabets: Given the unary alphabet $\Sigma := \{\mathtt{a}\}$ and a string T of length n over Σ, $\mathsf{SA}_\mathsf{T} = [n, n-1, \ldots, 1]$ is an arithmetically progressed permutation with ratio $-1 \equiv n-1 \mod n$.

For the case of a binary alphabet $\{\mathtt{a},\mathtt{b}\}$, several strings of length n exist that solve the problem. Trivially, the solutions for the unary alphabet also solve the problem for the binary alphabet. However, studying those strings of length n whose suffix array is $[n, n-1, \ldots, 1]$, there are now multiple solutions: each $\mathsf{T} = \mathtt{b}^r\mathtt{a}^s$ with $r, s \in [0..n]$ such that $r+s=n$ has this suffix array. Similarly, $\mathsf{T} = \mathtt{a}^{n-1}\mathtt{b}$ has the suffix array $\mathsf{SA}_\mathsf{T} = [1,2,\ldots,n]$, which is an arithmetically progressed permutation with ratio 1.

A non-lexicographic ordering on strings, *the V-order*, considered for a modified FM-index [16], provides a curious example for the case with ratio -1: if S is a proper subsequence of a string T of length n, then S precedes T in V-order, written $\mathsf{S} \prec_\mathsf{V} \mathsf{T}$. This implies that $\mathsf{T}[n] \prec_\mathsf{V} \mathsf{T}[n-1..n] \prec_\mathsf{V} \cdots \prec_\mathsf{V} \mathsf{T}[1..n]$, so that $\mathsf{SA}_\mathsf{T}[i] = n-i+1$ for every $i \in [1..n]$, thus enabling trivial suffix sorting.

Clearly, any string ordering method that prioritizes minimal length within its definition will have a suffix array progression ratio of -1. Binary Gray codes[1] have this property and are ordered so that adjacent strings differ by exactly one bit [17]. These codes, which exhibit non-lexicographic order, have numerous applications, notably in error detection schemes such as flagging unexpected changes in data for digital communication systems, and logic circuit minimization. While it seems that Gray code order has not been applied directly to order the rows in a BWT matrix, just four years after the BWT appeared in 1994 [2], Chapin and Tate applied the concept when they investigated the effect of both alphabet and string reordering on BWT-based compressibility [18]. Their string sorting technique for the BWT matrix ordered the strings in a manner analogous to reflected Gray codes, but for more general alphabets. This modification inverted the sorting order for alternating character positions with which they demonstrated improved compression.

To date the only known BWT designed specifically for binary strings is the binary block order *B*-BWT [19]. This binary string sorting method prioritizes length, thus also exhibiting a suffix array with progression ratio -1.[2] Ordering strings of the same length, as applicable to forming the BWT matrix, is by a play on decreasing ($>$) and increasing ($<$) lexicographic ordering of the run length exponents of blocks of bits. Experimentation showed roughly equal performance to the original BWT in binary lexicographic order when comparing the number of character runs; however, there are a few instances where the *B*-BWT has significantly fewer runs.

[1] Originally known as *Reflected Binary Code* by the inventor Frank Gray.

[2] To see that the binary block order and the Gray code order are distinct: 1101 comes before 1110 in Gray code order, whereas, 1110 comes before 1101 in binary block order.

In what follows, we present a comprehensive analysis of strings whose suffix arrays are arithmetically progressed permutations (under the standard lexicographic order). In practice, such knowledge can reduce the $O(n)$ space for the suffix array to $O(1)$. Compared to the conference version [20], we here additionally analyze the shapes of BWTs of strings whose suffix arrays are arithmetically progressed. Additionally, we give applications for Christoffel words, balanced words, and meta strings. Finally, we extend our study on binary and ternary alphabets to general alphabets.

Arithmetic progressions have arisen in further applications of strings, and we conclude this section by mentioning a few. An effective approach in search-based tasks can be to reduce convolution problems, such as pattern matching with don't cares, or approximate matching, to polynomial multiplication. The practicality of complex convolutions-based bounded divide-and-conquer algorithms was investigated in [21] which demonstrated that, despite its theoretical efficiency, the convolution approach may not be practical across all problems and trade-offs. This led to devising a sequences method which was shown for some real problems to be a superior method to convolutions. In their experimentation, Amir et al. [21] included arithmetically progressed sequences. A lossless coding method distinct from Lempel–Ziv coding, known as *Arithmetic Progressions Tree coding* (APT), was proposed in [22]. The APT coding method is based on a tree-structure principle and involves finding arithmetic progressions in a given binary string. Empirical evidence showed APT coding to be asymptotically optimal up to a constant factor. Finding periodicity in real-world data often leads to insights on the structure of the data, and can be useful in prediction of future events or flagging anomalies in systems. Input to periodicity detection algorithms is typically a sequence of events, each associated with a timestamp where in most practical applications, these timestamps are imprecise — hence the study of approximate periodic arithmetic progressions. Motivated by naturally occurring periodic events in time intervals, such as load patterns on web servers, an algorithm to find the longest ϵ-relative error periodic pattern in a sequence is presented in [23] and runs in sub-cubic time for many practical situations. Here, reflecting the nature of real sequence data, the constant difference between any two consecutive numbers in the longest subsequence found allows for a tolerance of $\epsilon \in [0, 1)$ thus determining an approximate arithmetic progression.

The structure of the paper is as follows.[3] In Section 2 we give the basic definitions and background, and also deal with the elementary case of a unary alphabet. We present the main results in Section 3, where we cover ternary and binary alphabets, and consider inverse permutations. In Section 4, we illustrate the binary case for Christoffel words and in particular establish that every (lower) Christoffel word has an arithmetically progressed suffix array. We go on to link the binary characterization to balanced words and Fibonacci words. A characterization of strings with larger alphabets follows in Section 5. We overview the theme concepts for meta strings in Section 6. We conclude in Section 7 and propose a list of open problems and research directions, showing there is plenty of scope for further investigation. We proceed to the foundational concepts presented in the following section, starting with the case of a unary alphabet.

2. Preliminaries

Let Σ be an alphabet with size $\sigma := |\Sigma|$. An element of Σ is called a *character*.[4] Let Σ^+ denote the set of all nonempty finite strings over Σ. The *empty string* of length zero is denoted by ε; we write $\Sigma^* = \Sigma^+ \cup \{\varepsilon\}$. Given an integer $n \geq 1$, a *string*[5] of length n over Σ takes the form $\mathsf{T} = t_1 \cdots t_n$ with each $t_i \in \Sigma$. We write $\mathsf{T} = \mathsf{T}[1..n]$ with $\mathsf{T}[i] = t_i$. The length n of a string T is denoted by $|\mathsf{T}|$. If $\mathsf{T} = \mathsf{uwv}$ for some strings $\mathsf{u}, \mathsf{w}, \mathsf{v} \in \Sigma^*$, then u is a *prefix*, w is a *substring*, and v is a *suffix* of T; we say u (resp. w and v) is *proper* if $\mathsf{u} \neq \mathsf{T}$ (resp. $\mathsf{w} \neq \mathsf{T}$ and $\mathsf{v} \neq \mathsf{T}$). We say that a string T of length n has *period* $p \in [1..n-1]$ if $\mathsf{T}[i] = \mathsf{T}[i+p]$ for every $i \in [1..n-p]$ (note that we allow periods larger than $n/2$). If $\mathsf{T} = \mathsf{uv}$, then vu is said to be a *cyclic rotation* of T. A string T is said to be a *repetition* if and only if it has a factorization $\mathsf{T} = \mathsf{T}^k$ for some integer $k > 1$; otherwise, T is said to be *primitive*. A string that is both a proper prefix and a proper suffix of a string $\mathsf{T} \neq \varepsilon$ is called a *border* of T; a string is *border-free* if the only border it has is the empty string ε.

If Σ is a totally ordered alphabet with order $<$, then this order $<$ induces the *lexicographic ordering* $\prec$ on Σ^* such that $\mathsf{u} \prec \mathsf{v}$ for two strings $\mathsf{u}, \mathsf{v} \in \Sigma^*$ if and only if either u is a proper prefix of v, or $\mathsf{u} = \mathsf{r}a\mathsf{s}$, $\mathsf{v} = \mathsf{r}b\mathsf{t}$ for two characters $a, b \in \Sigma$ such that $a < b$ and for some strings $\mathsf{r}, \mathsf{s}, \mathsf{t} \in \Sigma^*$. In the following, we select a totally ordered alphabet Σ having three characters $\mathtt{a}, \mathtt{b}, \mathtt{c}$ with $\mathtt{a} < \mathtt{b} < \mathtt{c}$.

A string T is a *Lyndon word* if it is strictly least in the lexicographic order among all its cyclic rotations [24]. For instance, $\mathtt{abcac}$ and $\mathtt{aaacbaabaaacc}$ are Lyndon words, while the string $\mathtt{aaacbaabaaac}$ with border $\mathtt{aaac}$ is not.

A reciprocal relationship exists between the suffix array of a text and its Lyndon factorization, that is, the unique factorization of the text by greedily choosing the maximal length Lyndon prefix while processing the text from start to end: the Lyndon factorization of a text can be obtained from its suffix array [25]; conversely, the suffix array of a text can be constructed iteratively from its Lyndon factorization [26].

Lyndon words have numerous applications in combinatorics and algebra, and prove challenging entities due to their non-commutativity. Additionally, Lyndon words can arise naturally in Big Data — an instance in a biological sequence

[3] Compared to the conference version, we added applications to Christoffel words and balanced words, generalized our results for larger alphabets, and gave examples for indeterminate strings.

[4] Also known as *letter* or *symbol* in the literature.

[5] Also known as *word* in the literature.

over the DNA alphabet, with $\Sigma = \{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}\}$ and $\mathtt{A} < \mathtt{C} < \mathtt{G} < \mathtt{T}$, is the following substring occurring in a SARS-CoV-2 genome[6]:

AAAAACAGTAAAGTACAAATAGGAGAGTACACCTTTGAAAAAGGTGACTATGGTGAT

For the rest of the article, we take a string T of length $n \geq 2$. The suffix array $\mathsf{SA} := \mathsf{SA}_\mathsf{T}[1..n]$ of T is a permutation of the integers $[1..n]$ such that $\mathsf{T}[\mathsf{SA}[i]..n]$ is the ith lexicographically smallest suffix of T. We denote with ISA its inverse, i.e., $\mathsf{ISA}[\mathsf{SA}[i]] = i$. By definition, ISA is also a permutation. The string BWT with $\mathsf{BWT}[i] = \mathsf{T}[\mathsf{SA}[i] - 1 \bmod n]$ is the (SA-based) BWT of T. Here we shift the usual image $[0..n-1]$ of $x \mapsto x \mod n$ by one to admit values in $[1..n]$. Formally, $x \bmod n := x$ if $1 \leq x \leq n$, $x - n \bmod n$ for an integer $x > n$, and $x + n \bmod n$ for $x < 1$. In particular, $n \bmod n = 0 \bmod n = n$.

The focus of this paper is on arithmetic progressions. An *arithmetic progression* is a sequence of numbers such that the differences between all two consecutive terms are of the same value: Given an arithmetic progression $\{p_i\}_{i\geq 1}$, there is an integer $k \geq 1$ such that $p_{i+1} = p_i + k$ for all $i \geq 1$. We call k the *ratio* of this arithmetic progression. Similarly to sequences, we can define permutations that are based on arithmetic progressions: An *arithmetically progressed permutation* with ratio $k \in [1..n-1]$ is an array $P := [p_1, \ldots, p_n]$ with $p_{i+1} = p_i + k \mod n$ for all $i \in [1..n]$, where we stipulate that $p_{n+1} := p_1$.[7] In what follows, we want to study (a) strings whose suffix arrays are arithmetically progressed permutations, and (b) the shape of these suffix arrays. For a warm-up, we completely classify the case when the alphabet is unary with the following theorem.

Theorem 2.1. *Given the unary alphabet* $\{\mathtt{a}\}$*, the suffix array of a string of length* n *over* $\{\mathtt{a}\}$ *is uniquely defined by the arithmetically progressed permutation* $[n, n-1, \ldots, 1]$ *with ratio* $n-1$.

From now on, we stick to a general alphabet. While the suffix array $[n, n-1, \ldots, 1]$ in the statement of Theorem 2.1 uniquely defines the input string if the alphabet is unary, this is no longer the case if we allow larger alphabet sizes: Given the arithmetically progressed permutation $P = [n, n-1, \ldots, 1]$, we want to know the number of strings from a general totally ordered alphabet $\Sigma = [1..\sigma]$ with the natural order $1 < 2 < \cdots < \sigma$, having P as their suffix array. For that, we fix a string T of length n with $\mathsf{SA}_\mathsf{T} = P$. Let $s_j \geq 0$ be the number of occurrences of the character $j \in \Sigma$ appearing in T. Then $\sum_{j=1}^{\sigma} s_j = n$. For all $j < k$ it holds that $T[j] \geq T[k]$ because k appears before j in SA_T. Therefore, $\mathsf{T} = \sigma^{s_\sigma}(\sigma-1)^{s_{\sigma-1}}\cdots 1^{s_1}$ such that the position of the characters are uniquely determined.[8] In other words, we can reduce this problem to the classic stars and bars problem[9] [27, Chp. II, Sect. 5] with n stars and $\sigma - 1$ bars, yielding $\binom{n+\sigma-1}{n}$ possible strings. Hence we obtain:

Theorem 2.2. *There are* $\binom{n+\sigma-1}{n}$ *strings of length* n *over an alphabet with size* σ *having the suffix array* $[n, n-1, \ldots, 1]$.

As described above, strings of Theorem 2.2 have the form $\sigma^{s_\sigma}(\sigma-1)^{s_{\sigma-1}}\cdots 1^{s_1}$. The BWT based on the suffix array $[n, n-1, \ldots, 1]$ is $1^{s_1-1}2^{s_2}\cdots\sigma^{s_\sigma}1$. For $s_1 \geq 2$, it does not coincide with the BWT based on the rotations since the lexicographically smallest rotation is $1^{s_1}\sigma^{s_\sigma}\cdots 2^{s_2}$, and hence the first entry of this BWT is 2. For $s_1 = 1$, the last character '1' acts as the dollar sign being unique and least among all characters, making both BWT definitions equivalent.

For the rest of the analysis, we omit the arithmetically progressed permutation $[n, n-1, \ldots, 1]$ of ratio $k = n-1$ as this case is complete. All other permutations (including those of ratio $k = n-1$) are covered in our following theorems whose results we summarized in Fig. 1.

3. Arithmetically progressed suffix arrays

We start with the claim that each arithmetically progressed permutation coincides with the suffix array of a string on a ternary alphabet. Subsequently, given an arithmetically progressed permutation P, we show that either there is precisely one string T with $\mathsf{SA}_\mathsf{T} = P$ whose characters are drawn from a *ternary* alphabet, or, if there are multiple candidate strings, then there is precisely one whose characters are drawn from a *binary* alphabet. For this aim, we start with the restriction on k and n to be coprime.

