On Arithmetically Progressed Suffix Arrays*

Jacqueline W. Daykin[1], Dominik Köppl[2], David Kübel[3], and Florian Stober[4]

[1] Department of Computer Science, Aberystwyth University, UK; Department of Information Science, Stellenbosch University, South Africa
jwd6@aber.ac.uk; jackie.daykin@gmail.com,
[2] Kyushu University, Japan Society for Promotion of Science, Japan
dominik.koeppl@inf.kyushu-u.ac.jp,
[3] University of Bonn, Institute of Computer Science, Germany
kuebel@cs.uni-bonn.de,
[4] University of Stuttgart, Institute for Formal Methods of Computer Science (FMI), Germany
florian.stober@t-online.de

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 n. 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. These results give rise to numerous future research directions.

Keywords: arithmetic progression, suffix array, string combinatorics

1 Introduction

The integral relationship between the suffix array [18] (SA) and Burrows-Wheeler transform [4] (BWT) is explored in [1], which also illustrates the versatility of the BWT beyond its original motivation in lossless block compression [4]. 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., [3]), or on distributed systems (e.g. [11]). 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 with 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 [20,5,23].

Other studies based on Fibonacci words are the suffix tree [22] or the Lempel-Ziv 77 (LZ77) factorization [2]. In [16], the suffix array and its inverse of each even

* This work was initiated at the open problems session of StringMasters colocated with IWOCA 2019 (International Workshop on Combinatorial Algorithms).

Jacqueline W. Daykin, Dominik Köppl, David Kübel, Florian Stober: On Arithmetically Progressed Suffix Arrays, pp. 96–110.
Proceedings of PSC 2020, Jan Holub and Jan Žďárek (Eds.), ISBN 978-80-01-06749-9

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.

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 $\mathtt{\$} < \mathtt{a} < \mathtt{b}$. However, the traditional BWT constructed by reading the last characters of the lexicographically sorted cyclic rotations $[\mathtt{ab}\boldsymbol{\mathtt{b}}, \mathtt{ba}\boldsymbol{\mathtt{b}}, \mathtt{bb}\boldsymbol{\mathtt{a}}]$ 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. [13] for identifying all the positions in a string S that 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 [16, Sect. 5]. Similarly, both methods for constructing the BWT are compatible when the string T consists of a Lyndon word. The authors of [16, 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 $\Theta(n)$ time [21] compared to $\Theta(n^2)$ with the BWT matrix method [4]. By utilizing combinatorial properties of the BWT, an in-place algorithm is given by Crochemore et al. [8], 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.

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 $O(1)$.

The structure of the paper is as follows. In Section 2 we give the basic definitions and background, and also deal with the elementary case of a unary alphabet. The

main results are presented in Section 3: we justify the need for coprimality, then cover ternary and binary alphabets followed by considering inverse permutations, and finally link the binary characterization to Fibonacci words. We conclude in Section 4 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.

2 Preliminaries

Let Σ be an alphabet with size $\sigma := |\Sigma|$. An element of Σ is called a *character*[1]. 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*[2] 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 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 $\mathsf{a}, \mathsf{b}, \mathsf{c}$ with $\mathsf{a} < \mathsf{b} < \mathsf{c}$.

A string T is a *Lyndon word* if it is strictly least in the lexicographic order among all its cyclic rotations [17]. For instance, $\mathtt{abcac}$ and $\mathtt{aaacbaabaaacc}$ are Lyndon words, while the string $\mathtt{aaacbaabaaac}$ with border $\mathtt{aaac}$ is not.

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 i-th 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 $\mathsf{BWT}[i] = \mathsf{T}[\mathsf{SA}[i] - 1 \mod n]$ is the (SA-based) BWT of T.

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

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 $p_{n+1} := p_1$.[3] Here $x \bmod n := x$ if $x \leq n$ and $x - n \mod n$ otherwise for an integer $x \geq 1$. In what follows, we want to study (a) strings whose suffix arrays are

[1] Also known as *letter* or *symbol* in the literature.
[2] Also known as *word* in the literature.
[3] We can also support negative values of k: Given a negative $k < 0$, we exchange it with $k' := n - k \bmod n \in [1..n]$ and use k' instead of k.

arithmetically progressed permutations, and (b) the shape of these suffix arrays. For a warm-up, we start with the unary alphabet:

Theorem 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* $n-1$*.*

Conversely, 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 $\sum_{j=1}^{\sigma} s_j = n$. By construction, each character j has to appear after all characters k with $k > j$. Therefore, $\mathsf{T} = \sigma^{s_\sigma}\sigma^{s_{\sigma-1}} \cdots 1^{s_1}$ such that the position of the characters are uniquely determined. In other words, we can reduce this problem to the classic stars and bars problem [10, Chp. II, Sect. 5] with n stars and σ bars, yielding $\binom{n+\sigma-1}{n}$ possible strings. Hence we obtain:

Theorem 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 have the form $\sigma^{s_\sigma}\sigma^{s_{\sigma-1}} \cdots 1^{s_1}$. The BWT based on the suffix array 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.

p_1	k	Min. Size of Σ	Properties of Strings	Reference
1		2	unique, Lyndon word	Theorem 11
$k+1$		2	unique, period $(n-k)$	Theorem 11
n	$\neq (n-1)$	2	unique, period $(n-k)$	Theorem 11
	$= (n-1)$	1	trivially periodic	Theorem 2
$\notin \{1, k+1, n\}$		3	unique	Theorem 4

Figure 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. Size of* Σ denotes the smallest possible size of Σ for which there exists such a string whose characters are drawn from Σ.

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

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

Figure 2. T of Eq. 1 for each arithmetically progressed permutation P of length $n = 8$ with ratio $k = 5$, starting with $p_1 := P[1] = k = 5$. The permutation of the k-th row is the k-th cyclic rotation of the permutation P in the first row. The splitting of P into the subarrays is visualized by the | 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 11). For $i \in [1..n]$, $\mathsf{BWT_T}[i] = \mathsf{T}[P[i + n - k^{-1} \bmod n]] = \mathsf{T}[P[i + 3 \bmod n]]$ with $k^{-1} = k = 5$ defined in Section 3.4.

with $\mathsf{SA_T} = P$ 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*[4] 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 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 $\{(ki) \bmod n \mid i \geq 1\} \subsetneq [1..n]$, from which we obtain:

Lemma 3. *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 $(p_1 - k - 1) \bmod n$ into the three subarrays A, B, and C (one of which is possibly empty) such that $P = ABC$. Subsequently, we set

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

Figure 2 gives an example of induced ternary/binary strings.

[4] Also known as *relatively prime* in the literature.

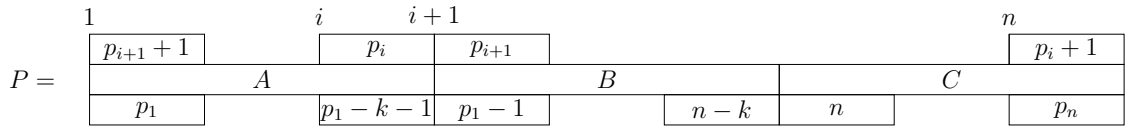

Figure 3. Setting of the proof of Theorem 4 with the condition $p_i + 1 = p_1 - k = p_n$. In the Figure we assume that the entry $p_1 - k - 1$ appears before $n - k$ in P.

Theorem 4. *Given an arithmetically progressed permutation* $P := [p_1, \ldots, p_n] \neq [n, n-1, \ldots, 1]$ *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 $\mathtt{a}$ lexicographically precede the suffixes starting with $\mathtt{b}$, which lexicographically precede the suffixes starting with $\mathtt{c}$. 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 $p_n = n$).

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 $p_{i+1} + jk \in \{p_1 - 1, n\}$.

□

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 5. *Given an arithmetically progressed permutation* $P := [p_1, \ldots, p_n] \neq [n, n-1, \ldots, 1]$ *with ratio* k *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.*

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`

Figure 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 down 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 5, this is not the case in general for the binary strings studied in Section 3.3, where we observe that $\mathsf{BWT}_{\texttt{bbabbabb}} = \texttt{bbbbbaab}$ defined by the suffix array differs from `bbbbbaba` (the last column on the right)

Proof. According to Theorem 4, $\mathsf{SA}_\mathsf{T} = P$, 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 $\mathsf{BWT}_{\text{matrix}}$ based on the BWT matrix is obtained by reading the last column of the BWT matrix from top down (see Fig. 4). Formally, $\mathsf{BWT}_{\text{matrix}}[i] = \mathsf{T}[Q[i] - 1 \mod n]$, where $Q[i]$ is the starting position of the lexicographically i-th smallest rotation $\mathsf{T}[Q[i]..n]\mathsf{T}[1..Q[i]-1]$. 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 6. *Let $P := [p_1, \ldots, p_n] \neq [n, n-1, \ldots, 1]$ be an arithmetically progressed permutation with ratio k. Further, let $\mathsf{T}[1..n]$ be given by Eq. 1 such that $\mathsf{SA}_\mathsf{T} = P$ according to Theorem 4. Given that $p_t = p_1 - 1 - k \mod n$ for a $t \in [1..n]$, BWT_T is*

