On Arithmetically Progressed Suffix Arrays*

Jacqueline W. Daykin¹, Dominik Köppl², David Kübel³, and Florian Stober⁴

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.koepl@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 compare 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 [10], 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).
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 $\text{BWT}_T[i] = T[\text{SA}_T[i] - 1]$ for a string $T$ with suffix array $\text{SA}_T$ incompatible with the traditional BWT defined on the BWT matrix, namely the lexicographic sorting of all cyclic rotations of the string $T$.

For instance, $\text{SA}_{\text{bab}} = [2, 3, 1]$ with $\text{BWT}_{\text{bab}} = \text{bab}$, while $\text{SA}_{\text{babs}} = [4, 2, 3, 1]$ and $\text{BWT}_{\text{babs}} = \text{bba}$ with $\$ < a < b$. However, the traditional BWT constructed by reading the last characters of the lexicographically sorted cyclic rotations $[\text{abb}, \text{bab}, \text{bba}]$ of $\text{bab}$ yields $\text{bba}$, which is equal to $\text{BWT}_{\text{babs}} = \text{bba}$ 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. [5], 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 $\Sigma := \{a\}$ and a string $T$ of length $n$ over $\Sigma$, $\text{SA}_T = [n, n - 1, \ldots, 1]$ is an arithmetically progressed permutation with ratio $-1 \equiv n - 1 \mod n$.

For the case of a binary alphabet $\{a, b\}$, 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 $T = b^r a^s$ with $r, s \in [0..n]$ such that $r + s = n$ has this suffix array. Similarly, $T = a^{n-1} b$ has the suffix array $\text{SA}_T = [1, 2, \ldots, n]$, 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 $\Sigma$ be an alphabet with size $\sigma := |\Sigma|$. An element of $\Sigma$ is called a character\footnote{Also known as letter or symbol in the literature.}. Let $\Sigma^+$ denote the set of all nonempty finite strings over $\Sigma$. The empty string of length zero is denoted by $\varepsilon$; we write $\Sigma^* = \Sigma^+ \cup \{\varepsilon\}$. Given an integer $n \geq 1$, a string\footnote{Also known as word in the literature.} of length $n$ over $\Sigma$ takes the form $T = t_1 \cdots t_n$ with each $t_i \in \Sigma$. We write $T = T[1..n]$ with $T[i] = t_i$. The length $n$ of a string $T$ is denoted by $|T|$. If $T = uvw$ for some strings $u, w, v \in \Sigma^*$, 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 $u \neq T$ (resp. $w \neq T$ and $v \neq T$). We say that a string $T$ of length $n$ has period $p \in [1..n-1]$ if $T[i] = T[i+p]$ for every $i \in [1..n-p]$ (note that we allow periods larger than $n/2$). If $T = uv$, 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 $T \neq \varepsilon$ is called a border of $T$; a string is border-free if the only border it has is the empty string $\varepsilon$.

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

A string $T$ is a Lyndon word if it is strictly least in the lexicographic order among all its cyclic rotations\footnote{We can also support negative values of $k$: Given a negative $k < 0$, we exchange it with $k' := -k \mod n \in [1..n]$ and use $k'$ instead of $k$.}. For instance, $abcac$ and $aaacbaabaac$ are Lyndon words, while the string $aaacbaabaac$ with border $aac$ is not.

For the rest of the article, we take a string $T$ of length $n \geq 2$. The suffix array $SA := SA[T[1..n]]$ of $T$ is a permutation of the integers $[1..n]$ such that $T[SA[i][n]]$ is the $i$-th lexicographically smallest suffix of $T$. We denote with $ISA$ its inverse, i.e., $ISA[SA[i]] = i$. By definition, $ISA$ is also a permutation. The string $BWT$ with $BWT[i] = T[SA[i] - 1 \mod n]$ is the (SA-based) BWT of $T$.

