

Counting and Verifying Abelian Border Arrays of Binary Words

Mursalin Habib^a, Md. Salman Shamil^a, M. Sohel Rahman^{a,*}

^a*ALEDA Group, Department of CSE, BUET, Dhaka-1000, Bangladesh*

Abstract

In this note, we consider the problem of counting and verifying abelian border arrays of binary words. We show that the number of valid abelian border arrays of length n is 2^{n-1} . We also show that verifying whether a given array is the abelian border array of some binary word reduces to computing the abelian border array of a specific binary word. Thus, assuming the word-RAM model, we present an $O\left(\frac{n^2}{\log^2 n}\right)$ time algorithm for the abelian border array verification problem.

Keywords: abelian, algorithms, border array, binary word.

1. Introduction

In recent years, there has been much interest in the field of abelian stringology. The central concept of abelian stringology is that of *abelian equivalence*: two strings are abelian equivalent if they have the same letters with the same multiplicities. For example, the words LISTEN and SILENT are abelian equivalent. We refer the reader to Section 2 for more precise definitions, especially in the case of binary words.

By substituting string equality with abelian equivalence, we can get abelian analogs of many natural string problems and regularities, e.g., abelian pattern matching [1–3], abelian borders [4], abelian squares [5], common abelian factors [6] - just to name a few of the topics touched on in recent literature.

There has also been a long practice of studying string inference or string reverse engineering problems where, given an instance of a string data structure, one attempts to find a string that generates that given data structure (or report

*Corresponding author:

Email address: msrahman@cse.buet.ac.bd (M. Sohel Rahman)

if none exists). The first string reverse engineering problem was introduced by Franěk *et al.* [7] who proposed a method to check if any integer array was the border array of some string. Since then a plethora of string inference problems have been studied in the literature (e.g., [4, 8–16]).

In this note, we study the abelian analog of the problem introduced in [7] for binary words. In particular, given an integer array, we propose a method that decides whether the array is the abelian border array of some binary words. We show that this “abelian border array verification” problem reduces to computing the abelian border array of a specific binary word (Section 3.2). In addition, we count the number of “valid” abelian border arrays (Section 3.1) and present some properties thereof (Section 3.2). We also briefly discuss the problems for larger alphabets (Section 5).

2. Preliminaries

Let Σ be a finite set of *letters* called an *alphabet*. Then Σ^* is the set of all finite words over Σ . For a binary alphabet, we assume $\Sigma = \{0, 1\}$ and w is called a *binary word* if $w \in \Sigma^*$. The *length* of a word w is denoted by $|w|$. We will denote by Σ^n the set of all words of length n over Σ . The word w will often be represented as $w[1]w[2] \cdots w[|w|]$ where $w[i]$ refers to the i -th letter of w . For $1 \leq i \leq j \leq |w|$, we let $w[i \cdots j]$ denote the $j - i + 1$ length word $w[i]w[i + 1] \cdots w[j]$, also referred to as a factor of w . Furthermore, $w[i \cdots j]$ is called a prefix (suffix) if $i = 1$ ($j = |w|$). A prefix or a suffix of w is called *proper* if it is not equal to w . Given a binary word w , let $\text{ones}(w)$ be equal to the number of 1’s in w . Although not very common, we sometimes conveniently use the following notation: if $w = w[1]w[2] \cdots w[|w| - 1]w[|w|]$, $w' = w[1]w[2] \cdots w[|w| - 1]$ and $w[|w|] = \alpha$, then $w = w'\alpha$.

Two binary words x and y of equal length are said to be *abelian equivalent* if $\text{ones}(x) = \text{ones}(y)$. An *abelian border* of a binary word w is a proper prefix of w that is abelian equivalent to a proper suffix of w . Our results center around a data structure called the abelian border array which we define below.

Definition 2.1. *Let x be a binary word. The abelian border array of x , denoted by π_x , is an array of length $|x|$ such that for $1 \leq i \leq |x|$, $\pi_x[i]$ contains the length of the longest abelian border of $x[1 \cdots i]$. An array π is called a valid abelian border array if $\pi = \pi_x$ for some binary word x ; in that case, x is called a generating word of π .*

For example, $\pi_{0001001} = (0, 1, 2, 0, 4, 5, 0)$. Also, $(0, 1, 2, 0, 4, 4)$ is not a valid

abelian border array since there does not exist any binary word x such that $\pi_x = (0, 1, 2, 0, 4, 4)$.