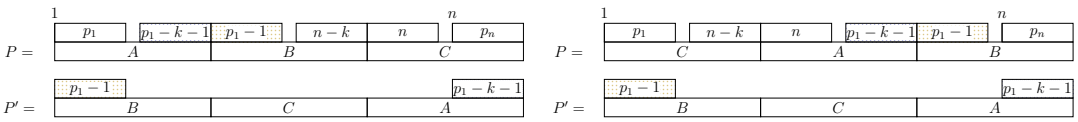

Figure 5. Setting of Eq. 1 with the distinction whether the entry $p_1 - k - 1$ appears before (left) or after (right) $n - k$ in P, yielding a different shape of the $\mathsf{BWT_T}$ defined as $\mathsf{BWT_T}[i] = \mathsf{T}[P'[i]]$ with $P'[i] = \mathsf{SA_T}[i] - 1 \mod n$.

given by the t-th rotation of $\mathsf{T}[\mathsf{SA}[1]] \cdots \mathsf{T}[\mathsf{SA}[n]]$, *i.e.,* $\mathsf{BWT_T}[i] = \mathsf{T}[P[i + t \mod n]]$ *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, $p_1 - 1 - k$ 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 $\mathsf{BWT_T}$ with $\mathsf{BWT_T}[i] = \mathsf{T}[P'[i]]$ is the t-th 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 6, we obtain the following corollary which shows that the number of runs in $\mathsf{BWT_T}$ for T defined in Eq. 1 are minimal:

Corollary 7. *For an arithmetically progressed permutation* $P := [p_1, \ldots, p_n] \neq [n, n-1, \ldots, 1]$ *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 8. *Given an arithmetically progressed permutation* $P := [p_1, \ldots, p_n]$ *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 8, 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 9. *Let $P := [p_1, \ldots, p_n]$ be an arithmetically progressed permutation with ratio k and $p_1 = 1$ for a binary string T over $\Sigma = \{\mathtt{a}, \mathtt{b}\}$ with $\mathsf{SA}_\mathsf{T} = P$. 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 $\mathsf{SA}_{\mathsf{T}'}$ is the m-th rotation of P with $\mathsf{SA}_{\mathsf{T}'}[1] = n$. Furthermore, $\mathsf{BWT}_\mathsf{T} = \mathsf{BWT}_{\mathsf{T}'}$.*

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

Consider two text positions $p_i, p_j \in [p_2, \ldots, p_n]$ 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 $[p_2, \ldots, p_n]$ and $[p'_2, \ldots, p'_n]$ is the same. In total, we have $p'_i = p_i - 1 \mod n$ for $i \in [1..n]$, hence P' is an arithmetically progressed permutation with ratio k. Given the first m entries in P index represent all suffixes of T starting with $\mathtt{a}$, P' is the m-th 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 6, 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 10. *Given an arithmetically progressed permutation $P := [p_1, \ldots, p_n] \neq [n, n-1, \ldots, 1]$ with ratio k 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 $\mathsf{SA}_\mathsf{T} = P$.*

Proof. If $p_1 = 1$, then P is split after the occurrences of the values $n-k$ and $-k = n-k \bmod n$, 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 symbol $\mathsf{T}[n] \leftarrow \mathtt{b}$ as in Theorem 4, 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[5], periodic, or a Lyndon word.

[5] The exact number of these binary strings is not covered by Theorem 8.

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

Figure 6. All binary strings of length 8 whose suffix arrays are arithmetically progressed permutations with ratio $k = 5$. Theorem 11 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.

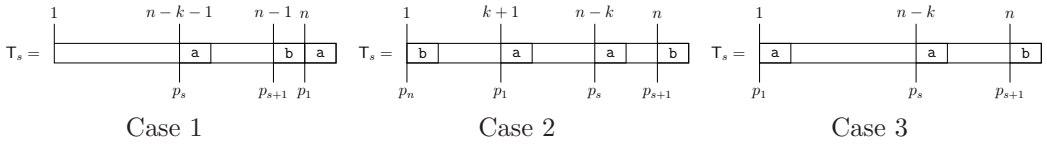

Figure 7. Sketches of the cases of Theorem 11. T_s is uniquely determined if the suffix array SA of T_s is arithmetically progressed with ratio k and the first entry $\mathsf{SA}[1] \in \{1, k+1, n-k\}$ is given.

Theorem 11. *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 $\mathsf{T}_s \in \{a, b\}^+$ 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 $s \in [1..n-1]$ (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.