3.1. Coprimality

Two integers are *coprime*[10] if their greatest common divisor (gcd) is one. An *ideal* $k\mathbb{N} := \{ki\}_{i\in\mathbb{N}}$ is a subgroup of $([1..n], +)$. It *generates* $[1..n]$ if $|k\mathbb{N}| = n$, i.e., $k\mathbb{N} = [1..n]$. Fixing one element $P[1] \in k\mathbb{N}$ of an ideal $k\mathbb{N}$ generating $[1..n]$ induces an arithmetically progressed permutation $P[1..n]$ with ratio k by setting $P[i+1] \leftarrow P[i]+k$ for every $i \in [1..n-1]$. On the contrary, each arithmetically progressed permutation with ratio k induces an ideal $k\mathbb{N}$ (the induced ideals are the

6 https://www.ncbi.nlm.nih.gov/nuccore/NC_045512.2?report=fasta.

7 We can also support negative values of k: Given a negative $k < 0$, we exchange it with $k' := n - k \mod n \in [1..n]$ and use k' instead of k.

8 Note that we slightly misused notation as the exponentiations of the characters being integers have to be understood as writing a character as many times as the exponent. So 1^{s_1} does not give 1 but a string of length s_1 having only 1's as characters.

9 This fundamental result in probability is useful in combinatorial counting problems involving variations of distributing q identical items into r distinguishable bins. This is equivalent to the number of permutations of q stars and $r-1$ bars and given by the binomial coefficient $\binom{q+r-1}{q}$.

10 Also known as *relatively prime* in the literature.

p_1	k	Min. σ	Properties of Strings	Reference
1		2	unique, Lyndon word	Theorem 3.9
$k+1$		2	unique, period $(n-k)$	Theorem 3.9
n	$\neq (n-1)$	2	unique, period $(n-k)$	Theorem 3.9
	$= (n-1)$	1	trivially periodic	Theorem 2.2
$\notin \{1, k+1, n\}$		3	unique	Theorem 3.2

Fig. 1. Characterization of strings whose suffix array is an arithmetic progression $P = [p_1, \ldots, p_n]$ of ratio k. The choice of p_1 determines the minimum size of the alphabet and whether a string is unique, periodic or a Lyndon word. The column *Min.* σ denotes the smallest possible size σ for which there exists such a string whose characters are drawn from an alphabet Σ with $|\Sigma| = \sigma$.

Rotation	T	P	p_1-k-1 mod n	$\mathsf{BWT_T}$
	1 2 3 4 5 6 7 8	1 2 3 4 5 6 7 8		
(1)	b a b b a b a c	[5, 2, 7| 4, 1, 6, 3| 8]	7	$\mathtt{b}^4\mathtt{c}\mathtt{a}^3$
(2)	b a b a c b a c	[2, 7, 4| 1, 6, 3| 8, 5]	4	$\mathtt{b}^3\mathtt{c}^2\mathtt{a}^3$
(3)	a c b a c b a c	[7, 4, 1| 6, 3| 8, 5, 2]	1	$\mathtt{b}^2\mathtt{c}^3\mathtt{a}^3$
(4)	a c b a c a c c	[4, 1, 6| 3| 8, 5, 2, 7]	6	$\mathtt{b}\mathtt{c}^4\mathtt{a}^3$
(5)	a b a b b a b b	[1, 6, 3| 8, 5, 2, 7, 4]	3	$\mathtt{b}^5\mathtt{a}^3$
(6)	c c a c c a c b	[6, 3| 8| 5, 2, 7, 4, 1]	8	$\mathtt{c}^5\mathtt{a}^2\mathtt{b}$
(7)	c c a c b c c b	[3| 8, 5| 2, 7, 4, 1, 6]	5	$\mathtt{c}^5\mathtt{a}\mathtt{b}^2$
(8)	b a b b a b b a	[8, 5, 2| 7, 4, 1, 6, 3]	2	$\mathtt{b}^5\mathtt{a}^3$

Fig. 2. T of Eq. (1) for each arithmetically progressed permutation P of length $n = 8$ with ratio $k = 5$, starting with $p_1 := P[1] = k = 5$. For $h \in [2..n]$, the permutation of the hth row is the $(h-1)$th cyclic rotation of the permutation P in the first row. The splitting of P into the subarrays is visualized by the vertical bar (|) symbol. For (5) and (8), the alphabet is binary and the BWTs are the same. The strings of (3) and (8) are periodic with period $n-k$, since the last text position of each subarray is at most as large as $n-k=3$ (*cf.* the proof of Theorem 3.9). For $i \in [1..n]$, $\mathsf{BWT_T}[i] = \mathsf{T}[P[i+n-k^{-1} \mod n]] = \mathsf{T}[P[i+3 \mod n]]$ with $k^{-1} = k = 5$ defined in Section 3.4.

same for two arithmetically progressed permutations that are shifted). Consequently, there is no arithmetically progressed permutation with ratio k if k and n are not coprime since in this case $\{(ki) \mod n \mid i \geq 1\} \subsetneq [1..n]$, from which we obtain:

Lemma 3.1. *The numbers k and n must be coprime if there exists an arithmetically progressed permutation of length n with ratio k.*

3.2. Ternary alphabet

Given an arithmetically progressed permutation $P := [p_1, \ldots, p_n]$ with ratio k, we define the ternary string $\mathsf{T}[1..n]$ by splitting P right after the values $n-k$ and $(p_1-k-1) \mod n$ into the three subarrays A, B, and C (one of which is possibly empty) such that $P = ABC$. Subsequently, we set

$$\mathsf{T}[p_i] := \begin{cases} \mathtt{a} & \text{if } p_i \in A, \text{ or} \\ \mathtt{b} & \text{if } p_i \in B, \text{ or} \\ \mathtt{c} & \text{if } p_i \in C. \end{cases} \tag{1}$$

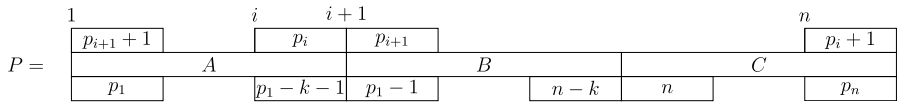

Fig. 3. Setting of the proof of Theorem 3.2 with the condition $p_i + 1 = p_1 - k = p_n$. In the figure we assume that the entry $p_1 - k - 1$ appears before $n - k$ in P.

BWT matrix of babbabac:	BWT matrix of ccaccacb:	BWT matrix of bbabbabb:
abacbabb	acbccacc	abbabbbb
abbabacb	accacbcc	abbbbabb
acbabbab	bccaccac	babbabbb
babacbab	cacbccac	babbbbab
babbabac	caccacbc	bbabbabb
bacbabba	cbccacca	bbabbbba
bbabacba	ccacbcca	bbbabbab
cbabbaba	ccaccacb	bbbbabba

Fig. 4. BWTs defined by the lexicographic sorting of all rotations of strings whose suffix arrays are cyclic rotations. This figure shows (from left to right) the BWT matrices of the strings of Rotation (1) and (6) of Fig. 2 as well as of Case (2) from Fig. 6. Reading the last column of a BWT matrix (whose characters are italic) from top downwards yields the BWT defined on the BWT matrix. While the BWT defined on the BWT matrix and the one defined by the suffix array coincides for the strings of Eq. (1) due to Theorem 3.3, this is not the case in general for the binary strings studied in Section 3.3, where we observe that $\mathsf{BWT}_{\mathtt{bbabbabb}} = \mathtt{bbbbbaab}$ defined by the suffix array differs from bbbbbaba (the last column on the right).

Fig. 2 gives examples for induced ternary/binary strings. Fig. 3 illustrates the case, where the value $(p_1 - k - 1) \bmod n$ is assumed to be left of $n - k$ in P.

Theorem 3.2. *Given an arithmetically progressed permutation $P \coloneqq [p_1, \ldots, p_n] \neq [n, n-1, \ldots, 1]$ with ratio k, $\mathsf{SA}_\mathsf{T} = P$ for T defined in Eq. (1).*

Proof. Suppose we have constructed SA_T. Since $\mathtt{a} < \mathtt{b} < \mathtt{c}$, according to the above assignment of T, the suffixes starting with a lexicographically precede the suffixes starting with b, which lexicographically precede the suffixes starting with c. Hence, $\mathsf{SA}[1..|A|]$, $\mathsf{SA}[|A|+1..|A|+|B|]$ and $\mathsf{SA}[|A|+|B|+1..n]$ store the same text positions as A, B, and C, respectively. Consequently, it remains to show that the entries of each subarray (A, B or C) are also sorted appropriately. Let p_i and p_{i+1} be two neighboring entries within the same subarray. Thus, $\mathsf{T}[p_i] = \mathsf{T}[p_{i+1}]$ holds, and the lexicographic order of their corresponding suffixes $\mathsf{T}[p_i..n]$ and $\mathsf{T}[p_{i+1}..n]$ is determined by comparing the subsequent positions, starting with $\mathsf{T}[p_i+1]$ and $\mathsf{T}[p_{i+1}+1]$. Since we have $(p_{i+1}+1)-(p_i+1) = p_{i+1}-p_i = k$, we can recursively show that these entries remain in the same order next to each other in the suffix array until either reaching the last array entry or a subarray split, that is, (1) $p_i + 1 = p_1 - k$ or (2) $p_i = n - k$.

(1) When $p_i + 1$ becomes $p_1 - k \mod n = p_n$ (the last entry in SA), p_{i+1} is in the subsequent subarray of the subarray of $p_i = p_1 - k - 1$ (remember that A or B ends directly after $p_1 - k - 1$, *cf.* Fig. 3). Hence $\mathsf{T}[p_i] < \mathsf{T}[p_{i+1}]$, and $\mathsf{T}[p_i..n] \prec \mathsf{T}[p_{i+1}..n]$.

(2) The split at the value $n - k$ ensures that when reaching $p_i = n - k$ and $p_{i+1} = n$, we can stop the comparison here as there is no character following $\mathsf{T}[p_{i+1}]$. The split here ensures that we can compare the suffixes $\mathsf{T}[n-k..n]$ and $\mathsf{T}[n]$ by the characters $\mathsf{T}[n-k] < \mathsf{T}[n]$. If we did not split here, $\mathsf{T}[n] = \mathsf{T}[n-k]$, and the suffix $\mathsf{T}[n]$ would be a prefix of $\mathsf{T}[n-k..n]$, resulting in $\mathsf{T}[n] \prec \mathsf{T}[n-k..n]$ (which yields a contradiction unless $p_1 = n$).

To sum up, the text positions stored in each of A, B and C are in the same order as in SA_T since the $j-1$ subsequent text positions of each consecutive pair of entries p_i and p_{i+1} are consecutive in P for the smallest integer $j \in [1..n]$ such that $p_{i+1} + jk \in \{p_1 - 1, n\}$. □

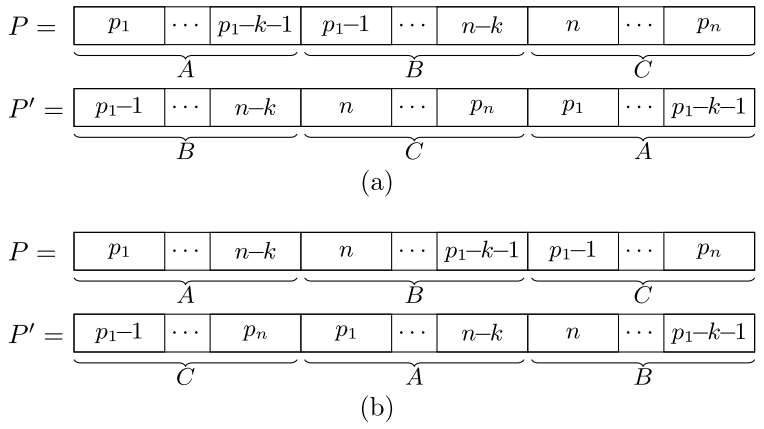

Fig. 5. Setting of Eq. (1) with the distinction whether the entry $p_1 - k - 1$ appears before (a) or after (b) $n - k$ in P, yielding a different shape of the BWT_T defined as $\mathsf{BWT}_{\mathsf{T}[i]} = \mathsf{T}[P'[i]]$ with $P'[i] = \mathsf{SA}_\mathsf{T}[i] - 1 \mod n$.

Knowing the suffix array of the ternary string T of Eq. (1), we can give a characterization of its BWT. We start with the observation that both BWT definitions (rotation based and suffix array based) coincide for the strings of Eq. (1) (but do not in general as highlighted in the introduction, *cf.* Fig. 4), and then continue with insights in how the BWT looks like.

Theorem 3.3. *Given an arithmetically progressed permutation $P := [p_1, \ldots, p_n]$ with ratio $k \neq n - 1$ and the string T of Eq. (1), the BWT of T defined on the BWT matrix coincides with the BWT of T defined on the suffix array.*

Proof. According to Theorem 3.2, $\mathsf{SA}_\mathsf{T} = P$, and therefore the BWT of T defined on the suffix array is given by $\mathsf{BWT}_\mathsf{T}[i] = \mathsf{T}[p_i - 1 \bmod n]$. The BWT matrix is constituted of the lexicographically ordered cyclic rotations of T. The BWT based on the BWT matrix, denoted by $\mathsf{BWT}_{\text{matrix}}$, is obtained by reading the last column of the BWT matrix from top downwards (see Fig. 4). Formally, let $Q[i]$ be the starting position of the lexicographically ith smallest rotation $\mathsf{T}[Q[i]..n]\mathsf{T}[1..Q[i]-1]$. Then $\mathsf{BWT}_{\text{matrix}}[i]$ is $\mathsf{T}[Q[i] - 1 \mod n]$ if $Q[i] > 1$, or $\mathsf{T}[n]$ if $Q[i] = 1$. We prove the equality $P = Q$ by showing that, for all $i \in [1..n-1]$, the rotation $R_i := \mathsf{T}[p_i..n]\mathsf{T}[1..p_i-1]$ starting at $p_i = \mathsf{SA}_\mathsf{T}[i]$ is lexicographically smaller than the rotation $R_{i+1} := \mathsf{T}[p_{i+1}..n]\mathsf{T}[1..p_{i+1} - 1]$ starting at $p_{i+1} = \mathsf{SA}_\mathsf{T}[i+1]$. We do that by comparing both rotations R_i and R_{i+1} characterwise:

Let j be the first position where R_i and R_{i+1} differ, i.e., $R_i[j] \neq R_{i+1}[j]$ and $R_i[q] = R_{i+1}[q]$ for every $q \in [1..j)$.

First we show that $j \neq p_n - p_i + 1 \bmod n$ by a contradiction: Assuming that $j = p_n - p_i + 1 \bmod n$, we conclude that $j \neq 1$ by the definition of $i \in [1..n-1]$. Since k is the ratio of P, we have

$$R_i[j-1] = \mathsf{T}[p_i + j - 2 \bmod n] = \mathsf{T}[p_n - 1 \bmod n] = \mathsf{T}[p_1 - k - 1 \bmod n]$$

and $R_{i+1}[j-1] = \mathsf{T}[p_1 - 1 \bmod n]$. By Eq. (1), $p_1 - k - 1 \mod n$ and $p_1 - 1 \mod n$ belong to different subarrays of P, therefore $\mathsf{T}[p_1 - k - 1] \neq \mathsf{T}[p_1 - 1]$ and $R_i[j-1] \neq R_{i+1}[j-1]$, contradicting the choice of j as the first position where R_i and R_{i+1} differ.