Next, we introduce the notion of abelian border equivalence. Two binary words x and y are called *abelian border equivalent* if $\pi_x = \pi_y$, i.e., they have the same abelian border array. For example, $\pi_{0100} = \pi_{1011}$ and hence the two strings, 0100 and 1011 are abelian border equivalent.

Finally, it comes in handy to define the complement of a binary word.

Definition 2.2. *Given a binary word x , the complement of x , denoted by \bar{x} , is a binary word of length $|x|$ such that for $1 \leq i \leq |x|$*

$$\bar{x}[i] = \begin{cases} 1 & \text{if } x[i] = 0 \\ 0 & \text{otherwise.} \end{cases}$$

3. Results

3.1. Counting the Number of Valid Abelian Border Arrays

The first problem we tackle is counting the number of valid abelian border arrays of a particular length. More formally, suppose that Π_n is the set of n -length arrays such that for each $\pi \in \Pi_n$ there exists a binary word x with $|x| = n$ and $\pi = \pi_x$. Let T_n be the number of arrays in the set Π_n , i.e., $T_n = |\Pi_n|$. We prove the following proposition.

Proposition 3.1. $T_n = 2^{n-1}$.

To prove Proposition 3.1, we first prove the following lemma.

Lemma 3.1. *If x and y are two different binary words that are abelian border equivalent, then $y = \bar{x}$.*

Proof. We prove this by induction on the length of x and y . Clearly, the claim is true when $|x| = |y| = 1$. Assume the claim is true whenever $|x| = |y| = n - 1$.

Now let x and y be two different abelian border equivalent binary words of length $n > 1$. If we have $x[1 \cdots n - 1] = y[1 \cdots n - 1]$, then the fact that x and y are different will force $\pi_x \neq \pi_y$ making x and y abelian border non-equivalent, contradicting our hypothesis.

So, $x[1 \cdots n - 1] \neq y[1 \cdots n - 1]$. Since $\pi_{x[1 \cdots n - 1]} = \pi_{y[1 \cdots n - 1]}$, by the inductive hypothesis, we have $y[1 \cdots n - 1] = \overline{x[1 \cdots n - 1]}$. Therefore, it suffices to show that $x[n] \neq y[n]$.

For the sake of contradiction, let $x[n] = y[n] = \alpha$. Since $y[1 \cdots n - 1] = \overline{x[1 \cdots n - 1]}$, it follows that $x[1] \neq y[1]$. Without loss of generality, we can assume $x[1] = \alpha$. However, this forces $\pi_x[n] = n - 1$ and $\pi_y[n] < n - 1$ contradicting the assumption that x and y are abelian border equivalent. \square

From the lemma above, it is clear that every valid abelian border array has exactly two generating words and they are complements of each other. In other words, the generating word of a valid abelian border array is unique *up to complementation*. As a result, as far as abelian border arrays of binary words are concerned, it suffices to only consider words that start with a 0. We make this notion formal by defining *the* generating word of a valid abelian border array to be the generating word that starts with a 0.

The key to finding T_n lies in the observation that given a valid abelian border array of length $n - 1$, there are exactly two ways to extend it into a valid abelian border array of length n . The following lemmas explore this idea.

Lemma 3.2. *Let π be a valid abelian border array of length $n - 1$ and x be the generating word thereof. Then $\pi_{x0}[n] = n - 1$.*

Proof. Consider the word $y = x0$. By definition, we have $|y| = n$ and $y[1] = x[1] = 0 = y[n]$. So, $\text{ones}(y[1 \cdots n - 1]) = \text{ones}(x[2 \cdots n - 1]) = \text{ones}(y[2 \cdots n])$. Therefore $y = x0$ has an abelian border of length $n - 1$ and the result follows. \square

Lemma 3.3. *Let π be a valid abelian border array of length $n - 1$. If S is the set of all possible non-negative integers k such that appending k to the end of π gives a valid abelian border array of length n , then $|S| = 2$.*

Proof. Let x be the generating word of π . Now consider the string $x0$. By Lemma 3.2, we must have $\pi_{x0}[n] = n - 1$. Therefore, $n - 1 \in S$. The other element in S is also completely determined by x . In fact it is equal to $\pi_{x1}[n]$. The fact that there are no other elements in S follows from the uniqueness of x . \square

For example, $(0, 0, 0, 3, 3)$ is a valid abelian border array for which the generating word is 01101 and in this case, $S = \{3, 5\}$. Note that $(0, 0, 0, 3, 3, 3) = \pi_{011011}$ and $(0, 0, 0, 3, 3, 5) = \pi_{011010}$.