For $k = n - 1$, Cases 2 and 3 each yields exactly one binary string, but Case 1 yields n binary strings according to Theorem 2.

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 1: 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 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 $\mathsf{S}[p_1] = \ldots = \mathsf{S}[j-1] = \mathtt{a}$ 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 arithmentically progressed permutation with ratio k, and first entry $p_1 \in \{1, k+1, n\}$, the string T_s of Theorem 11 coincides with T of Theorem 10.

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.

Theorem 12. *The inverse P^{-1} of an arithmetically progressed permutation P with ratio k is an arithmetically progressed permutation with ratio k^{-1} and $P^{-1}[1] = (1 - P[n]) \cdot k^{-1} \mod n$.*

Proof. Let $x := P[i]$ for an index $i \in [1..n]$. Then $P[i + k^{-1} \bmod n] = x - k \cdot k^{-1} \bmod n = x - 1 \bmod n$. 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 $P^{-1}[x-1 \bmod n] - P^{-1}[x]$ 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)

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

□

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

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

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 11:

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, $\mathsf{ISA}[p_s] = (-1)k^{-1} \mod n = n - k^{-1} \mod n$.
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, $\mathsf{ISA}[p_s] = (n - k - 1)k^{-1} \mod n = nk^{-1} - 1 - k^{-1} \mod n = n - 1 - k^{-1} \mod n$.
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, $\mathsf{ISA}[p_s] = (-1)k^{-1} \mod n = n - k^{-1} \mod n$ 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 9. Finally, we can fix the parameter t in Lemma 6 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 $\mathsf{ISA}[p_t] = \mathsf{ISA}[p_1 - 1 - k] = (-1)k^{-1} \mod n = n - k^{-1} \mod n$.

3.5 Relation to the Fibonacci word sequence

Köppl and I [16, 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 11 generalizes this observation by characterizing all binary strings whose suffix arrays are arithmetically progressed. Hence, F_m must coincide with Case 1 of Theorem 11 since it ends with character $\mathtt{a}$.

Lemma 13. *The Fibonacci word F_m for even m is given by*

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

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 [24, Fibonacci numbers]. Hence, f_{m-1} and f_m are coprime for every $m \ge 2$.
- $f_{m-2}^2 \bmod f_m = 1$ holds for every even $m \ge 3$ [14]. 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 12, $\mathsf{ISA}_{\mathsf{F}_m}[i] = if_{m-2} - \mathsf{SA}_{\mathsf{F}_m}[f_m]f_{m-2} \bmod f_m = if_{m-2} + 1 \bmod f_m$, 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 m-th 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 14. $\mathsf{SA}_{\bar{\mathsf{F}}_m}$ *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 11. We show that $k = f_{m-2}$ by proving

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

in a way similar to [16, 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 $\mathsf{S} \lessdot \mathsf{T} \iff \bar{\mathsf{S}} \gtrdot \bar{\mathsf{T}}$ 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 $\mathsf{F}_m[i..] \gtrdot \mathsf{F}_m[i + f_{m-2}..]$ due to [16, 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. □

4 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.We conclude by proposing some research directions.

- Prove which of the solutions for binary strings (if any) yield *balanced words*, i.e., binary strings T such that 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.
- 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 [12] or extended BWT [19], which are always invertible.

  For example, given the arithmetically progressed permutation 3214, then the run-length compressed string $\mathtt{a}^3\mathtt{c}^2\mathtt{\$}\mathtt{b}^4$ (a) matches the permutation 3214 and (b) is a BWT image because its inverse is $\mathtt{b}^2\mathtt{c}\mathtt{b}^2\mathtt{c}\mathtt{a}^3\mathtt{\$}$, which can be computed by the Last-First mapping. However, for the same permutation, $\mathtt{a}^3\mathtt{b}^2\mathtt{\$}\mathtt{b}^4$ does not work since it is not a BWT image.
- Arithmetic properties can likewise be considered for the following stringology integer arrays:
  - Firstly the *longest common prefix* (LCP) array LCP, whose entry $\mathsf{LCP}[i]$ is the length of the longest common prefix of the *lexicographically* i-th 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. [7]).
  - The i-th 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 [9] 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 in [6] gives at each position i the number of factors in the Lyndon factorization starting at i, that is the number of factors in the suffix $\mathsf{F}_\mathsf{T}[i..n]$. The problems are to characterize those arithmetic progressions which 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 $\mathsf{T} = \mathtt{z}^t\mathtt{y}^t\mathtt{x}^t \ldots \mathtt{a}^t$.
- A challenging research direction is to consider arithmetic progressions for multi-dimensional suffix arrays and Fibonacci word sequences.