This concludes that $j \leq n$ (hence, $R_i \neq R_{i+1}$) and $j \neq p_n - p_i + 1 \bmod n$. Hence, $R_i[j] = \mathsf{T}[p_i + j - 1 \bmod n]$ and $R_{i+1}[j] = \mathsf{T}[p_{i+1} + j - 1 \bmod n] = \mathsf{T}[p_i + j - 1 + k \bmod n]$ are characters given by two consecutive entries in SA_T, i.e., $\mathsf{SA}_\mathsf{T}[q] = p_i + j - 1 \bmod n$ and $\mathsf{SA}_\mathsf{T}[q+1] = p_i + j - 1 + k \bmod n$ for a $q \in [1..n-1]$. Thus $R_i[j] \leq R_{i+1}[j]$, and by definition of j we have $R_i[j] < R_{i+1}[j]$, leading finally to $R_i \prec R_{i+1}$. Hence, $Q = P$. □

Lemma 3.4. *Let $P := [p_1, \ldots, p_n]$ be an arithmetically progressed permutation with ratio $k \neq n - 1$. Further, let $\mathsf{T}[1..n]$ be given by Eq. (1) such that $\mathsf{SA}_\mathsf{T} = P$ according to Theorem 3.2. Given that $p_t = p_1 - k - 1 \mod n$ for a $t \in [1..n]$, BWT_T is given by the tth rotation of $\mathsf{T}[\mathsf{SA}[1]] \cdots \mathsf{T}[\mathsf{SA}[n]]$, i.e., $\mathsf{BWT}_\mathsf{T}[i] = \mathsf{T}[P[i + t \mod n]]$ for $i \in [1..n]$.*

Proof. Since P is an arithmetically progressed permutation with ratio k then so is the sequence $P' := [p'_1, \ldots, p'_n]$ with $p'_i = p_i - 1 \mod n$. In particular, P' is a cyclic shift of P with $p'_n = p_1 - 1 - k \mod n$ because $p'_1 = p_1 - 1$. However, $p'_n = p_n - 1 = p_1 - 1 - k$ is a split position of one of the subarrays A, B, or C, meaning that P' starts with one of these subarrays and ends with another of them (*cf.* Fig. 5). Consequently, there is a t such that $p_t = p'_n$, and we have the property that BWT_T with $\mathsf{BWT}_\mathsf{T}[i] = \mathsf{T}[P'[i]]$ is the tth rotation of $\mathsf{T}[\mathsf{SA}[1]] \cdots \mathsf{T}[\mathsf{SA}[n]]$. □

We will determine the parameter $t = n - k^{-1} \mod n$ after Eq. (3) in Section 3.4, where k^{-1} is defined such that $k \cdot k^{-1} \mod n = 1 \mod n$. With Lemma 3.4, we obtain the following corollary which shows that the number of runs in BWT_T for T defined in Eq. (1) is minimal:

Corollary 3.5. *For an arithmetically progressed permutation $P := [p_1, \dots, p_n]$ with ratio $k \neq n-1$ and the string T defined by Eq. (1), $\mathsf{BWT_T}$ consists of exactly 2 runs if T is binary, while it consists of exactly 3 runs if T is ternary.*

Theorem 3.6. *Given an arithmetically progressed permutation $P := [p_1, \dots, p_n]$ with ratio k such that $p_1 \notin \{1, k+1, n\}$, the string T given in Eq. (1) is unique.*

Proof. The only possible way to define another string T' would be to change the borders of the subarrays A, B, and C. Since $p_1 \notin \{1, n\}$, $n-k$ and n, as well as $p_1 - k - 1$ and $p_1 - 1$, are stored as a consecutive pair of text positions in P.

- If P is not split between its consecutive text positions $n-k$ and n, then $\mathsf{T}'[n-k] = \mathsf{T}'[n]$. Consequently, we have the contradiction $\mathsf{T}'[n] \prec \mathsf{T}'[n-k..n]$.
- If P is not split between its consecutive text positions $(p_1 - k - 1) \mod n$ and $(p_1 - 1) \mod n$, then $\mathsf{T}'[p_1 - k - 1 \mod n] = \mathsf{T}'[p_1 - 1 \mod n]$. Since $p_1 \neq k+1$, and $\mathsf{T}'[p_1 - k \mod n] = \mathsf{T}'[p_n] > \mathsf{T}'[p_1]$, this leads to the contradiction $\mathsf{T}'[(p_1 - k - 1 \bmod n)..n] \succ \mathsf{T}'[(p_1 - 1 \bmod n)..n]$, *cf.* Fig. 3. □

Following this analysis of the ternary case we proceed to consider binary strings. A preliminary observation is given in Fig. 2, which shows, for the cases p_1 is 1 and n in Theorem 3.6, namely Rotations (5) and (8), that a rotation of $n-k$ in the permutation gives a rotation of one in the corresponding binary strings. We formalize this observation in the following lemma, drawing a connection between binary strings whose suffix arrays are arithmetically progressed and start with 1 or n.

Lemma 3.7. *Let $P := [p_1, \dots, p_n]$ be an arithmetically progressed permutation with ratio k and $p_1 = 1$ for a binary string T over $\Sigma = \{\mathtt{a}, \mathtt{b}\}$ with $\mathsf{SA_T} = P$. Suppose that the number of $\mathtt{a}$'s in T is m and that $\mathsf{T}' = \mathsf{T}[2] \cdots \mathsf{T}[n]\mathsf{T}[1]$ is the first rotation of T. Then $\mathsf{SA_{T'}}$ is the mth rotation of P with $\mathsf{SA_{T'}}[1] = n$. Furthermore, $\mathsf{BWT_T} = \mathsf{BWT_{T'}}$.*

Proof. Since $p_1 = 1$, $\mathsf{T}[1] = \mathtt{a}$ and $\mathsf{T}[n] = \mathtt{b}$. In the following, we show that $P' = \mathsf{SA_{T'}}$ for $P' := [p'_1, \dots, p'_n] := [p_1 - 1 \bmod n, \dots, p_n - 1 \bmod n]$ with $p'_1 = p_1 - 1 = n$ (since $\mathsf{T}'[n] = \mathtt{a}$). For that, we show that each pair of suffixes in $\mathsf{SA_T}$ is kept in the same relative order in P' (excluding $\mathsf{SA_T}[1] = 1$):

Consider two text positions $p_i, p_j \in [p_2, \dots, p_n]$ with $\mathsf{T}[p_i..n] = u_1 \cdots u_s \prec \mathsf{T}[p_j..n] = v_1 \cdots v_t$.

- If $u_h \neq v_h$ for the least $h \in [1..\min\{s,t\}]$, then $u_1 \cdots u_s\mathtt{a} \prec v_1 \cdots v_t\mathtt{a}$.
- Otherwise, $u_1 \cdots u_s$ is a proper prefix of $v_1 \cdots v_t = u_1 \cdots u_s v_{s+1} \cdots v_t$.
  - If $v_{s+1} = \mathtt{b}$, then $u_1 \cdots u_s\mathtt{a} \prec u_1 \cdots u_s\mathtt{b}v_{s+2} \cdots v_t\mathtt{a} = v_1 \cdots v_t\mathtt{a}$.
  - Otherwise ($v_{s+1} = \mathtt{a}$), $u_1 \cdots u_s\mathtt{a}$ is a proper prefix of $v_1 \cdots v_t\mathtt{a}$, and similarly $u_1 \cdots u_s\mathtt{a} \prec u_1 \cdots u_s\mathtt{a}v_{s+2} \cdots v_t\mathtt{a} = v_1 \cdots v_t\mathtt{a}$.

Hence the relative order of these suffixes given by $[p_2, \dots, p_n]$ and $[p'_2, \dots, p'_n]$ is the same. In total, we have $p'_i = p_i - 1 \bmod n$ for $i \in [1..n]$, hence P' is an arithmetically progressed permutation with ratio k. Given the first m entries in P represent all suffixes of T starting with $\mathtt{a}$, P' is the mth rotation of P since $p'_1 = n$ is the $(m+1)$th entry of P, i.e., the smallest suffix starting with $\mathtt{b}$ in T. Finally, since the strings T and T' are rotations of each other, their BWTs are the same. □

Like the parameter t of Lemma 3.4, we will determine the parameter m after Eq. (3) in Section 3.4.

3.3. Binary alphabet

We start with the construction of a binary string from an arithmetically progressed permutation:

Theorem 3.8. *Given an arithmetically progressed permutation $P := [p_1, \dots, p_n]$ with ratio $k \neq n-1$ such that $p_1 \in \{1, k+1, n\}$, we can modify T of Eq. (1) to be a string over the binary alphabet $\{\mathtt{a}, \mathtt{b}\}$ with $\mathsf{SA_T} = P$.*

Proof. If $p_1 = 1$, then P is split after the occurrences of the values $n-k$ and $-k = n-k \mod n$, which gives only two non-empty subarrays. If $p_1 = n$, P is split after the occurrence of $n-k-1$, which implies that C is empty since $p_n = n-k$. Hence, T can be constructed with a binary alphabet in those cases, *cf.* Fig. 2.

For the case $p_1 = k+1$, P is split after the occurrences of the values $n-k$ and $k+1-k-1 \mod n = n \mod n$, so B contains only the text position n. By construction, the requirement is that the suffix $\mathsf{T}[n]$ is smaller than all other suffixes starting with $\mathtt{c}$. So instead of assigning the unique character $\mathsf{T}[n] \leftarrow \mathtt{b}$ like in Theorem 3.2, we can assign $\mathsf{T}[n] \leftarrow \mathtt{c}$, which still makes $\mathsf{T}[n]$ the smallest suffix starting with $\mathtt{c}$. We conclude this case by converting the binary alphabet $\{\mathtt{a}, \mathtt{c}\}$ to $\{\mathtt{a}, \mathtt{b}\}$. *Cf.* Fig. 2, where T in Rotation (6) has become $\mathtt{bbabbabb}$ with period $n-k=3$. □

The main result of this section is the following theorem. There, we characterize all binary strings whose suffix arrays are arithmetically progressed permutations. More precisely, we identify which of them are unique,[11] periodic, or a Lyndon word.

[11] The exact number of these binary strings is not covered by Theorem 3.6.

Case	T								SA								p_s	s
	1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8		
(1)	b	a	b	b	a	b	b	a	[8,	5,	2|	7,	4,	1,	6,	3]	2	3
(2)	b	b	a	b	b	a	b	b	[6,	3|	8,	5,	2,	7,	4,	1]		3	2
(3)	a	b	a	b	b	a	b	b	[1,	6,	3|	8,	5,	2,	7,	4]		3	3

Fig. 6. All binary strings of length 8 whose suffix arrays are arithmetically progressed permutations with ratio $k = 5$. Theorem 3.9 characterizes these strings (and also gives the definition of p_s). Cases (1) and (3) also appear in Fig. 2 at Rotation (8) and (5), respectively, while Case (2) can be obtained from Rotation (6) by exchanging the last character with c. Cases (1) and (2) both have period $n - k = 3$, and Case (3) is a Lyndon word.

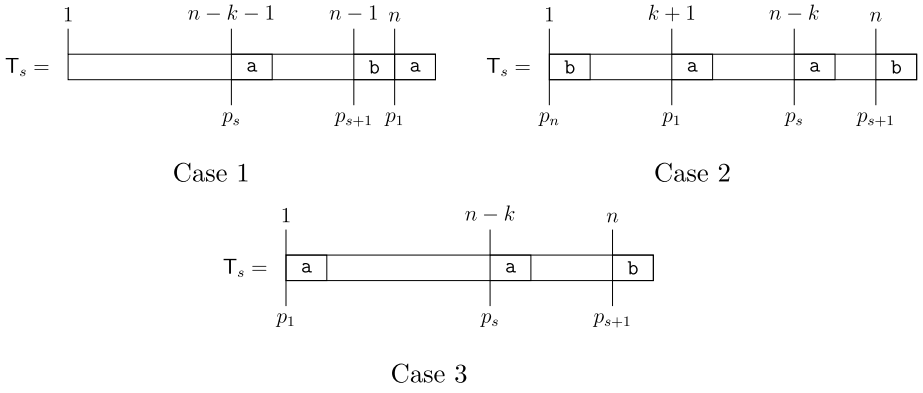

Fig. 7. Sketches of the cases of Theorem 3.9. T_s is uniquely determined if the suffix array SA of T_s is arithmetically progressed with ratio k and the first entry $\mathsf{SA}[1] \in \{1, k+1, n-k\}$ is given.

Theorem 3.9. *Let n and $k \in [1..n-1]$ be two coprime integers. If $k \neq n-1$, there are* exactly three *binary strings of length n whose suffix arrays are arithmetically progressed permutations with ratio k. Each such solution $\mathsf{T}_s \in \{\mathtt{a}, \mathtt{b}\}^+$ is characterized by*

$$\mathsf{T}_s[i] = \begin{cases} \mathtt{a} & \text{for } i \in \mathsf{SA}_{\mathsf{T}_s}[1..s], \text{ or} \\ \mathtt{b} & \text{otherwise,} \end{cases} \tag{2}$$

for all text positions $i \in [1..n]$ and an index $s \in [1..n-1]$, called the split index.

The individual solutions are obtained by fixing the values for p_1 and p_s, the position of the lexicographically largest suffix starting with a*, of $\mathsf{SA}_{\mathsf{T}_s} = [p_1, \ldots, p_n]$:*

1. *$p_1 = n$ and $p_s = n - k - 1$,*
2. *$p_1 = k + 1$ and $p_s = n - k$, and*
3. *$p_1 = 1$ and $p_s = n - k$.*

The string T_s has period $n - k$ in Cases 1 and 2, while T_s of Case 3 is a Lyndon word, which is not periodic by definition.

Proof. Let S be a binary string of length n, and suppose that $\mathsf{SA}_\mathsf{S} = P := [p_1, \ldots, p_n]$ is an arithmetically progressed permutation with ratio k. Further let p_s be the position of the largest suffix of S starting with a. Then $\mathsf{S}[p_i..n] \prec \mathsf{S}[p_{i+1}..n]$ and thus $\mathsf{S}[p_i] \leq \mathsf{S}[p_{i+1}]$. We have $\mathsf{S}[j] = \mathsf{S}[j + k \mod n]$ for all $j \in [1..n] \setminus \{p_n, p_s\}$ since

- $p_n = \mathsf{SA}_\mathsf{S}[n]$ is the starting position of the largest suffix ($\mathsf{S}[p_n] = \mathtt{b} \neq \mathtt{a} = \mathsf{S}[p_1] = \mathsf{S}[p_n + k]$).
- $\mathsf{S}[p_s..n]$ and $\mathsf{S}[p_{s+1}..n]$ are the lexicographically largest suffix starting with a and the lexicographically smallest suffix starting with b, respectively, such that $\mathsf{S}[p_s] = \mathtt{a} \neq \mathtt{b} = \mathsf{S}[p_{s+1}]$.