We are now ready to prove the main result of this section.

Proof of Proposition 3.1. Since prefixes of valid abelian border arrays are themselves valid abelian border arrays, the only way to get a valid abelian border array of length n is to extend a valid abelian border array of length $n - 1$. From Lemma 3.3, it follows that $T_n = 2T_{n-1}$. Since $T_1 = 1$, we have $T_n = 2^{n-1}$. \square

3.2. Verifying Valid Abelian Border Arrays

Now we turn to the problem of verifying abelian border arrays. More formally, given an array π , we want to find whether or not it is a valid abelian border array. In addition, if the answer is positive, we want to find the generating word thereof. We first look at some general properties of abelian border arrays.

Proposition 3.2. *Let x be a binary word of length n . For $1 \leq i \leq n$, the length of the shortest non-empty abelian border of $x[1 \cdots i]$ is equal to $i - \pi_x[i]$ provided that $\pi_x[i] \neq 0$.*

Proof. This follows from the fact that if a word w has an abelian border of length i , then it also has an abelian border of length $|w| - i$. It is then immediately clear why the lengths of the longest and shortest non-empty abelian borders should be related in this way. \square

Proposition 3.3. *Let π be a valid abelian border array of length n and let x be the generating word of π . For $1 \leq i \leq n$, $\pi[i] = i - 1$ if and only if $x[i] = 0$.*

Proof. The claim is obviously true if $i = 1$. So, let $i > 1$. Since x is the generating word of π , $x[1] = 0$. By proposition 3.2, $\pi[i] = i - 1$ if and only if the length of the shortest non-empty abelian border of $x[1 \cdots i]$ is 1. But since $x[1] = 0$, this can happen if and only if $x[i] = 0$. \square

Proposition 3.4. *Let x be a binary word of length n such that $x[1] = 0$. For $1 < i \leq n$, if $x[i] = 1$, then $\pi_x[i] \leq \pi_x[i - 1]$.*

Proof. Let $k = i - 1 - \pi_x[i - 1]$ and $k' = i - \pi_x[i]$. By proposition 3.2, k and k' are the lengths of the shortest non-empty abelian borders of $x[1 \cdots i - 1]$ and $x[1 \cdots i]$ respectively. Therefore, it suffices to show that $k' \geq k$.

Since k is the length of the shortest non-empty abelian border of $x[1 \cdots i - 1]$, for $1 \leq j < k$, $\text{ones}(x[1 \cdots j]) < \text{ones}(x[i - j \cdots i - 1])$. However, since $x[i] = 1$, $\text{ones}(x[i - j + 1 \cdots i]) \geq \text{ones}(x[i - j \cdots i - 1])$. So, $\text{ones}(x[1 \cdots j]) < \text{ones}(x[i - j + 1 \cdots i])$ for $1 \leq j < k$. So, we can conclude that for $1 \leq j < k$, we must have $\text{ones}(x[1 \cdots j]) \neq \text{ones}(x[i - j + 1 \cdots i])$. Therefore, k' can not be smaller than k . \square

Propositions 3.3 and 3.4 provide us with an insight into the structure of valid abelian border arrays. They tell us that consecutive elements of a valid abelian array can not increase “slowly”. Given a binary word x with $x[1] = 0$, $\pi_x[i]$ either jumps up to $i - 1$ (happens when $x[i] = 0$) or stays at most as high as $\pi_x[i - 1]$ (happens when $x[i] = 1$). A long run of 1s in x eventually brings $\pi_x[i]$ down to 0; after which a 0 in x brings it again up to $i - 1$.

Proposition 3.3 actually suggests a direct algorithm for our verification problem as we show below.

Proposition 3.5. *Let π be an array of length n . We define x_π to be a binary word of length n such that*

$$x_\pi[i] = \begin{cases} 0 & \text{if } \pi[i] = i - 1 \\ 1 & \text{otherwise.} \end{cases}$$

If π is a valid abelian border array, then $\pi = \pi_{x_\pi}$. \square

So, the problem of checking whether an array π is a valid abelian border array reduces to computing the abelian border array of a specific binary word x_π . If the computed abelian border array matches π , we output **yes** along with the word x_π . Otherwise, we output **no**.