The Fibonacci sequence is a sequence of binary strings $\{F_m\}_{m \geq 1}$ with $F_1 := 0$, $F_2 := 1$, and $F_m := F_{m-1}F_{m-2}$. Then $F_m$ and $f_m := |F_m|$ 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 $p_{n+1} := p_1$. Here $x \mod n := x$ 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...
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 \{ a \}, the suffix array of a string of length \( n \) over \{ 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 \( \Sigma = [1 \ldots \sigma] \) 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 \( SA_T = P \). Let \( s_j \geq 0 \) be the number of occurrences of the character \( j \in \Sigma \) appearing in \( T \). By construction, each character \( j \) has to appear after all characters \( k \) with \( k > j \). Therefore, \( T = \sigma^{s_\sigma} \sigma^{s_{\sigma-1}} \cdots 1^{s_1} \) 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 \( \sigma \) 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 \( \sigma \) 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 \( 1^{s_1} 2^{s_2} \cdots \sigma^{s_\sigma} \). 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.

**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 \( \Sigma \) denotes the smallest possible size of \( \Sigma \) for which there exists such a string whose characters are drawn from \( \Sigma \).

<table>
<thead>
<tr>
<th>( p_1 )</th>
<th>( k )</th>
<th>Min. Size of ( \Sigma )</th>
<th>Properties of Strings</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>unique, Lyndon word</td>
<td>Theorem 11</td>
<td></td>
</tr>
<tr>
<td>( k + 1 )</td>
<td>2</td>
<td>unique, period ( (n - k) )</td>
<td>Theorem 11</td>
<td></td>
</tr>
<tr>
<td>( n \neq (n - 1) )</td>
<td>2</td>
<td>unique, period ( (n - k) )</td>
<td>Theorem 11</td>
<td></td>
</tr>
<tr>
<td>( n = (n - 1) )</td>
<td>1</td>
<td>trivially periodic</td>
<td>Theorem 2</td>
<td></td>
</tr>
<tr>
<td>( \notin {1, k + 1, n} )</td>
<td>3</td>
<td>unique</td>
<td>Theorem 4</td>
<td></td>
</tr>
</tbody>
</table>

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 \( T \)
Given an arithmetically progressed permutation $P := [p_1, \ldots, p_n]$ with ratio $k$, we define the ternary string $T[1..n]$ 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

$$T[p_i] := \begin{cases} 
  a & \text{if } p_i \in A, \\
  b & \text{if } p_i \in B, \\
  c & \text{if } p_i \in C.
\end{cases} \quad (1)$$

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

---

4 Also known as relatively prime in the literature.
Theorem 4. Given an arithmetically progressed permutation \( P := [p_1, \ldots, p_n] \neq [n, n-1, \ldots, 1] \) with ratio \( k \), \( SA_T = P \) for \( T \) defined in Eq. 1.

Proof. Suppose we have constructed \( SA_T \). Since \( a < b < c \), 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, \( SA[1..|A|], SA[|A| + 1..|A| + |B|] \) and \( SA[|A| + |B| + 1..n] \) store the same text positions as \( A \), \( B \), and \( C \), respectively. Consequently, it remains to show that the entries of each subarray \( (A, B \text{ or } C) \) are also sorted appropriately. Let \( p_i \) and \( p_{i+1} \) be two neighboring entries within the same subarray. Thus, \( T[p_i] = T[p_{i+1}] \) holds, and the lexicographic order of their corresponding suffixes \( T[p_i..n] \) and \( T[p_{i+1}..n] \) is determined by comparing the subsequent positions, starting with \( T[p_i+1] \) and \( T[p_{i+1}+1] \). 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) \( p_i + 1 = p_i - k \) or (2) \( p_i = n - k \).

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

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 \( T[p_{i+1}] \). The split here ensures that we can compare the suffixes \( T[n-k..n] \) and \( T[n] \) by the characters \( T[n-k] < T[n] \). If we did not split here, \( T[n] = T[n-k] \), and the suffix \( T[n] \) would be a prefix of \( T[n-k..n] \), resulting in \( T[n] < T[n-k..n] \) (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 \( p_{i+1} + jk \in \{p_i - 1, n\} \).