To sum up, since $\mathsf{S}[p_s..n] \prec \mathsf{S}[p_{s+1}..n]$ by construction, $\mathsf{S}[p_i..n] \prec \mathsf{S}[p_{i+1}..n]$ holds for $p_i > p_{i+1}$ whenever $p_i \neq p_n$. This, together with the coprimality of n and k, determines p_s uniquely in the three cases (*cf.* Fig. 6 for the case that $n = 8$ and $k = 5$, and Fig. 7 for sketches of the proof):

Case $p_1 = n$: We first observe that the case $k = n - 1$ gives us $P = [n, n-1, \ldots, 1]$, and this case was already treated with Theorem 2.1. In the following, we assume $k < n - 1$, and under this assumption we have $s > 1$,

$\mathsf{T}_s[n] = \mathsf{T}_s[p_1] = \mathtt{a}$ and $\mathsf{T}_s[n-1] = \mathtt{b}$ (otherwise $\mathsf{T}_s[n-1..n]$ would be the second smallest suffix, i.e., $P[2] = n-1$ and hence $k = n-1$). Consequently, $\mathsf{T}_s[n-1..n] = \mathsf{T}_s[p_{s+1}..n]$ is the smallest suffix starting with $\mathtt{b}$, namely $\mathtt{ba}$, and therefore $p_s = n-1-k$.

Case $p_1 \neq n$: If $p_1 \neq n$, then $\mathsf{T}_s[n] = \mathtt{b}$ (otherwise $\mathsf{T}_s[n] \prec \mathsf{T}_s[p_1..n]$). Therefore, $\mathsf{T}_s[n]$ is the smallest suffix starting with $\mathtt{b}$, and consequently $p_s = n-k$.

For the periodicity, with $\mathsf{T}_s[j] = \mathsf{T}_s[j+k \mod n] = \mathsf{T}_s[j-(n-k) \mod n]$ for $j \in [1..n] \setminus \{p_1, p_s\}$ we need to check two conditions:

- If $p_n - (n-k) > 0$, then $\mathsf{T}_s[p_n-(n-k)] = \mathsf{T}_s[p_1] \neq \mathsf{T}_s[p_n]$ breaks the periodicity.
- If $p_s - (n-k) > 0$, then $\mathsf{T}_s[p_s-(n-k)] = \mathtt{a} \neq \mathtt{b} = \mathsf{T}_s[p_{s+1}]$ breaks the periodicity.

For Case 1, $p_n = n-k$ and $p_s = n-k-1$ (hence $p_n-(n-k) = 0$ and $p_s-(n-k) = -1$), thus Case 1 is periodic.

Case 2 is analogous to Case 1.

For Case 3, T_s does not have period $n-k$ as $p_n = n-k+1$, and hence $p_n-(n-k) > 0$. It cannot have any other period since Case 3 yields a Lyndon word (because the lexicographically smallest suffix $\mathsf{T}_s[p_1..n] = \mathsf{T}_s[1..n]$ starts at the first text position). Note that Case 3 can be obtained from Case 2 by setting $\mathsf{T}_s[1] \leftarrow \mathtt{a}$ (the smallest suffix $\mathsf{T}_s[k+1..n]$ thus becomes the second smallest suffix).

Finally, we need to show that no other value for p_1 admits a binary string S having an arithmetically progressed permutation $P := [p_1, \ldots, p_n]$ with ratio k as its suffix array. So suppose that $p_1 \notin \{1, k+1, n\}$, then this would imply the following:

- $\mathsf{S}[p_1] = \mathtt{a}$ because the smallest suffix starts at text position p_1, and
- $\mathsf{S}[p_1-1] = \mathtt{b}$ because of the following: First, the text position $\mathsf{S}[p_1-1]$ exists due to $p_1 > 1$. Second, since $p_1 < n$, there is a text position $j \in [p_1+1..n]$ such that $\mathsf{S}[p_1] = \cdots = \mathsf{S}[j-1] = \mathtt{a}$ and $\mathsf{S}[j] = \mathtt{b}$ (otherwise $\mathsf{S}[n]$ would be the smallest suffix). If $\mathsf{S}[p_1-1] = \mathtt{a}$, then the suffix $\mathsf{S}[p_1-1..n]$ starting with $\mathtt{a}^{j-p_1+1}\mathtt{b}$ is lexicographically smaller than the suffix $\mathsf{S}[p_1..n]$ starting with $\mathtt{a}^{j-p_1}\mathtt{b}$. Hence, $\mathsf{S}[p_1-1] = \mathtt{b}$ must hold.
- $p_n - 1 \geq 1$ (since $p_1 \neq k+1$) and $\mathsf{S}[p_n-1] = \mathtt{a}$. If $\mathsf{S}[p_n-1] = \mathtt{b}$, then the suffix $\mathsf{S}[p_n-1..n]$ has a longer prefix of $\mathtt{b}$'s than the suffix $\mathsf{S}[p_n..n]$, and is therefore lexicographically larger.

Since $\mathsf{S}[p_n-1] = \mathtt{a}$ and $\mathsf{S}[p_1-1] = \mathtt{b}$ with $p_n - 1 + k \mod n = p_1 - 1$, the smallest suffix starting with $\mathtt{b}$ is located at index p_1-1. This is a contradiction as $p_1 \neq n$ implies $\mathsf{S}[n] = \mathtt{b}$ (if $\mathsf{S}[n] = \mathtt{a}$, then $\mathsf{SA}[1] = n$ instead of $\mathsf{SA}[1] = p_1$) and thus the smallest suffix starting with $\mathtt{b}$ is located at index n (this is a contradiction since we assumed that this suffix starts at $p_1 - 1 \in [1..n-1]$). This establishes the claim for p_1. □

For a given arithmetically progressed permutation with ratio k, and first entry $p_1 \in \{1, k+1, n\}$, the string T_s of Theorem 3.9 coincides with T of Theorem 3.8.

3.4. Inverse permutations

Since the inverse P^{-1} of a permutation P with $P^{-1}[P[i]] = i$ is also a permutation, one may wonder whether the inverse P^{-1} of an arithmetically progressed permutation is also arithmetically progressed. We affirm this question in the following. For that, we use the notion of the *multiplicative inverse* k^{-1} of an integer k (to the congruence class $[1..n] = \mathbb{Z}/n\mathbb{Z}$), which is given by $k^{-1} \cdot k \mod n = 1 \mod n$. The multiplicative inverse k^{-1} is uniquely defined if k and n are coprime.

In what follows, we show that the inverse of an arithmetically progressed permutation P with ratio k is also arithmetically progressed with ratio k^{-1}. As an example consider Fig. 2, where $n = 8$ and $k = 5$. The multiplicative inverse of k is given by $k^{-1} = 5 = k$. Hence, the set of arithmetically progressed permutations with the same parameters $n = 8$ and $k = 5$ contains also all its inverse permutations. Speaking of Fig. 2, the suffix arrays of (1) and (5) are each self-inverse. The suffix arrays of (2) and (4), (3) and (7), as well as (6) and (8) are inverse pairs, meaning that one is the inverse of the other.

Theorem 3.10. *The inverse P^{-1} of an arithmetically progressed permutation P with ratio k is an arithmetically progressed permutation with ratio k^{-1} and $P^{-1}[1] = (1-P[n]) \cdot k^{-1} \bmod n$.*

Proof. Let $x := P[i]$ for an index $i \in [1..n]$. Then $P[i+k^{-1} \bmod n] = P[i] + k \cdot k^{-1} \bmod n = x + k \cdot k^{-1} \bmod n = x+1 \bmod n$. For the inverse permutation P^{-1} this means that $P^{-1}[x] = i$ and $P^{-1}[x+1 \bmod n] = i + k^{-1} \bmod n$. Thus the difference between two consecutive entries of P^{-1}, i.e., $P^{-1}[x+1 \bmod n] - P^{-1}[x]$, is k^{-1}.

Since $P[i] = j \iff P[n] + ik \mod n = j$ holds for all indices $i \in [1..n]$, we have (using $i \leftarrow P^{-1}[1]$ and $j \leftarrow 1$ in the above equivalence)

$$\begin{aligned} P[P^{-1}[1]] = 1 \mod n &\iff P[n] + P^{-1}[1] \cdot k = 1 \mod n \\ &\iff P^{-1}[1] \cdot k = 1 - P[n] \mod n \\ &\iff P^{-1}[1] = (1-P[n]) \cdot k^{-1} \mod n. \quad \square \end{aligned}$$

Consequently, using the split index s of p_s for SA and

$$\begin{aligned}\mathsf{ISA}[i] &= \mathsf{ISA}[1] + (i-1)k^{-1} \mod n \\ &= (1-\mathsf{SA}[n])\cdot k^{-1} + (i-1)k^{-1} \mod n \\ &= (i-\mathsf{SA}[n])\cdot k^{-1} \mod n,\end{aligned}$$

we can rewrite T_s defined in Eq. (2) as

$$\mathsf{T}_s[i] = \begin{cases} \mathtt{a} & \text{if } \mathsf{ISA}[i] \le s, \text{ or} \\ \mathtt{b} & \text{otherwise} \end{cases} \tag{3}$$

where SA and ISA denote the suffix array and the inverse suffix array of T_s, respectively. Another result is that $\mathsf{ISA}[p_s] = s$ is the number of a's in T_s, for which we split the study into the cases of Theorem 3.9:

1. If $\mathsf{SA}[1] = n$ and $p_s = n-k-1$, then $\mathsf{SA}[n] = n-k$ and $\mathsf{ISA}[i] = (i-n+k)k^{-1} \mod n$. Consequently, $s = \mathsf{ISA}[p_s] = (-1)k^{-1} \mod n = n-k^{-1} \mod n$.
2. If $\mathsf{SA}[1] = k+1$ and $p_s = n-k$, then $\mathsf{SA}[n] = 1$ and $\mathsf{ISA}[i] = (i-1)k^{-1} \mod n$. Consequently, $s = \mathsf{ISA}[p_s] = (n-k-1)k^{-1} \mod n = nk^{-1} - 1 - k^{-1} \mod n = n-1-k^{-1} \mod n$.
3. If $\mathsf{SA}[1] = 1$ and $p_s = n-k$, then $\mathsf{SA}[n] = n-k+1$ and $\mathsf{ISA}[i] = (i-n+k-1)k^{-1} \mod n$. Consequently, $s = \mathsf{ISA}[p_s] = (-1)k^{-1} \mod n = n-k^{-1} \mod n$ as in Case (1).

For Fig. 6 with $k=5$ and $n=8$, we know that the number of a's is $\mathsf{ISA}[p_s] = 3$ in Cases (1) and (3), and $\mathsf{ISA}[p_s] = 2$ in Case (2) because $k^{-1} = 5 \Leftrightarrow k\cdot k^{-1} \mod n = 1 \mod n$. This also determines the constant m used in Lemma 3.7. Finally, we can fix the parameter t in Lemma 3.4 defined such that $p_t = p_1 - 1 - k \mod n$: For that, write $\mathsf{ISA}[i] = (i-p_n)k^{-1} \mod n = (i+k-p_1)k^{-1} \mod n$ and compute $\mathsf{ISA}[p_t] = \mathsf{ISA}[p_1 - 1 - k] = (-1)k^{-1} \mod n = n-k^{-1} \mod n$.

4. Applications

We conclude our main results of Section 3 by drawing connections between strings having arithmetically progressed suffix arrays and Christoffel words (Section 4.1), balanced words (Section 4.2), and Fibonacci words (Section 4.3).

4.1. Christoffel words

Christoffel words are binary strings whose origins are considered to date from Bernoulli's 1771 work [28]. Christoffel words can be described geometrically in terms of a line segment and associated polygon traversal [29]: let $(p,q) \in \mathbb{N}^2$ where (p,q) are coprime and let $\mathcal{S}$ be the line segment with endpoints $(0,0)$ and (p,q). The *induced path* of a binary string T is a list of points $v_0, \ldots, v_n \in \mathbb{N}^2$ such that $v_0 = (0,0)$, $v_n = (p,q)$, and for each $i \in [1..n]$, $v_i - v_{i-1} = (1,0)$ if $\mathsf{T}[i] = \mathtt{a}$ and $v_i - v_{i-1} = (0,1)$ if $\mathsf{T}[i] = \mathtt{b}$. The string $\mathsf{T} \in \{\mathtt{a},\mathtt{b}\}^*$ is a *lower Christoffel* word if the path induced by T from the origin to (p,q) is below the line segment $\mathcal{S}$ and the path and $\mathcal{S}$ determines a simple polygon[12] which contains no other point in $\mathbb{N}^2$. An *upper Christoffel* word is defined analogously by taking the path above $\mathcal{S}$. Hence, a Christoffel word is defined by a direction (above or below) and the *slope* $\frac{q}{p}$, which determines p and q since p and q are coprime. We illustrate the upper and lower Christoffel words for the slope $\frac{4}{7}$ in Fig. 8. Furthermore, Case (3) of Fig. 6 determines a Christoffel word with slope $\frac{5}{3}$ and Fig. 11 gives an example of the geometric description where $p=7$ and $q=5$.

It follows from the coprimality of p and q that Christoffel words are necessarily primitive. In what follows, we focus on lower Christoffel words, and may drop the adjective *lower* when it is clear we are speaking of (lower) Christoffel words.

To show that every lower Christoffel word has an arithmetically progressed suffix array, we use an alternative characterization of Christoffel words based on Cayley graphs [30, Def. 1.4]. The Cayley graph (V,E) of a group G with generating set Σ is defined by $V = G$ and $E = \{(g, ga) \mid g \in G, a \in \Sigma\}$. Let again $p,q \in \mathbb{N}$ be coprime. Fig. 9 is the Cayley graph of $\mathbb{Z}/(p+q)\mathbb{Z}$ generated by q. Cayley graphs are always simple cycles since q and $p+q$ are coprime. In what follows, we use mod_0 with $n \bmod_0 n = 0 \bmod_0 n = 0$ to match the definition in [30], as opposed to mod with $0 \bmod n = n \bmod n = n$ elsewhere.

An edge $s \to t$ in the Cayley graph has the label $\mathtt{a}$ if $s < t$, otherwise it has the label $\mathtt{b}$. Reading the edge labels, starting from node 0, following the edges of the Cayley graph and stopping when reaching node 0 again, yields the lower Christoffel word T_c parameterized by p and q. The ith node in the Cayley graph (0 being the first node) is $(i-1)q \bmod_0 (p+q)$. Hence the ith character of T_c is

$$\mathsf{T}_c[i] = \begin{cases} \mathtt{a} & \text{if } (i-1)q \bmod_0 (p+q) < iq \bmod_0 (p+q), \\ \mathtt{b} & \text{if } (i-1)q \bmod_0 (p+q) > iq \bmod_0 (p+q). \end{cases} \tag{4}$$

Given the suffix array $\mathsf{SA}_{\mathsf{T}_c}$ of a lower Christoffel word T_c, the split index s (defined in Theorem 3.9) is given by p, the total number of units along the x-axis in the polygonal path. All lower Christoffel words are Lyndon words [29] and so necessarily border-free. Hence $\mathsf{SA}_{\mathsf{T}_c}[p_1] = 1$, and $\mathsf{SA}_{\mathsf{T}_c}[s+1] = n$ since the string T_c must end $\mathtt{b}$. The following theorem now gives the connection between lower Christoffel words and the strings of Section 3:

[12] Note that this polygon is unique and the line segment $\mathcal{S}$ contains no point in $\mathbb{N}^2$ as p and q are coprime.