The abelian border array of a binary word can be computed naively in $O(n^2)$ where n is the length of the word. But a recent result by Kociumaka *et al.* [17] shows that it can be done in $O\left(\frac{n^2}{\log^2 n}\right)$ time assuming the word-RAM model.

Proposition 3.6. *Assuming the word-RAM model, the valid abelian border array verification problem can be solved in $O\left(\frac{n^2}{\log^2 n}\right)$ time. \square*

4. Extending to Larger Alphabets

A natural thing to do is to try extending these results for words over larger alphabets. However, the problem becomes quickly difficult even for ternary words. The main reason is that it is hard to find a good characterization of abelian border equivalent words on larger alphabets. Two words can be very different but can still give the same abelian border array. For example, the words 011021 and 012022 are abelian border equivalent but at a first glance, they do not look anything alike.

Despite this, it is possible to come up with upper bounds for the answer to the counting problem for larger alphabets. The key idea is the following definition.

Definition 4.1. *Two words w_1 and w_2 with $|w_1| = |w_2| = n$ are said to be letter-equivalent if for all $1 \leq i, j \leq n$, $w_1[i] = w_1[j]$ if and only if $w_2[i] = w_2[j]$.*

Note that letter-equivalent words are a generalization of complement words (Definition 2.2) for larger alphabets. Clearly, if two words are letter-equivalent, then they are abelian border equivalent. However, the converse is not necessarily true for words on larger alphabets. We have already provided an example of this: the two words 011021 and 012022, despite not being letter-equivalent, are abelian border equivalent.

Letter-equivalence, as the name suggests, is an equivalence relation on the set Σ^n of all words of length n over Σ . Therefore, the set of distinct equivalence classes of letter-equivalence forms a partition of Σ^n . Clearly, the number of parts in this partition is an upper bound for T_n . Thus we have the following two results.

Proposition 4.1. *Let T_n be the number of n -length arrays π such that there exists a word x over $\Sigma = \{0, 1, 2\}$ with $\pi = \pi_x$. Then $T_n \leq \frac{3^{n-1}+1}{2}$.*

Proof. We count the number of distinct equivalence classes of letter-equivalence in Σ^n . Out of the 3^n words that form Σ^n , the 3 words that contain only one letter are in an equivalence class of their own. Each of the remaining $3^n - 3$ words are in an equivalence class with 5 other words that can be found by simply relabeling the letters (as an example, the word 0110 is in an equivalence class with the five words 0220, 1001, 1221, 2002, and 2112). Therefore, $T_n \leq 1 + \frac{3^n-3}{6} = \frac{3^{n-1}+1}{2}$. \square

Proposition 4.2. *Let $n \geq 2$ be an integer and $\Sigma = \{0, 1, 2, \dots, n-1\}$. If T_n is the number of n -length arrays π such that there exists a word x over Σ with $\pi = \pi_x$, then $T_n \leq B_n$ where B_n is the n th Bell number.*

Proof. Each word $w \in \Sigma^n$ induces a partition of the indices $1, 2, 3, \dots, n$ in the following way: for all $1 \leq i < j \leq n$, the indices i and j are in the same part of the partition if and only if $w[i] = w[j]$. Two words $w_1, w_2 \in \Sigma^n$ are letter-equivalent if and only if they induce the same partition of the indices. Therefore, an upper bound on T_n is the number of ways you can partition the set $\{1, 2, 3, \dots, n\}$. This number is precisely B_n [18]. \square

Therefore, for an unbounded alphabet T_n is upper-bounded by the n th Bell number. However, this bound is very loose and does not offer much insight into the structure of valid abelian border arrays.

5. Conclusion

In this note, we have taken on the problem of inferring a binary word from its abelian border array. Although regular string inference problems are abundant in the literature, inference problems of the abelian variety are surprisingly rare. We hope our work will be one of the first of many ventures into the world of abelian string inference problems.

Possible future work might include extending our results for words over larger alphabets. However, as the last section shows, doing this is non-trivial. Another line of work would be to ask if it is actually *necessary* to compute abelian border arrays at all to solve the verification problem. We have shown that it is sufficient (Proposition 3.5). But it might be possible for some other verification algorithm to exist that does not do any border array computation at all.

Declarations

Funding

Not Applicable.