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.
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 $T[1..n]$ be given by Eq. 4 such that $SA_T = P$ according to Theorem 4. Given that $p_i = p_1 - k \mod n$ for $i \in [1..n]$, $BWT_T$ is

\[
\begin{array}{c|c|c}
\text{BWT matrix of } \text{babbabac} & \text{BWT matrix of } \text{ccaccacb} & \text{BWT matrix of } \text{bbabbabb} \\
\hline
\text{abacabbb} & \text{acbcaccc} & \text{abbbabbb} \\
\text{abababab} & \text{acccaccc} & \text{abbbabbb} \\
\text{acababac} & \text{bcacaccc} & \text{abbbabbb} \\
\text{bacabbab} & \text{ccacaccc} & \text{bbababbb} \\
\text{babcabba} & \text{ccacccca} & \text{bbababbb} \\
\text{cbababca} & \text{ccaccabc} & \text{bbababbb} \\
\end{array}
\]
given by the $t$-th rotation of $T[SA[1]] \cdots T[SA[n]]$, i.e., $BWT_T[i] = T[P[i + t \mod n]]$ 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 $p'_1 = p_1 - 1 - k \mod n$ 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 $BWT_T$ with $BWT_T[i] = T[P'[i]]$ is the $t$-th rotation of $T[SA[1]] \cdots T[SA[n]]$. \hfill $\square$

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

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 $\Sigma = \{a, b\}$ with $SA_T = P$. Suppose that the number of $a$'s in $T$ is $m$ and that $T' = T[2] \cdots T[n]T[1]$ is the first rotation of $T$. Then $SA_{T'}$ is the $m$-th rotation of $P$ with $SA_{T'}[1] = n$. Furthermore, $BWT_T = BWT_{T'}$.

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

Consider two text positions $p_i, p_j \in [p_1, \ldots, p_n]$ with $T[p_i..n] = u_1 \cdots u_s < T[p_j..n] = v_1 \cdots v_t$.

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

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 $a$, $P'$ 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 $b$ in $T$. Finally, since the strings $T$ and $T'$ are rotations of each other, their BWTs are the same. \hfill \Box

Like the parameter $t$ of Lemma\textsuperscript{6} we will determine the parameter $m$ after Eq.\textsuperscript{3} in Section\textsuperscript{3.3}.

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.\textsuperscript{2} to be a string over the binary alphabet $\{a, b\}$ with $SA_T = P$.

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.\textsuperscript{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 $T[n]$ is smaller than all other suffixes starting with $c$. So instead of assigning the unique symbol $T[n] \leftarrow b$ as in Theorem\textsuperscript{4} we can assign $T[n] \leftarrow c$, which still makes $T[n]$ the smallest suffix starting with $c$. We conclude this case by converting the binary alphabet $\{a, c\}$ to $\{a, b\}$. Cf. Fig.\textsuperscript{2} where $T$ in Rotation (6) has become $babbabb$ with period $n - k = 3$. \hfill \Box

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\textsuperscript{4}, periodic, or a Lyndon word.

\textsuperscript{5} The exact number of these binary strings is not covered by Theorem\textsuperscript{8}

\[\text{Proceedings of the Prague Stringology Conference 2020}\]
To sum up, since $S$ 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 6](image_url)

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](image_url)

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 $SA[1] \in \{1, k + 1, n - k\}$ 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 $T_s \in \{a, b\}^+$ is characterized by

$$T_s[i] = \begin{cases} a & \text{for } i \in SA_{T_s}[1..s], \\ b & \text{otherwise,} \end{cases}$$

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 $SA_{T_s} = [p_1, \ldots, p_n]$:

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 $SA_S = P := [p_1, \ldots, p_n]$ 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 $S[p_i..n] < S[p_{s+1}..n]$ and thus $S[p_i] \leq S[p_{s+1}]$. We have $S[j] = S[j + k \mod n]$ for all $j \in [1..n] \setminus \{p_n, p_s\}$ since

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

To sum up, since $S[p_s..n] < S[p_{s+1}..n]$ by construction, $S[p_i..n] < S[p_{i+1}..n]$ 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 \( s > 1 \), \( T_s[n] = T_s[p_1] = a \) and \( T_s[n - 1] = b \) (otherwise \( T_s[n - 1..n] \) would be the second smallest suffix, i.e., \( P[2] = n - 1 \) and hence \( k = n - 1 \)). Consequently, \( T_s[n - 1..n] = T_s[p_s + 1..n] \) is the smallest suffix starting with \( b \), namely \( ba \), and therefore \( p_s = n - 1 - k \).