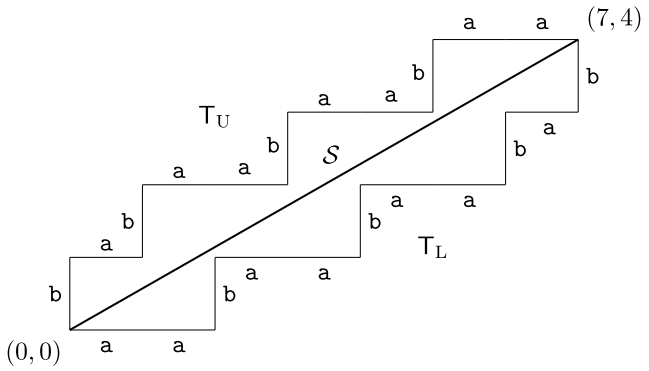

Fig. 8. An upper Christoffel word $\mathsf{T}_\mathrm{U} = \mathtt{babaabaabaa}$ and its reversal, the lower Christoffel word $\mathsf{T}_\mathrm{L} = \mathtt{aabaabaabab}$, shown geometrically with respect to the line segment $\mathcal{S} = ((0, 0), (7, 4))$.

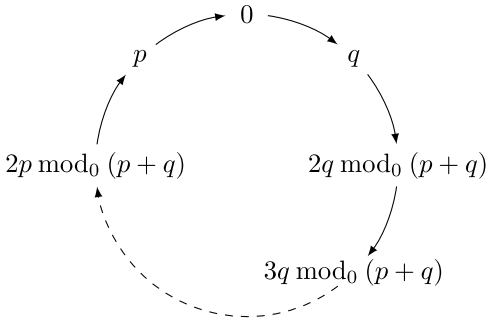

Fig. 9. Cayley graph of $\mathbb{Z}/(p+q)\mathbb{Z}$ generated by q. See [30, Fig. 1.4] for a concrete example.

Theorem 4.1. *Let $p, q \in \mathbb{N}$ be coprime. Then the lower Christoffel word T_c characterized by p and q has an arithmetically progressed suffix array. The suffix array is given by the arithmetic progression P with $p_1 = 1$ and $k = q^{-1}$, where q^{-1} is the multiplicative inverse of q in $\mathbb{Z}/n\mathbb{Z}$. The string T_c is identical to Case 3 of Theorem 3.9 characterizing the binary case.*

Proof. We prove the theorem by showing that the Christoffel word T_c is equal to the string described in Theorem 3.9 as Case 3 whose suffix array is the arithmetically progressed array P.

Let $n = p + q$. By Eq. (4) the ith character of T_c is an $\mathtt{a}$ if and only if $(i-1)q \bmod_0 n \;<\; iq \bmod_0 n$. We can rewrite that to $(i-1)q \bmod_0 n \;<\; (i-1)q \bmod_0 n + q \bmod_0 n$. This condition is fulfilled if and only if $((i-1)q \bmod_0 n) + q < n$. We can rewrite that to $(i-1)q \bmod_0 n < n - q$. Replacing $\bmod_0$ with mod, we obtain

$$\mathsf{T}_\mathrm{c}[i] = \begin{cases} \mathtt{a} & \text{if } \; 1 + (i-1)q \bmod n \;<\; n + 1 - q, \\ \mathtt{b} & \text{otherwise.} \end{cases}$$

Let $k = q^{-1}$. Let T_s denote the string of Case 3 in Theorem 3.9, which is characterized by $p_1 = 1$ and $p_s = n - k = n - q^{-1}$. Using the results from Section 3.4, Eq. (3), T_s can be written as

$$\mathsf{T}_s[i] = \begin{cases} \mathtt{a} & \text{if } \; (i + k - 1)k^{-1} \bmod n \;\le\; s, \\ \mathtt{b} & \text{otherwise.} \end{cases} \tag{5}$$

Substituting $k = q^{-1}$, $k^{-1} = q$ and $s = \mathsf{ISA}[p_s] = \mathsf{ISA}[n - q^{-1}] = n - q$, we can rewrite the condition for $\mathtt{a}$ in Eq. (5) as $1 + (i-1)q \bmod n \le n - q$. Replacing $\le$ with $<$ we obtain the same definition for T_s as for T_c, concluding the proof.

$$\mathsf{T}_s[i] = \begin{cases} \mathtt{a} & \text{if } \; 1 + (i-1)q \bmod n \;<\; n + 1 - q, \\ \mathtt{b} & \text{otherwise.} \end{cases} \quad \square$$

In the geometric setting, traversing the path of the polygon from $(0, 0)$ to (p, q) is equivalent to scanning the characters in the defining Christoffel word T_c; hence, we can associate each character with its polygon coordinates. The BWT can be obtained from iterating across the suffix array and for each index selecting the preceding character in the text — the BWT for the first string in Fig. 10 has maximal runs in the form $\mathtt{b}^5\mathtt{a}^7$; more generally, the BWT of a Christoffel word over $\Sigma = \{\mathtt{a}, \mathtt{b}\}$ with slope $\frac{q}{p}$ takes the form $\mathtt{b}^q\mathtt{a}^p$ [30].

The progression ratio q^{-1} of Theorem 4.1 may be useful for accessing geometries of interest in the polygon or aiding discovery of geometric repetitive structures. For instance, to access the 'steps' in the polygon in decreasing width, start at

	T	SA	p_s	s
	1 2 3 4 5 6 7 8 9 10 11 12	1 2 3 4 5 6 7 8 9 10 11 12		
(1)	a a b a b a a b a b a b	[1, 6, 11, 4, 9, 2, 7, 12, 5, 10, 3, 8]	7	7
(2)	b a b a b a a b a b a a	[1, 8, 3, 10, 5, 12, 7, 2, 9, 4, 11, 6]	5	5

Fig. 10. Two Christoffel words with $S = ((0,0),(7,5))$, slope $\frac{5}{7}$, and length 12. The suffix array of the lower Christoffel word (1) has an arithmetically progressed permutation with ratio $k = 5$, $p_s = 7 = n - k$, and hence is an instance of Theorem 3.9, Case 3. Then, assuming b < a, the suffix array of the upper Christoffel word (2) has an arithmetic progression ratio of $n - k = 7$ with split index $n - s$ and position of largest suffix starting b of $n - p_s$.

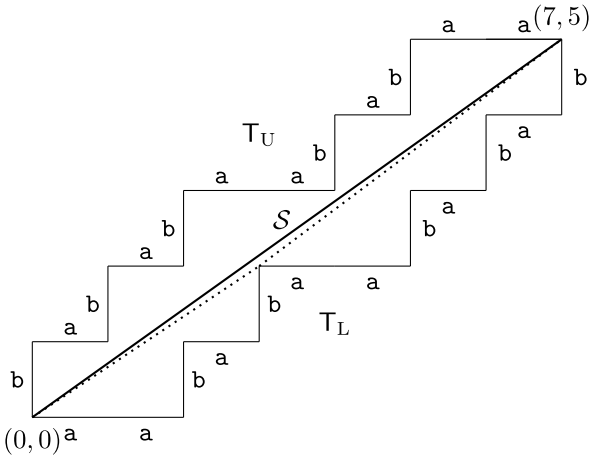

Fig. 11. An upper Christoffel word $\mathsf{T}_\mathrm{U} = \mathtt{bababaababaa}$ and its reversal, the lower Christoffel word $\mathsf{T}_\mathrm{L} = \mathtt{aababaababab}$, shown geometrically with respect to the line segment $\mathcal{S} = ((0,0),(7,5))$. The dotted lines indicate the balanced$_2$ factorization of T_L into the factors aabab and aababab.

the origin and apply increments of k: for the first string in Fig. 10 the widest steps are at coordinates $(0,0)$ and $(3,2)$ — see Fig. 11. Note that the geometric width of this polygon at a certain height (that is, a step) reflects a run of the character a in its representing Christoffel word T_c.

We now extend a known result for lower Christoffel words [31, Corollary 5.1], [30, Sect. 6.1], which distinguishes consecutive positions in consecutive rows in an associated BWT matrix:

Lemma 4.2. *Let T_c be a Christoffel word of length $n \geq 2$ over $\Sigma = \{\mathtt{a}, \mathtt{b}\}$. Suppose T_c has an arithmetically progressed suffix array with progression ratio k. Then in the BWT matrix $\mathbf{M}$ of T_c with $\mathbf{M}[i,j] = \mathsf{T}_\mathrm{c}[(\mathsf{SA}_{\mathsf{T}_\mathrm{c}}[i] + j - 1) \mod n]$, rows i and $i+1$ for $1 \leq i < n$ differ in exactly two consecutive positions. For each row i, these positions are at $\mathbf{M}[i, i(n-k) \bmod n]$ and $\mathbf{M}[i, i(n-k)+1 \bmod n]$.*

Proof. The Christoffel word T_c is characterized by Theorem 3.9 (3). Further, since T_c is also a Lyndon word then it starts with a and ends with b. The second row in the BWT matrix $\mathbf{M}$ of T_c is $\mathsf{T}_\mathrm{c}[\mathsf{SA}_{\mathsf{T}_\mathrm{c}}[2]..n]\mathsf{T}_\mathrm{c}[1..\mathsf{SA}_{\mathsf{T}_\mathrm{c}}[2]-1]$ where the prefix $\mathsf{T}_\mathrm{c}[\mathsf{SA}_{\mathsf{T}_\mathrm{c}}[2]..n]$ has length $n-k$ and ends with b.

From Case 3 of Theorem 3.9, $p_s = n - k$, and so $\mathsf{T}_\mathrm{c}[n-k] = \mathtt{a}$, and since $\mathsf{T}_\mathrm{c}[n-k..n]$ is the largest suffix starting with a and $n \geq 2$, then it has the prefix ab. So $\mathbf{M}[1, n-k] = \mathtt{a}$ and $\mathbf{M}[1, n-k+1] = \mathtt{b}$, while from $\mathsf{T}_\mathrm{c}[1] = \mathtt{a}$ and $\mathsf{T}_\mathrm{c}[n] = \mathtt{b}$, $\mathbf{M}[2, n-k] = \mathtt{b}$ and $\mathbf{M}[2, n-k+1] = \mathtt{a}$.

This gives an arrangement, denoted by $\mathcal{A}$, across two adjacent rows i and $i+1$ and depicted as $\mathcal{A} = \frac{\mathtt{ab}}{\mathtt{ba}}$, where at some index h, $\mathbf{M}(i,h) = \mathtt{a}$, $\mathbf{M}(i,h+1) = \mathtt{b}$, $\mathbf{M}(i+1,h) = \mathtt{b}$, $\mathbf{M}(i+1,h+1) = \mathtt{a}$. We will map the locations of $\mathcal{A}$ in the matrix $\mathbf{M}$ which for the first two rows start at $\mathbf{M}(1, n-k)$ and $\mathbf{M}(2, 2(n-k) \bmod n)$. Similarly for each subsequent row i in $\mathbf{M}$, $i < n$, there is an additional factor of $(n-k) \bmod n$ for the position, where $\mathcal{A}$ occurs.

We proceed to show that there are no other differences between adjacent rows other than those occurring in $\mathcal{A}$. So suppose instead that $\mathbf{M}[i,j] = \mathtt{c}$ and $\mathbf{M}[i+1,j] = \mathtt{d}$ for two distinct characters $\mathtt{c}, \mathtt{d} \in \Sigma$, $j \neq i(n-k) \bmod n$ and $j \neq i(n-k)+1 \bmod n$. Then the character d in row $i+1$ is the same character in T_c as the one at position $j-(n-k) \bmod n$ in row i. Consider the character c at position $j-(n-k)-k \bmod n$ in row i. If c is a b then d cannot be an a. If c is an a and d is a b then this must occur at $\mathcal{A}$, that is $i = \mathsf{SA}^{-1}[p_s]$ and $j = 1$, thereby contradicting the conditions on j. So if c is an a then d cannot be a b.

Hence the only differences between adjacent rows in **M** occur at $\mathcal{A}$ which coincides with $\mathsf{T_c}[p_s]$ and $\mathsf{T_c}[n]$ and their adjacent letters in the rows. In particular, $\mathcal{A}$ determines the lexicographic ordering of the rows of the matrix **M**. □

The arithmetic progression of arithmetically progressed suffix arrays of Christoffel words allows us to determine the following factorizations of such words in constant time (without looking at the explicit characters of the word):

- *Right factorization* [30]; originally called the *standard factorization* [32,33]. If $\mathsf{w} = \mathsf{uv}$ is a Lyndon word with v its lexicographically least proper suffix, then u and v are also Lyndon words and $\mathsf{u} < \mathsf{v}$. Equivalently, the right factor v of the standard factorization can be defined to be the longest proper suffix of w that is a Lyndon word [34].
- *Left factorization* [35,36]. If $\mathsf{w} = \mathsf{uv}$ is a Lyndon word with u a proper Lyndon prefix of maximal length, then v is also a Lyndon word and $\mathsf{u} < \mathsf{v}$.
- *balanced$_2$ factorization* [37]. A Lyndon word w is balanced$_2$ if w is a character or there is a factorization $\mathsf{w} = (\mathsf{u}, \mathsf{v})$ that is simultaneously the left and the right factorization of w, and u and v are also balanced$_2$.

For any of the three above factorizations $\mathsf{w} = (\mathsf{u}, \mathsf{v})$, we will say that the *factorization index* is the index of the last character of u in w. The factorization index determines the split position and hence the factorization.

Lemma 4.3. *Let $\mathsf{T_c}$ be a Christoffel word of length $n \geq 3$ over $\Sigma = \{\mathtt{a}, \mathtt{b}\}$. Suppose $\mathsf{T_c}$ has an arithmetically progressed suffix array $\mathsf{SA}_{\mathsf{T_c}}$ with progression ratio $k \geq 1$ and split index s. Then $\mathsf{T_c}$ is a balanced$_2$ Lyndon word with a factorization $\mathsf{T_c} = \mathsf{T_u}\mathsf{T_v}$ such that $|\mathsf{T_u}| = \mathsf{SA}_{\mathsf{T_c}}[(s+2) \bmod n]$. For $n = 1, 2$ we have that $\mathsf{T_c} \in \{\mathtt{a}, \mathtt{b}, \mathtt{ab}\}$, $k = 1$, and that the split index s is 1.*

Proof. We apply the result that a word is a lower Christoffel word if and only if it is a balanced$_2$ Lyndon word [37] or [30, Theorem 6.7]. So given $\mathsf{T_c}$, it remains to prove that $|\mathsf{T_u}| = \mathsf{SA}_{\mathsf{T_c}}[(s+2) \bmod n]$. Let $j \in [1..n]$ be such that $|\mathsf{T_u}| = \mathsf{SA}_{\mathsf{T_c}}[j]$. Then $\mathsf{T_u} = \mathsf{T_c}[1..\mathsf{SA}_{\mathsf{T_c}}[j]]$ and $\mathsf{T_v} = \mathsf{T_c}[\mathsf{SA}_{\mathsf{T_c}}[j]+1..n]$. Since $\mathsf{T_u}$ and $\mathsf{T_v}$ are Lyndon words, we have that

- $j > s$, otherwise $\mathsf{T_c}[\mathsf{SA}_{\mathsf{T_c}}[j]] = \mathtt{a}$ and $\mathsf{T_u}$ would have a border.
- $\mathsf{SA}_{\mathsf{T_c}}[(s+1) \bmod n] = n$ with $\mathsf{T_c}[n] = \mathtt{b}$ since $\mathsf{T_c}$ is a Lyndon word larger than one, and the smallest suffix starting with $\mathtt{b}$ is $\mathsf{T_c}[n..n]$.
- The second smallest suffix starting with $\mathtt{b}$ is $\mathsf{T_c}[\mathsf{SA}_{\mathsf{T_c}}[2]-1..n]$. To see that observe that, because $\mathsf{T_c}$ is a Lyndon word, $\mathsf{SA}_{\mathsf{T_c}}[1] = 1$, and hence $\mathsf{SA}_{\mathsf{T_c}}[2] \geq 2$. Next, $\mathsf{T_c}[\mathsf{SA}_{\mathsf{T_c}}[2]-1] = \mathtt{b}$, otherwise $\mathsf{T_c}[\mathsf{SA}_{\mathsf{T_c}}[2]-1..n] \prec \mathsf{T_c}[\mathsf{SA}_{\mathsf{T_c}}[2]..n]$ would give a smaller suffix. Hence, $\mathsf{SA}_{\mathsf{T_c}}[(s+2) \bmod n] = \mathsf{SA}_{\mathsf{T_c}}[2]-1$.