Conflicts of interest/Competing interests

None declared.

Availability of data and material

Not Applicable.

Code availability

Not Applicable.

References

- [1] Péter Burcsi, Ferdinando Cicalese, Gabriele Fici, and Zsuzsanna Lipták. Algorithms for jumbled pattern matching in strings. *Internat. J. Found. Comput. Sci.*, 23(2):357–374, 2012.
- [2] Tanaeem M. Moosa and M. Sohel Rahman. Sub-quadratic time and linear space data structures for permutation matching in binary strings. *J. Discrete Algorithms*, 10:5–9, 2012.
- [3] Tanaeem M. Moosa and M. Sohel Rahman. Indexing permutations for binary strings. *Inform. Process. Lett.*, 110(18-19):795–798, 2010.
- [4] Manolis Christodoulakis, Michalis Christou, Maxime Crochemore, and Costas S. Iliopoulos. Abelian borders in binary words. *Discrete Appl. Math.*, 171:141–146, 2014.
- [5] L. J. Cummings and W. F. Smyth. Weak repetitions in strings. *J. Combin. Math. Combin. Comput.*, 24:33–48, 1997.
- [6] Ali Alatabbi, Costas S. Iliopoulos, Alessio Langiu, and M. Sohel Rahman. Algorithms for longest common abelian factors. *Internat. J. Found. Comput. Sci.*, 27(5):529–543, 2016.
- [7] František Franěk, Shudi Gao, Weilin Lu, P. J. Ryan, W. F. Smyth, Yu Sun, and Lu Yang. Verifying a border array in linear time. *J. Combin. Math. Combin. Comput.*, 42:223–236, 2002. 14th MCCC (Wichita, KS, 2000).
- [8] Tanaeem M. Moosa, Sumaiya Nazeen, M. Sohel Rahman, and Rezwana Reaz. Inferring strings from cover arrays. *Discrete Math. Algorithms Appl.*, 5(2):1360005, 15, 2013.

- [9] Dipankar Ranjan Baisya, Mir Md Faysal, Mohammad Sohel Rahman, et al. Degenerate string reconstruction from cover arrays. In *Stringology*, pages 191–205, 2013.
- [10] Sumaiya Nazeen, M. Sohel Rahman, and Rezwana Reaz. Indeterminate string inference algorithms. *J. Discrete Algorithms*, 10:23–34, 2012.
- [11] Jacqueline W. Daykin, Frantisek Franek, Jan Holub, A. S. M. Sohidul Islam, and W. F. Smyth. Reconstructing a string from its Lyndon arrays. *Theoret. Comput. Sci.*, 710:44–51, 2018.
- [12] Joel Helling, P. J. Ryan, W. F. Smyth, and Michael Soltys. Constructing an indeterminate string from its associated graph. *Theoret. Comput. Sci.*, 710:88–96, 2018.
- [13] Sumaiya Nazeen, M. Sohel Rahman, and Rezwana Reaz. Indeterminate string inference algorithms. *J. Discrete Algorithms*, 10:23–34, 2012.
- [14] Dipankar Ranjan Baisya, Mir Md. Faysal, and Mohammad Sohel Rahman. Degenerate string reconstruction from cover arrays. In Jan Holub and Jan Zdárek, editors, *Proceedings of the Prague Stringology Conference 2013, Prague, Czech Republic, September 2-4, 2013*, pages 191–205. Department of Theoretical Computer Science, Faculty of Information Technology, Czech Technical University in Prague, 2013.
- [15] Tanaeem M. Moosa, Sumaiya Nazeen, M. Sohel Rahman, and Rezwana Reaz. Inferring strings from cover arrays. *Discret. Math. Algorithms Appl.*, 5(2), 2013.
- [16] Ali Alatabbi, M. Sohel Rahman, and William F. Smyth. Inferring an indeterminate string from a prefix graph. *J. Discrete Algorithms*, 32:6–13, 2015.
- [17] Tomasz Kociumaka, Jakub Radoszewski, and Bartłomiej Wiśniewski. Subquadratic-time algorithms for abelian stringology problems. In *Mathematical aspects of computer and information sciences*, volume 9582 of *Lecture Notes in Comput. Sci.*, pages 320–334. Springer, 2016.
- [18] Martin Gardner. Bells-versatile numbers that can count partitions of a set, primes and even rhymes. *Scientific American*, 238(5):24–30, 1978.