Case 2: If \( p_1 \neq n \), then \( T_s[n] = b \) (otherwise \( T_s[n] < T_s[p_1..n] \)). Therefore, \( T_s[n] \) is the smallest suffix starting with \( b \), and consequently \( p_s = n - k \).

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

- If \( p_n - (n - k) > 0 \), then \( T_s[p_n - (n - k)] = T_s[p_1] \neq T_s[p_n] \) breaks the periodicity.
- If \( p_s - (n - k) > 0 \), then \( T_s[p_s - (n - k)] = a \neq b = T_s[p_s + 1] \) 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 \( T_s[p_1..n] = T_s[1..n] \) starts at the first text position).

Note that Case 3 can be obtained from Case 2 by setting \( T_s[1] \leftarrow a \) (the smallest suffix \( T_s[k + 1..n] \) 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:

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

Since \( S[p_n - 1] = a \) and \( S[p_1 - 1] = b \) with \( p_n - 1 + k \mod n = p_1 - 1 \), the smallest suffix starting with \( b \) is located at index \( p_1 - 1 \). This is a contradiction as \( p_1 \neq n \) implies \( S[n] = b \) (if \( S[n] = a \), then \( SA[1] = n \) instead of \( SA[1] = p_1 \)) and thus the smallest suffix starting with \( b \) 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 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 $P[i + k^{-1} \mod n] = x - k \cdot k^{-1} \mod n = x - 1 \mod n$ for the inverse permutation $P^{-1}$ this means that $P^{-1}[x] = i$ and $P^{-1}[x-1 \mod n] = i+k^{-1} \mod n$. Thus the difference $P^{-1}[x-1 \mod n] - P^{-1}[x]$ 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)

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

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

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

we can rewrite $T_s$ defined in Eq. 2 as

$$T_s[i] = \begin{cases} a & \text{if } ISA[i] \leq p_s, \text{ or} \\ b & \text{otherwise} \end{cases}$$

where SA and ISA denote the suffix array and the inverse suffix array of $T_s$, respectively. Another result is that $ISA[p_s] = s$ is the number of a’s in $T_s$, for which we split the study into the cases of Theorem 11.

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

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

For that, write $ISA[i] = (i - p_t)k^{-1} \mod n = (i + k - p_t)k^{-1} \mod n$ and compute $ISA[p_t] = ISA[p_1 - 1 - k] = (-1)k^{-1} \mod n = n - k^{-1} \mod n$. 


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 $SA_{F_m}$ with ratio $f_{m-2} \mod f_m$ and $SA_{F_m}[1] = f_m$. 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 $a$.

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

$$F_m[i] = \begin{cases} a & \text{if } 1 + i \cdot f_{m-2} \mod f_m \leq f_{m-1}, \\ b & \text{otherwise}. \end{cases}$$

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 $m \geq 2$.
- $f_{m-2}^2 \mod f_m = 1$ holds for every even $m \geq 3$. Hence, $k^{-1} = k = f_{m-2}$.
- By definition, $F_m[f_m] = a$ if $m$ is even, and therefore $SA_{F_m}[1] = f_m$.

The split position $p_s$ is $f_m - k = f_{m-1}$. So $SA_{F_m}[f_m] = f_m - k = f_m - f_{m-2}$. By Theorem 12, $SA_{F_m}[i] = i f_{m-2} - SA_{F_m}[f_m] f_{m-2} \mod f_m = i f_{m-2} + 1 \mod f_m$, where $-SA_{F_m}[f_m] f_{m-2} \mod f_m = (m_{m-2} - f_m) f_{m-2} \mod f_m = 1 - f_m f_{m-2} \mod f_m = 1$.