Finally, $\mathsf{T_v} = \mathsf{T_c}[\mathsf{SA}_{\mathsf{T_c}}[2]..n]$ since otherwise we yield a contradiction to the right factorization that $\mathsf{T_v}$ is the lexicographically least proper suffix. So $j = \mathsf{SA}_{\mathsf{T_c}}[(s+2) \bmod n]$. □

For example, the Christoffel word in Fig. 10 Case (1) has factorization index $\mathsf{SA}_{\mathsf{T_c}}[9] = 5$ with the balanced$_2$ factorization $(\mathtt{aabab})(\mathtt{aababab})$; the first level of recursion gives the balanced$_2$ factorizations $(\mathtt{aab})(\mathtt{ab})$ and $(\mathtt{aabab})(\mathtt{ab})$.

Conditions for the factorization of a Lyndon word into exactly two non-overlapping Lyndon factors are given in [38], where *overlapping factors* have non-empty suffixes and prefixes sharing the same characters. We can geometrically consider the balanced$_2$ factorization of Christoffel words as follows:

Lemma 4.4. *Let $p, q \in \mathbb{N}$ be coprime and $\mathsf{T_c}$ be the lower Christoffel word characterized by p and q with factorization index i. Further, let the line segment $\mathcal{S} = ((0,0),(p,q))$ be associated with $\mathsf{T_c}$. Then for all points on the path determined by $\mathsf{T_c}$ (apart from the end points), $\mathsf{T_c}[i]$ has the shortest Euclidean distance to $\mathcal{S}$.*

Proof. Since $\mathsf{T_c}$ is a Christoffel word it is also a balanced$_2$ Lyndon word. Let the balanced$_2$ factorization be $\mathsf{T_c} = \mathsf{T_u}\mathsf{T_v}$, where the factors have defining line segments $\mathcal{S}_\mathsf{u}$, $\mathcal{S}_\mathsf{v}$ respectively, i.e., $\mathcal{S}_\mathsf{u}$ is the line from $(0,0)$ to the point s being the geometric representation of $\mathsf{T_c}[1..|\mathsf{T_u}|]$, and $\mathcal{S}_\mathsf{v}$ is the line from s to (p,q), cf. Fig. 11 with $\mathcal{S}_\mathsf{u} = \mathtt{aabab}$ and $s = (3,2)$. Now assume that there exists a point r that geometrically represents a prefix $\mathsf{T_c}[1..i]$ with the property that r does not have the shortest Euclidean distance to $\mathcal{S}$. Then at least one of the paths for $\mathsf{T_u}$ and $\mathsf{T_v}$ must cross their associated line $\mathcal{S}_\mathsf{u}$ or $\mathcal{S}_\mathsf{v}$, contradicting the geometric definition of a Christoffel word. □

We can also interpret the increasing sequences in the suffix array of a Christoffel word geometrically:

Lemma 4.5. *Let $p, q \in \mathbb{N}$ be coprime and $\mathsf{T_c}$ be the lower Christoffel word over $\Sigma = \{\mathtt{a}, \mathtt{b}\}$ characterized by p and q and with the associated line segment $\mathcal{S} = ((0,0),(p,q))$. Suppose $\mathsf{T_c}$ of length $n \geq 2$ has the arithmetically progressed suffix array $\mathsf{SA}_{\mathsf{T_c}}$ with progression ratio $k \geq 1$. Let $\mathsf{SA}_{\mathsf{T_c}} = A_1, A_2, \ldots, A_j$, $1 \leq j < n$, be the factorization of $\mathsf{SA}_{\mathsf{T_c}}$ into j maximally increasing sequences. Then,*
(i) for $A_i = a_1 < a_2 < \cdots < a_f$, $1 \leq i \leq j$, the geometry given by the coordinates of the characters indexed in $\mathsf{T_c}$ by A_i is a line (possibly degenerate with $f = 1$), and
(ii) the j lines in (i) are parallel, and
(iii) the line for A_i has shorter Euclidean distance to $\mathcal{S}$ than that for A_{i+1}, $1 \leq i < j$.
(For $n = 1$, $\mathsf{T_c} = \mathtt{a}$ and there is one degenerate line, a point. For $n = 2$, $\mathsf{T_c} = \mathtt{ab}$ and the unique line is $((0,0),(1,0))$.)

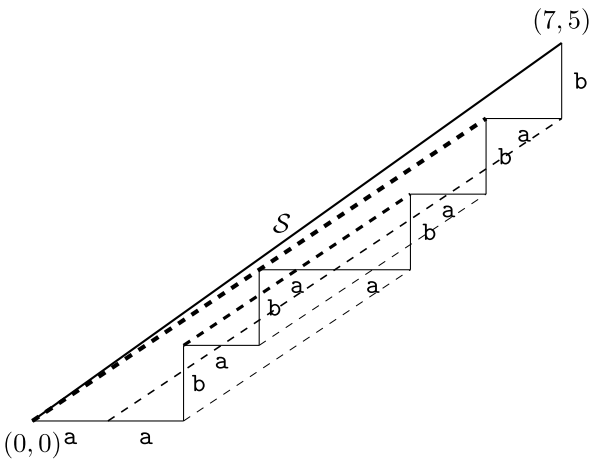

Fig. 12. The factorization of the suffix array [1, 6, 11, 4, 9, 2, 7, 12, 5, 10, 3, 8] of the lower Christoffel word $\mathsf{T}_{\mathrm{L}} = \mathtt{aababaababab}$ in Figs. 10 & 11 shown geometrically. The parallel lines associated with the increasing sequences in the suffix array, namely [1, 6, 11], [4, 9], [2, 7, 12], [5, 10], [3, 8], are depicted with decreasing thickness as they get further from the line $\mathcal{S}$.

Proof. Part (i). Consider the sequence $A_i = a_1, a_2, \ldots, a_f$ with $f > 1$. For $1 < d < f$, if $\mathsf{T}_{\mathrm{c}}[a_1] = \mathtt{a}$ then $\mathsf{T}_{\mathrm{c}}[a_d] = \mathtt{a}$, otherwise $\mathsf{T}_{\mathrm{c}}[a_1] = \mathsf{T}_{\mathrm{c}}[a_d] = \mathtt{b}$. Suppose there is a minimal $p > 0$ such that $\mathsf{T}_{\mathrm{c}}[a_1 + p] \neq \mathsf{T}_{\mathrm{c}}[a_d + p]$. If $\mathsf{T}_{\mathrm{c}}[a_1 + p] = \mathtt{a}$ and $\mathsf{T}_{\mathrm{c}}[a_d + p] = \mathtt{b}$ then the geometric path of $\mathsf{T}_{\mathrm{c}}[a_d..a_d + p]$ crosses $\mathcal{S}$; otherwise, $\mathsf{T}_{\mathrm{c}}[a_1 + p] = \mathtt{b}$ and $\mathsf{T}_{\mathrm{c}}[a_d + p] = \mathtt{a}$ and the region between $\mathsf{T}_{\mathrm{c}}[a_d..a_d + p]$ and $\mathcal{S}$ contains a point in $\mathbb{N}^2$ – either case contradicts the definition of a Christoffel word. Therefore the paths $\mathsf{T}_{\mathrm{c}}[a_1..a_1 + k - 1]$ and $\mathsf{T}_{\mathrm{c}}[a_d..a_d + k - 1]$ have the same shape and the same slope. For $d = f$, if $a_d \neq n$ then $\mathsf{T}_{\mathrm{c}}[a_d..n - 1]$ is a proper prefix of $\mathsf{T}_{\mathrm{c}}[a_1..a_1 + k - 1]$ hence with $\mathsf{T}_{\mathrm{c}}[a_d]$ on the slope, while if $a_d = n$ then $\mathsf{T}_{\mathrm{c}}[a_d]$ is connected to $\mathsf{T}_{\mathrm{c}}[a_d - 1]$, the suffix of the shape, and likewise is on the same slope.
Part (ii). Consider two maximal sequences of $\mathsf{SA}_{\mathsf{T}_{\mathrm{c}}}$, $A_i = a_1, a_2, \ldots, a_f$ and $A_{i+1} = a'_1, a'_2, \ldots, a'_g$ with $f, g > 1$. From part (i), A_i and A_{i+1} both describe lines which have the same slope, namely the relative number of b's to a's in $\mathsf{T}_{\mathrm{c}}[1..k - 1]$. Since the lines do not have any point in common they must be parallel.
Part (iii). Here, either of the above sequences $A_i = a_1, a_2, \ldots, a_f$ and $A_{i+1} = a'_1, a'_2, \ldots, a'_g$ may index a single character, that is $f, g \geq 1$. Since $a'_1 < a_f$, $\mathsf{T}_{\mathrm{c}}[a_f..n]$ is a proper suffix of $\mathsf{T}_{\mathrm{c}}[a'_1..n]$. Suppose that $a_f \neq n$. For $f \geq 1$, if $\mathsf{T}_{\mathrm{c}}[a'_1..a_f - 1]$ does not contain a b, then this would imply $\mathsf{ISA}_{\mathsf{T}_{\mathrm{c}}}[a'_1] < \mathsf{ISA}_{\mathsf{T}_{\mathrm{c}}}[a_f]$, thus showing that the point for $\mathsf{T}_{\mathrm{c}}[a'_1]$ is vertically further from $\mathcal{S}$ than that for $\mathsf{T}_{\mathrm{c}}[a_1]$. If $a_f = n$, then $f > 1$, and since a_f is the index of the smallest suffix starting b, $\mathsf{T}_{\mathrm{c}}[a_1..a_f] = \mathtt{a..b}$ while $\mathsf{T}_{\mathrm{c}}[a'_1..n] = \mathtt{b..b}$. Further, $a'_1 = k$ and so $a_1 < a'_1$. Since $\mathsf{T}_{\mathrm{c}}[a'_1] = \mathtt{b}$, the point given by $\mathsf{T}_{\mathrm{c}}[a'_1]$ is vertically further from $\mathcal{S}$ than the line given by the points $\mathsf{T}_{\mathrm{c}}[a_1]$ and $\mathsf{T}_{\mathrm{c}}[a_f]$. The rest follows from parts (i) & (ii). □

In a practical geometric application, Lemma 4.5 implies that for a maximal increasing sequence $A_i = a_1, a_2, \ldots, a_f$, the associated geometric path needs only be stored for $\mathsf{T}_{\mathrm{c}}[a_1..a_2 - 1]$ due to repetitions of the path structure. The lemma is illustrated in Fig. 12:

4.2. Balanced words

A binary string T is called a *balanced word* if for each character $c \in \{\mathtt{a}, \mathtt{b}\}$, the number of occurrences of c in U and in V differ by at most one, for all pairs of substrings U and V with $|\mathsf{U}| = |\mathsf{V}|$ of the infinite concatenation $\mathsf{T} \cdot \mathsf{T} \cdots$ of T.

Lemma 4.6. *Let T be a string over the binary alphabet $\{\mathtt{a}, \mathtt{b}\}$ with an arithmetically progressed suffix array $P = [p_1, \ldots, p_n]$ with ratio k. Given that $p_1 \neq k - 1$, T is balanced.*

Proof. According to Theorem 3.3 for T the BWT defined on the suffix array and the BWT defined on the BWT matrix are identical. From Corollary 3.5 we know that the BWT of T has the shape $\mathtt{b}^x\mathtt{a}^y$. By [39, Thm. 2] for binary words the following conditions are equivalent:

1. There are two coprime numbers p and q such that $\mathsf{BWT}_{\mathsf{T}} = \mathtt{b}^p\mathtt{a}^q$;
2. T is balanced.

We conclude the proof by showing that x and y are co-prime. Since $p_1 \neq k - 1$ we have $p_1 \in \{1, n\}$. Using the results from Section 3.4 the number of a's in T is $n - k^{-1}$. Thus the number of b's is k^{-1}. As k^{-1} is co-prime to n, $n - k^{-1}$ must be co-prime to k^{-1}. □

4.3. Relation to the Fibonacci word sequence

The Fibonacci sequence is a sequence of binary strings $\{\mathsf{F}_m\}_{m\geq 1}$ with $\mathsf{F}_1 := \mathtt{b}$, $\mathsf{F}_2 := \mathtt{a}$, and $\mathsf{F}_m := \mathsf{F}_{m-1}\mathsf{F}_{m-2}$. Then F_m and $f_m := |\mathsf{F}_m|$ are called the mth *Fibonacci word* and the mth *Fibonacci number*, respectively.

Köppl and I [11, Thm. 1] observed that the suffix array of F_m for even m is the arithmetically progressed permutation $\mathsf{SA}_{\mathsf{F}_m}$ with ratio $f_{m-2} \bmod f_m$ and $\mathsf{SA}_{\mathsf{F}_m}[1] = f_m$. Theorem 3.9 generalizes this observation by characterizing all binary strings whose suffix arrays are arithmetically progressed. Hence, F_m must coincide with Case 1 of Theorem 3.9 since it ends with character $\mathtt{a}$.

Lemma 4.7. *The Fibonacci word F_m for even m is given by*

$$\mathsf{F}_m[i] = \begin{cases} \mathtt{a} & \text{if } 1 + i \cdot f_{m-2} \bmod f_m \leq f_{m-1}, \text{ or} \\ \mathtt{b} & \text{otherwise.} \end{cases}$$

Proof. We use the following facts:

- The greatest common divisor of f_i and f_j is the Fibonacci number whose index is the greatest common divisor of i and j [40, Fibonacci numbers]. Hence, f_{m-1} and f_m are coprime for every $m \geq 2$.
- $f_{m-2}^2 \bmod f_m = 1$ holds for every even $m \geq 3$ [41]. Hence, $k^{-1} = k = f_{m-2}$.
- By definition, $\mathsf{F}_m[f_m] = \mathtt{a}$ if m is even, and therefore $\mathsf{SA}_{\mathsf{F}_m}[1] = f_m$.