Acknowledgements

We thank Gabriele Fici for the initial help in guiding this research started at String-Masters.

Funding: This research was part-funded by JSPS KAKENHI with grant number JP18F18120, and by the European Regional Development Fund through the Welsh Government [Grant Number 80761-AU-137 (West)]:

References

1. D. Adjeroh, T. Bell, and A. Mukherjee: *The Burrows-Wheeler Transform: Data compression, Suffix arrays, and Pattern matching*, Springer, 2008.
2. J. Berstel and A. Savelli: *Crochemore factorization of Sturmian and other infinite words*, in Proc. MFCS, vol. 4162 of LNCS, 2006, pp. 157–166.
3. T. Bingmann, J. Fischer, and V. Osipov: *Inducing suffix and LCP arrays in external memory.* ACM Journal of Experimental Algorithmics, 21(1) 2016, pp. 2.3:1–2.3:27.
4. M. Burrows and D. J. Wheeler: *A block sorting lossless data compression algorithm*, Tech. Rep. 124, Digital Equipment Corporation, Palo Alto, California, 1994.
5. M. Christodoulakis, C. S. Iliopoulos, and Y. J. P. Ardila: *Simple algorithm for sorting the Fibonacci string rotations*, in Proc. SOFSEM, vol. 3831 of LNCS, 2006, pp. 218–225.
6. A. Clare and J. W. Daykin: *Enhanced string factoring from alphabet orderings.* Inf. Process. Lett., 143 2019, pp. 4–7.
7. J. Clément and L. Giambruno: *Representing prefix and border tables: results on enumeration.* Mathematical Structures in Computer Science, 27(2) 2017, pp. 257–276.
8. M. Crochemore, R. Grossi, J. Kärkkäinen, and G. M. Landau: *Computing the Burrows-Wheeler transform in place and in small space.* J. Discrete Algorithms, 32 2015, pp. 44–52.
9. J. W. Daykin, F. Franek, J. Holub, A. S. M. S. Islam, and W. F. Smyth: *Reconstructing a string from its Lyndon arrays.* Theor. Comput. Sci., 710 2018, pp. 44–51.
10. W. Feller: *An introduction to probability theory and its applications*, Wiley, 1968.
11. J. Fischer and F. Kurpicz: *Lightweight distributed suffix array construction*, in Proc. ALENEX, 2019, pp. 27–38.
12. J. Y. Gil and D. A. Scott: *A bijective string sorting transform.* ArXiv 1201.3077, 2012.
13. S. Giuliani, Z. Lipták, and R. Rizzi: *When a dollar makes a BWT*, in Proc. ICTCS, vol. 2504 of CEUR Workshop Proceedings, 2019, pp. 20–33.
14. V. Hoggatt and M. Bicknell-Johnson: *Composites and Primes Among Powers of Fibonacci Numbers increased or decreased by one.* Fibonacci Quarterly, 15 1977, p. 2.
15. D. Köppl, D. Hashimoto, D. Hendrian, and A. Shinohara: *In-place bijective Burrows-Wheeler transformations*, in Proc. CPM, LIPIcs, 2020, p. to appear.
16. D. Köppl and T. I: *Arithmetics on suffix arrays of Fibonacci words*, in Proc. WORDS, vol. 9304 of LNCS, 2015, pp. 135–146.
17. M. Lothaire: *Combinatorics on Words*, Cambridge Mathematical Library, Cambridge University Press, 2 ed., 1997.
18. U. Manber and E. W. Myers: *Suffix arrays: A new method for on-line string searches.* SIAM J. Comput., 22(5) 1993, pp. 935–948.
19. S. Mantaci, A. Restivo, G. Rosone, and M. Sciortino: *An extension of the Burrows-Wheeler transform.* Theor. Comput. Sci., 387(3) 2007, pp. 298–312.
20. S. Mantaci, A. Restivo, and M. Sciortino: *Burrows-Wheeler transform and Sturmian words.* Inf. Process. Lett., 86(5) 2003, pp. 241–246.
21. G. Nong, S. Zhang, and W. H. Chan: *Linear suffix array construction by almost pure induced-sorting*, in Proc. DCC, 2009, pp. 193–202.
22. W. Rytter: *The structure of subword graphs and suffix trees of Fibonacci words.* Theor. Comput. Sci., 363(2) 2006, pp. 211–223.
23. J. Simpson and S. J. Puglisi: *Words with simple Burrows-Wheeler transforms.* Electr. J. Comb., 15(1) 2008.
24. D. Wells: *Prime Numbers: The Most Mysterious Figures in Math*, Wiley, 2005.