The rest follows from Eq. 3. 

Let $\bar{F}_m$ denote the $m$-th Fibonacci word whose $a$’s and $b$’s are exchanged, i.e., $\bar{F}_m = a \Leftrightarrow F_m = b$.

Lemma 14. $SA_{\bar{F}_m}$ is arithmetically progressed with ratio $f_{m-2}$ for odd $m$.

Proof. Since $\bar{F}_m[\bar{F}_m] = a$ for odd $m$, $\bar{F}_m[f_m]..$ is the lexicographically smallest suffix. Hence, $SA := SA_{\bar{F}_m} = |\bar{F}_m|$. 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

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

in a way similar to [16, Lemma 8]. For that, let $\bar{S}$ of a binary string $S \in \{a, b\}^*$ denote $S$ after exchanging $a$’s and $b$’s (i.e., $\bar{S} = a \Leftrightarrow S = b$). Further, let $\prec$ be the relation on strings such that $S \prec T$ if and only if $S \prec T$ and $S$ is not a prefix of $T$. We need this relation since $S \prec T \iff \bar{S} \succ \bar{T}$ while $S \prec T$ and $\bar{S} \prec \bar{T}$ holds if $S$ is a prefix of $T$.

- For $i \in [1..f_{m-1}]$, we have $F_m[i..] \succ F_m[i + f_{m-2}..]$ due to [16, Lemma 7], thus $\bar{F}_m[i..] \prec \bar{F}_m[i + f_{m-2}..]$.
- For $i \in (f_{m-1}..f_m]$, since $F_m = F_m - F_m - F_m - 3 F_m - 2 F_m = F_m[i..]$ is a prefix of $F_m[i - f_{m-1}..] = F_m[i + f_{m-2} \mod f_{m..}]$. Therefore, $\bar{F}_m[i..]$ is a prefix $\bar{F}_m[i + f_{m-2} \mod f_{m..}]$ and $\bar{F}_m[i..] \prec \bar{F}_m[i + f_{m-2} \mod f_{m..}]$.

Since $f_m$ and $f_{m-2}$ are coprime, $(i + f_{m-2} \mod f_m \mid i > 0] = [1..n]$. Starting with the smallest suffix $\bar{F}_m[f_m..]$, we end up at the largest suffix $\bar{F}_m[f_m + (f_{m-1} - 1) f_{m-2} \mod f_{m..}]$ after $m - 1$ arithmetic progression steps of the form $\bar{F}_m[f_m + i f_{m-2} \mod f_{m..}]$ 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 $c \in \{a, b\}$, 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 $|U| = |V|$ of the infinite concatenation $T \cdot T \cdots$ 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 $a^3c^2\$b^4$ (a) matches the permutation 3214 and (b) is a BWT image because its inverse is $b^2cb^2ca^3\$, which can be computed by the Last-First mapping. However, for the same permutation, $a^4b^2\$b^4$ 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 $LCP[i]$ 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 $T \in \Sigma^+$ of length $n$, the prefix table $P_T$ of $T$ is given by $P_T[i] = LCP(T, T[i..n])$ for $i \in [1..n]$; equivalently, the border table $B_T$ of $T$ is defined by
    
    $$B_T[i] = \max\{|S| : S \text{ is a border of } T[i..i]|.$$

  • 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 $\lambda = \lambda_T[1..n]$ of a given string $T = T[1..n]$ 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 $F = F_T[1..n]$ 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 $F_T[i..n]$. The problems are to characterize those arithmetic progressions which define a valid Lyndon array, respectively Lyndon factorization array. For example, consider the string $T = azyx$, then its Lyndon array is $\lambda_T = [4, 1, 1, 1]$, while the Lyndon factorization array is $F_T = [1, 3, 2, 1]$. Trivially, for $T = abc\ldots z$ the Lyndon array is an arithmetic progression and likewise for the Lyndon factorization array of $T = z'y'x'x'\ldots a'$.  

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 StringMasters.

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