The split position p_s is $p_s = f_m - k = f_{m-1}$. So $\mathsf{SA}_{\mathsf{F}_m}[f_m] = f_m - k = f_m - f_{m-2}$. By Theorem 3.10, $\mathsf{ISA}_{\mathsf{F}_m}[i] = if_{m-2} - \mathsf{SA}_{\mathsf{F}_m}[f_m]f_{m-2} \bmod f_m = if_{m-2} + 1 \bmod f_m$, where $-\mathsf{SA}_{\mathsf{F}_m}[f_m]f_{m-2} \bmod f_m = (f_{m-2} - f_m)f_{m-2} \bmod f_m = 1 - f_m f_{m-2} \bmod f_m = 1$. The rest follows from Eq. (3). □

Let $\bar{\mathsf{F}}_m$ denote the mth Fibonacci word whose $\mathtt{a}$'s and $\mathtt{b}$'s are exchanged, i.e., $\bar{\mathsf{F}}_m = \mathtt{a} \Leftrightarrow \mathsf{F}_m = \mathtt{b}$.

Lemma 4.8. $\mathsf{SA}_{\bar{\mathsf{F}}_m}$ *is arithmetically progressed with ratio f_{m-2} for odd m.*

Proof. Since $\bar{\mathsf{F}}_m[|\bar{\mathsf{F}}_m|] = \mathtt{a}$ for odd m, $\bar{\mathsf{F}}_m[f_m..]$ is the lexicographically smallest suffix. Hence, $\mathsf{SA} := \mathsf{SA}_{\bar{\mathsf{F}}_m} = |\bar{\mathsf{F}}_m|$. If SA is arithmetically progressed with ratio k, then its split position must be $p_s = n - k - 1$ according to Theorem 3.9. We show that $k = f_{m-2}$ by proving

$$\begin{aligned}\bar{\mathsf{F}}_m[f_m..] \prec \bar{\mathsf{F}}_m[f_m + f_{m-2} \bmod f_m..] &\prec \bar{\mathsf{F}}_m[f_m + 2f_{m-2} \bmod f_m..] \prec \dots \\ &\prec \bar{\mathsf{F}}_m[f_m + (f_m - 1)f_{m-2} \bmod f_m..]\end{aligned}$$

in a way similar to [11, Lemma 8]. For that, let $\bar{\mathsf{S}}$ of a binary string $\mathsf{S} \in \{\mathtt{a}, \mathtt{b}\}^*$ denote S after exchanging $\mathtt{a}$'s and $\mathtt{b}$'s (i.e., $\bar{\mathsf{S}} = \mathtt{a} \Leftrightarrow \mathsf{S} = \mathtt{b}$). Further, let $\lessdot$ be the relation on strings such that $\mathsf{S} \lessdot \mathsf{T}$ if and only if $\mathsf{S} \prec \mathsf{T}$ and S is *not* a prefix of T. We need this relation since $\mathsf{S} \lessdot \mathsf{T} \iff \bar{\mathsf{S}} \succ \bar{\mathsf{T}}$ while $\mathsf{S} \prec \mathsf{T}$ and $\bar{\mathsf{S}} \prec \bar{\mathsf{T}}$ holds if S is a prefix of T.

- For $i \in [1..f_{m-1})$, we have $\mathsf{F}_m[i..] \succ \mathsf{F}_m[i + f_{m-2}..]$ due to [11, Lemma 7], thus $\bar{\mathsf{F}}_m[i..] \lessdot \bar{\mathsf{F}}_m[i + f_{m-2}..]$.
- For $i \in (f_{m-1}..f_m]$, since $\mathsf{F}_m = \mathsf{F}_{m-1}\mathsf{F}_{m-2} = \mathsf{F}_{m-2}\mathsf{F}_{m-3}\mathsf{F}_{m-2}$, $\mathsf{F}_m[i..]$ is a prefix of $\mathsf{F}_m[i - f_{m-1}..] = \mathsf{F}_m[i + f_{m-2} \bmod f_m..]$. Therefore, $\bar{\mathsf{F}}_m[i..]$ is a prefix $\bar{\mathsf{F}}_m[i + f_{m-2} \bmod f_m..]$ and $\bar{\mathsf{F}}_m[i..] \prec \bar{\mathsf{F}}_m[i + f_{m-2} \bmod f_m..]$.

Since f_m and f_{m-2} are coprime, $\{i + f_{m-2} \bmod f_m \mid i > 0\} = [1..n]$. Starting with the smallest suffix $\bar{\mathsf{F}}_m[f_m..]$, we end up at the largest suffix $\bar{\mathsf{F}}_m[f_m + (f_m - 1)f_{m-2} \bmod f_m..]$ after $m-1$ arithmetic progression steps of the form $\bar{\mathsf{F}}_m[f_m + if_{m-2} \bmod f_m..]$ for $i \in [0..f_m - 1]$. By using one of the two above items we can show that these arithmetic progression steps yield a list of suffixes sorted in lexicographically ascending order. □

5. Extension to larger alphabets

In this section we extend our results of Section 3 to alphabets of arbitrary size. Let $P = [p_1, \dots, p_n]$ be an arithmetically progressed permutation with ratio k. Let $\Sigma = [1..\sigma]$ be an alphabet of size σ where the order is given by $1 < 2 < \dots < \sigma$. To construct a string T over the alphabet Σ having P as its suffix array we proceed similarly to the construction presented in the previous sections: First we split P into subarrays $S_1, \dots, S_\sigma$, then for each subarray S_i we assign the character i to each position $p \in S_i$. When splitting P into subarrays there are some fixed positions where we are required to split P, while the remaining splitting positions can be chosen freely. Let $\sigma_{\min}$ be the size of the minimal alphabet over which there is a string having P as its suffix array. Then there are $\sigma_{\min} - 1$ positions where we are required to split P. Those required splitting positions are after the following entries, modulo n:

$$\begin{aligned} &\{p_1 - k - 1, n - k\} && \text{if } p_1 \notin \{1, k+1, n\}, \\ &\{n - k\} && \text{if } p_1 \in \{1, k+1\}, \\ &\{p_1 - k - 1\} && \text{if } p_1 = n \text{ and } k \neq n - 1. \end{aligned}$$

Example 5.1. Consider the alphabet $\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}, \mathtt{e}\}$ with $\sigma := |\Sigma| = 5$ and order $\mathtt{a} < \mathtt{b} < \mathtt{c} < \mathtt{d} < \mathtt{e}$, and the permutation $P = [5, 2, 7, 4, 1, 6, 3, 8]$ which has length $n = 8$, is arithmetically progressed with ratio $k = 5$ and starts with the entry $p_1 = 5$. The minimum alphabet size to construct a string having P as suffix array is $\sigma_{\min} = 3$. Thus there are two required splitting positions. Those are after 3 and 7. Using vertical bars | to denote the splitting positions we obtain $P = [5, 2, 7|4, 1, 6, 3|8]$. There are $\sigma - \sigma_{\min}$ splitting positions left which can be chosen freely. Assume we choose to split after entries 1 and 2. This leads to $P = [5, 2|7|4, 1|6, 3|8]$. Assigning characters we obtain the string $\mathsf{T} = \mathtt{cadcadbe}$.

At the beginning of this paper we looked at the strings over an alphabet of size σ having the suffix array $[n, n-1, \ldots, 1]$. The number of those strings is given by Theorem 2.2 and is bounded by a polynomial in n and σ. We extend upon that result: For an arbitrary permutation, which is not necessarily arithmetically progressed we give a bound on the number of strings with that permutation as suffix array. For a fixed arithmetically progressed permutation P we give an exact formula for the number of strings having P as their suffix array. We conclude that in total the number of strings having an arithmetically progressed suffix array is bounded by a polynomial in n and σ.

Lemma 5.2. *Let Σ be an alphabet of size $\sigma = |\Sigma|$. Given a permutation P of length n, there are at most $\binom{n+\sigma-1}{n}$ strings of length n over the alphabet Σ having P as their suffix array.*

Proof. Let the alphabet Σ be given by $\Sigma = [1..\sigma]$ with the order $1 < 2 < \cdots < \sigma$. Let T be a string over Σ of length n whose suffix array is P. Permuting T by its suffix array P we obtain the string T' with $\mathsf{T}'[i] = \mathsf{T}[P[i]]$. As P is the suffix array of T we have $\mathsf{T}'[i..n] \prec \mathsf{T}'[j..n]$ and thus $\mathsf{T}'[i] \leq \mathsf{T}'[j]$ for all $i < j$. Therefore T' is given by $\mathsf{T}' = 1^{t_1} \cdots \sigma^{t_\sigma}$, where t_i describes the number of occurrences of i in T'. The problem of finding the number of strings with this particular shape can be reduced to the classic stars and bars problem[13] [27, Chp. II, Sect. 5] with n stars and $\sigma - 1$ bars, yielding $\binom{n+\sigma-1}{n}$ possible strings. As the permutation P is a bijection on Σ^n, there is only one string T that, permuted by P, gives the string T'. Thus there are at most $\binom{n+\sigma-1}{n}$ strings having P as their suffix array. There may be less because it is possible that a string V, which is a pre-image of some $\mathsf{T}' = 1^{t_1} \cdots \sigma^{t_\sigma}$ under the permutation P, has a suffix array different to P. □

Lemma 5.3. *Let Σ be an alphabet of size $\sigma = |\Sigma|$. Given a permutation P of length n, let σ_{min} be the size of the smallest alphabet over which there exists a string having P as its suffix array. Then there are at most $\binom{n+\sigma-1}{\sigma-\sigma_{min}}$ strings of length n over the alphabet Σ having P as their suffix array.*

Proof. As described at the beginning of this section, all strings over the alphabet Σ having the suffix array P can be constructed by splitting P into $\sigma - 1$ subarrays. There are $\sigma_{\min} - 1$ positions where we are required to split P and $\sigma - \sigma_{\min}$ splitting positions that can be chosen freely. We model the selection of the positions that can be chosen freely using the stars and bars problem[14] [27, Chp. II, Sect. 5]. The stars represent the entries of the permutation, the bars the splitting positions. Assume we start with $n + \sigma - 1$ stars. Then $\sigma - \sigma_{\min}$ bars (splitting positions) can be freely chosen. There are $\binom{n+\sigma-1}{\sigma-\sigma_{\min}}$ ways to do this. Then we replace the stars at the $\sigma_{\min} - 1$ required splitting positions with bars. This gives us n stars and $\sigma - 1$ bars, which we map to a partition of P into σ subarrays. □

Lemma 5.4. *Let Σ be an alphabet of size $\sigma = |\Sigma|$. Then the number of strings of length n with an arithmetically progressed suffix array is at most $p(n, \sigma) = n(n-1)\binom{n+\sigma-1}{\sigma-\sigma_{min}}$.*

Proof. Each arithmetically progressed permutation P of length n can be described by two parameters: Its first entry $P[1]$ and its ratio k. For $P[1]$ there are n different options, for k there are at most $n - 1$ different options. Thus the number of arithmetically progressed permutations of length n is at most $n(n-1)$. By Lemma 5.3 the number of strings of length n having a specific permutation as suffix array is bounded by $\binom{n+\sigma-1}{\sigma-\sigma_{\min}}$. Putting those two facts together we obtain that the number of strings of length n with an arithmetically progressed suffix array is bounded by $p(n) = n(n-1)\binom{n+\sigma-1}{\sigma-\sigma_{\min}}$. □

Given a fixed alphabet Σ, by the above lemma, the number of strings of length n with an arithmetically progressed suffix array is bounded by a polynomial in n and σ.

6. Applications on meta strings

In this final section we overview some connections of suffix arrays with generalized forms of words and meta strings. A generalization of strings was proposed in the 1974 groundbreaking paper of Fischer and Paterson [42], where string matching was considered in a more general setting than with the usual solid letters, whereby either string could have *don't care*[15] symbols, and was achieved in time nearly as fast as linear. Uncertain sequences, including indeterminate

13 See Footnote 9.

14 See Footnote 9.

15 Don't care symbols match with all characters.

strings, have application in inexact matching tasks, for instance, allowing for errors such as with Web interface data entry and Internet search. They are also useful for expressing DNA polymorphisms, that is biological sequence positions that can have multiple possibilities and encoded with IUPAC[16] meta characters, for example N denotes any of the DNA nucleotides. A *codon* is a form of meta character whereby a sequence of three nucleotides encodes a specific amino acid and are used for protein expression; so a genetic code can be composed of concatenated codon units. The *truncated generalized suffix automaton* was introduced for indexing length-bounded k-factors of degenerate DNA and RNA sequences [43]. An *elastic-degenerate* string is a sequence of sets of strings used for succinctly representing a multiple alignment of a collection of closely related sequences (e.g. a pan-genome, that is all genes and genetic variation within a species), and also supports approximate pattern matching [44]. Sequence alignment is useful for inferring evolutionary relationships between biological sequences.

Daykin and Watson proposed a simple degenerate BWT, the D-BWT [45], constructed by applying lex-extension order (*cf.* Example 6.1) to relabel the sets and order the degenerate strings in the D-BWT matrix. Subsequently in [46], the D-BWT was applied to pattern matching using the backward search technique. This formalized and extended work implemented in [47] presenting a bioinformatics software tool, BWBBLE, for pattern matching on a pan-genome that they called a reference multi-genome.

In what follows, we first formally define indeterminate strings, and subsequently illustrate various approaches for defining an *indeterminate suffix array*: An *indeterminate string*[17] $\mathsf{T}^I = \mathsf{T}^I[1..n]$ on an alphabet Σ is a sequence of nonempty subsets of Σ. Specifically, an indeterminate string T^I has the form $\mathsf{T}^I = t_1 \cdots t_n$ where each t_i is a set of characters over Σ; a singleton is known as a *solid letter*. For example,

$$\mathsf{T}' = \{\mathtt{c}, \mathtt{a}, \mathtt{b}, \mathtt{e}\}\{\mathtt{d}\}\{\mathtt{c}, \mathtt{a}, \mathtt{b}, \mathtt{e}\}\{\mathtt{d}\}\{\mathtt{d}\}\{\mathtt{c}, \mathtt{a}, \mathtt{b}, \mathtt{e}, \mathtt{f}, \mathtt{g}\}$$

is a ternary border-free indeterminate string of length 6, with the indeterminate letters {c,a,b,e}, {d}, {c,a,b,e,f,g} where {d} is a singleton.

Example 6.1. Let $\mathsf{T}^I = \{\mathtt{b}\}\{\mathtt{b}\}\{\mathtt{c},\mathtt{a}\}\{\mathtt{b},\mathtt{d},\mathtt{a}\}$ be an indeterminate string, then $\mathsf{SA}_{\mathsf{T}^I}$ can be alternatively defined by the following approaches:

- We can apply the lex-extension order by treating each character set as a string whose characters are sorted, and use the lexicographic order of these strings to sort the sets, resulting in ranked meta characters: $\{\mathtt{a}, \mathtt{b}, \mathtt{d}\} = A$; $\{\mathtt{a}, \mathtt{c}\} = B$; $\{\mathtt{b}\} = C$. Then $\mathsf{T}^I = \mathtt{CCBA}$, and $\mathsf{SA}_{\mathsf{T}^I} = [4, 3, 2, 1]$, an arithmetically progressed permutation. This method enables linear-time construction of an indeterminate suffix array.
- Given we do not want the input to be rearranged, then we can form the meta characters as follows: $\{\mathtt{b}\} = A$; $\{\mathtt{b}, \mathtt{d}, \mathtt{a}\} = B$; $\{\mathtt{c}, \mathtt{a}\} = C$. Here, $\mathsf{T}^I = \mathtt{AACB}$ with suffix array $\mathsf{SA}_{\mathsf{T}^I} = [1, 2, 4, 3]$, which is not arithmetically progressed.
- Finally, we can incorporate the suffix arrays of the individual sets treated as strings, so the above suffix array [1,2,4,3], with both indeterminate and solid letter positions, becomes

$$[(1, 1), (2, 1), ((4, 3), (4, 1), (4, 2)), ((3, 2), (3, 1))].$$

Clearly, there is a natural generalization of many types of well-known patterned strings over solid letters to the indeterminate form, such as *indeterminate Fibonacci words*, where the solid form can give a meta representation of the generalized form.

7. Conclusion and problems

Given an arithmetically progressed permutation P with ratio k, we studied the minimum alphabet size and the shape of those strings having P as their suffix array. Only in the case $P = [n, n-1, \ldots, 1]$, a unary alphabet suffices. For general $P = [p_1, \ldots, p_n] \neq [n, n-1, \ldots, 1]$, there is exactly one such string on the binary alphabet if and only if $p_1 \in \{1, k+1, n\}$. In all other cases, there is exactly one such string on the ternary alphabet. Altogether, for an arbitrary but fixed alphabet size σ, only a polynomial portion of the σ^n strings of length n have an arithmetically progressed suffix array.

We conclude by proposing some research directions.

- A natural question arising from this research is to characterize strings having arithmetic progression properties for the run length exponents of their BWTs, particularly for the bijective [48] or extended BWT [49], which are always invertible.
  For example, given the arithmetically progressed permutation 3214, then the run-length compressed string $\mathtt{a}^3\mathtt{c}^2\$\mathtt{b}^4$ (a) matches the permutation 3214 and (b) is a BWT image because its inverse is $\mathtt{b}^2\mathtt{c}\mathtt{b}^2\mathtt{c}\mathtt{a}^3\$$, which can be computed by the Last-First mapping. However, for the same permutation, $\mathtt{a}^3\mathtt{b}^2\$\mathtt{b}^4$ does not work since it is not a BWT image. Further examples of arithmetically progressed BWT exponents are: $\mathtt{a}^3\mathtt{c}^4\$\mathtt{b}^2$, $\mathtt{a}^4\mathtt{c}^3\mathtt{e}^2\$\mathtt{d}^6\mathtt{b}^5$, and $\mathtt{a}^4\mathtt{c}^3\mathtt{e}^2\$\mathtt{f}^7\mathtt{d}^6\mathtt{b}^5$.

16 International Union of Pure and Applied Chemistry.

17 Also known as *degenerate string* in the literature, particularly in the field of microbiology.

- Arithmetic properties can likewise be considered for the following stringology integer arrays:
  - Firstly the *longest common prefix* (LCP) array LCP, whose entry LCP[i] is the length of the longest common prefix of the *lexicographically* ith smallest suffix with its lexicographic predecessor for $i \in [2..n]$.
  - Given a string $\mathsf{T} \in \Sigma^+$ of length n, the *prefix table* P_T of T is given by $P_\mathsf{T}[i] = \mathsf{LCP}(\mathsf{T}, \mathsf{T}[i..n])$ for $i \in [1..n]$; equivalently, the *border table* B_T of T is defined by

    $$B_\mathsf{T}[i] = \max\{|\mathsf{S}| \mid \mathsf{S} \text{ is a border of } \mathsf{T}[1..i]\} \text{ for } i \in [1..n].$$

  - Integer *prefix lists* are more concise than prefix tables and give the lengths of overlapping LCPs of T and suffixes of T (cf. [50]).
  - The ith entry of the *Lyndon array* $\lambda = \lambda_\mathsf{T}[1..n]$ of a given string $\mathsf{T} = \mathsf{T}[1..n]$ is the length of the longest Lyndon word that is a prefix of $T[i..]$ — reverse engineering in [51] includes a linear-time test for whether an integer array is a Lyndon array. Likewise, the *Lyndon factorization array* $\mathsf{F} = \mathsf{F}_\mathsf{T}[1..n]$ of T stores in its ith entry the size of the Lyndon factorization (i.e., the number of factors) of the suffix $\mathsf{T}[i..n]$. The problems are to characterize those arithmetic progressions that define a valid Lyndon array, respectively Lyndon factorization array. For example, consider the string $\mathsf{T} = \mathtt{azyx}$, then its Lyndon array is $\lambda_\mathsf{T} = [4, 1, 1, 1]$, while the Lyndon factorization array is $\mathsf{F}_\mathsf{T} = [1, 3, 2, 1]$. Trivially, for $\mathsf{T} = \mathtt{abc}\ldots\mathtt{z}$ the Lyndon array is an arithmetic progression and likewise for the Lyndon factorization array of $\mathsf{T} = \mathtt{z}^t\mathtt{y}^t\mathtt{x}^t\ldots\mathtt{a}^t$ for $\mathtt{z} > \mathtt{y} > \mathtt{x} > \cdots > \mathtt{a}$.
- A challenging research direction is to consider arithmetic progressions for multi-dimensional suffix arrays and Fibonacci word sequences.

Data availability

No data was used for the research described in the article.

Acknowledgments

We thank the reviewer for the careful proofreading and perceptive comments, which helped us polish this article. We also thank Gabriele Fici for the initial help in guiding this research started at the 2019 Pisa StringMasters.

In line with the community norm, we used alphabetic author order. We confirm that all authors contributed to the mathematical concepts and solutions along with writing and editing the manuscript, where Florian Stober is distinguished as the lead.

Funding

This research was part-funded by JSPS KAKENHI, Japan with grant numbers JP21K17701 and JP23H04378, and by the European Regional Development Fund through the Welsh Government [Grant Number 80761-AU-137 (West)]:

References

[1] U. Manber, E.W. Myers, Suffix arrays: A new method for on-line string searches, SIAM J. Comput. 22 (5) (1993) 935–948.
[2] M. Burrows, D.J. Wheeler, A Block Sorting Lossless Data Compression Algorithm, Technical Report 124, Digital Equipment Corporation, Palo Alto, California, 1994.
[3] D. Adjeroh, T. Bell, A. Mukherjee, The Burrows–Wheeler Transform: Data Compression, Suffix Arrays, and Pattern Matching, Springer, 2008.
[4] T. Bingmann, J. Fischer, V. Osipov, Inducing suffix and LCP arrays in external memory, ACM J. Exp. Algorithmics 21 (1) (2016) 2.3:1–2.3:27.
[5] J. Fischer, F. Kurpicz, Lightweight distributed suffix array construction, in: Proc. ALENEX, 2019, pp. 27–38.
[6] S. Mantaci, A. Restivo, M. Sciortino, Burrows–Wheeler transform and Sturmian words, Inform. Process. Lett. 86 (5) (2003) 241–246.
[7] M. Christodoulakis, C.S. Iliopoulos, Y.J.P. Ardila, Simple algorithm for sorting the Fibonacci string rotations, in: Proc. SOFSEM, in: LNCS, vol. 3831, 2006, pp. 218–225.
[8] J. Simpson, S.J. Puglisi, Words with simple Burrows–Wheeler transforms, Electron. J. Combin. 15 (1) (2008).
[9] W. Rytter, The structure of subword graphs and suffix trees of Fibonacci words, Theoret. Comput. Sci. 363 (2) (2006) 211–223.
[10] J. Berstel, A. Savelli, Crochemore factorization of Sturmian and other infinite words, in: Proc. MFCS, in: LNCS, vol. 4162, 2006, pp. 157–166.
[11] D. Köppl, T. I, Arithmetics on suffix arrays of Fibonacci words, in: Proc. WORDS, in: LNCS, vol. 9304, 2015, pp. 135–146.
[12] S. Giuliani, Z. Lipták, R. Rizzi, When a dollar makes a BWT, in: Proc. ICTCS, in: CEUR Workshop Proceedings, vol. 2504, 2019, pp. 20–33.
[13] G. Nong, S. Zhang, W.H. Chan, Linear suffix array construction by almost pure induced-sorting, in: Proc. DCC, 2009, pp. 193–202.
[14] M. Crochemore, R. Grossi, J. Kärkkäinen, G.M. Landau, Computing the Burrows–Wheeler transform in place and in small space, J. Discrete Algorithms 32 (2015) 44–52.
[15] D. Köppl, D. Hashimoto, D. Hendrian, A. Shinohara, In-place bijective Burrows–Wheeler transformations, in: Proc. CPM, in: LIPIcs, 2020, pp. 21:1–21:15.

[16] A. Alatabbi, J.W. Daykin, N. Mhaskar, M.S. Rahman, W.F. Smyth, Applications of V-order: Suffix arrays, the Burrows–Wheeler transform & the FM-index, in: Proc. WALCOM, in: LNCS, vol. 11355, 2019, pp. 329–338.
[17] F. Gray, Pulse code communication, 1953, (filed 1947). U.S. Patent 2, 632, 058.
[18] B. Chapin, S.R. Tate, Higher compression from the Burrows–Wheeler transform by modified sorting, in: Data Compression Conference, DCC 1998, IEEE Computer Society, 1998, p. 532.
[19] J.W. Daykin, R. Groult, Y. Guesnet, T. Lecroq, A. Lefebvre, M. Léonard, É. Prieur-Gaston, Binary block order Rouen transform, Theoret. Comput. Sci. 656 (2016) 118–134.
[20] J.W. Daykin, D. Köppl, D. Kübel, F. Stober, On Arithmetically Progressed Suffix Arrays, 2020, pp. 96–110.
[21] A. Amir, A. Levy, L. Reuveni, The practical efficiency of convolutions in pattern matching algorithms, Fund. Inform. 84 (1) (2008) 1–15.
[22] A. Amir, Y. Aumann, A. Levy, Y. Roshko, Quasi-distinct parsing and optimal compression methods, Theoret. Comput. Sci. 422 (2012) 1–14.
[23] A. Amir, A. Apostolico, E. Eisenberg, G.M. Landau, A. Levy, N. Lewenstein, Detecting approximate periodic patterns, Theoret. Comput. Sci. 525 (2014) 60–67.
[24] M. Lothaire, Combinatorics on Words, second ed., Cambridge Mathematical Library. Cambridge University Press, 1997.
[25] C. Hohlweg, C. Reutenauer, Lyndon words, permutations and trees, Theoret. Comput. Sci. 307 (1) (2003) 173–178.
[26] S. Mantaci, A. Restivo, G. Rosone, M. Sciortino, Suffix array and Lyndon factorization of a text, J. Discrete Algorithms 28 (2014) 2–8.
[27] W. Feller, An Introduction to Probability Theory and Its Applications, Wiley, 1968.
[28] J. Bernoulli, Sur une nouvelle espèce de calcul, Recueil Pour Astron. (1771) 255–284.
[29] J.-P. Borel, F. Laubie, Quelques mots sur la droite projective Réelle, J. Théor. Nombres Bordeaux 5 (1) (1993) 23–51.
[30] J. Berstel, A. Lauve, C. Reutenauer, F. Saliola, Combinatorics on Words: Christoffel Words and Repetition in Words, in: CRM Monograph Series, vol. 27, American Mathematical Society, 2008.
[31] J. Borel, C. Reutenauer, On christoffel classes, RAIRO Theor. Inform. Appl. 40 (1) (2006) 15–27.
[32] K.T. Chen, R.H. Fox, R.C. Lyndon, Free differential calculus, IV – the quotient groups of the lower central series, Ann. of Math. 68 (1958) 81–95.
[33] J.-P. Duval, Factorizing words over an ordered alphabet, J. Algorithms 4 (4) (1983) 363–381.
[34] F. Bassino, J. Clément, C. Nicaud, The standard factorization of Lyndon words: an average point of view, Discret. Math. 290 (1) (2005) 1–25.
[35] A.I. Shirshov, On bases for free Lie algebra, Algebra Log. 1 (1) (1962) 14–19, (in Russian).
[36] X.G. Viennot, Algèbres de Lie Libres et Monoïdes Libres, in: Lecture Notes in Mathematics, vol. 691, Springer, Berlin, 1978, p. 124.
[37] G. Melançon, Visualisation de Graphes et Combinatoire des Mots (Thèse d'Habilitation à Diriger les Recherches), Université Bordeaux I, 1999.
[38] D.E. Daykin, J.W. Daykin, W.F. Smyth, Combinatorics of unique maximal factorization families (UMFFs), Fund. Inform. 97 (3) (2009) 295–309.
[39] A. Restivo, G. Rosone, Balanced words having simple Burrows–Wheeler transform, in: International Conference on Developments in Language Theory, Springer, 2009, pp. 431–442.
[40] D. Wells, Prime Numbers: The Most Mysterious Figures in Math, Wiley, 2005.
[41] V. Hoggatt, M. Bicknell-Johnson, Composites and primes among powers of Fibonacci numbers increased or decreased by one, Fibonacci Quart. 15 (1977) 2.
[42] M.J. Fischer, M.S. Paterson, String matching and other products, in: Complexity of Computation, Vol. 7, 1974, pp. 113–125.
[43] T. Flouri, C.S. Iliopoulos, M.S. Rahman, L. Vagner, M. Voráček, Indexing factors in DNA/RNA sequences, in: Proc. BIRD, in: Communications in Computer and Information Science, vol. 13, Springer, 2008, pp. 436–445.
[44] G. Bernardini, N. Pisanti, S.P. Pissis, G. Rosone, Approximate pattern matching on elastic-degenerate text, Theoret. Comput. Sci. 812 (2020) 109–122.
[45] J.W. Daykin, B.W. Watson, Indeterminate string factorizations and degenerate text transformations, Math. Comput. Sci. 11 (2) (2017) 209–218.
[46] J.W. Daykin, R. Groult, Y. Guesnet, T. Lecroq, A. Lefebvre, M. Léonard, L. Mouchard, É. Prieur, B.W. Watson, Efficient pattern matching in degenerate strings with the Burrows–Wheeler transform, Inform. Process. Lett. 147 (2019) 82–87.
[47] L. Huang, V. Popic, S. Batzoglou, Short read alignment with populations of genomes, Bioinformatics 29 (13) (2013) 361–370.
[48] J.Y. Gil, D.A. Scott, A bijective string sorting transform, 2012, arXiv:1201.3077.
[49] S. Mantaci, A. Restivo, G. Rosone, M. Sciortino, An extension of the Burrows–Wheeler transform, Theoret. Comput. Sci. 387 (3) (2007) 298–312.
[50] J. Clément, L. Giambruno, Representing prefix and border tables: results on enumeration, Math. Structures Comput. Sci. 27 (2) (2017) 257–276.
[51] J.W. Daykin, F. Franek, J. Holub, A.S.M.S. Islam, W.F. Smyth, Reconstructing a string from its Lyndon arrays, Theoret. Comput. Sci. 710 (2018) 44–51.