Inferring Strings from Runs

Wataru Matsubara*[1], Akira Ishino[2], and Ayumi Shinohara[1]

[1] Graduate School of Information Science, Tohoku University, Japan
{matsubara@shino., ayumi@}ecei.tohoku.ac.jp
[2] Google Japan Inc.
ishino@google.com

Abstract. A run in a string is a nonextendable periodic substring in the string. Detecting all runs in a string is important and studied both from theoretical and practical points of view. In this paper, we consider the reverse problem of it. We reveal that the time complexity depends on the alphabet size k of the string to be output. We show that it is solvable in polynomial time for both binary alphabet and infinite alphabet, while it is NP-complete for finite $k \geq 4$. We also consider a variant of the problem where only a subset of runs are given as an input. We show that it is solvable in polynomial time for infinite alphabet, while it is NP-complete for finite $k \geq 3$.

Keywords: repetition, runs, inferring problem

1 Introduction

A reverse problem on strings is for a given data structure, inferring a string that does not conflict with the input information. A motivation of considering reverse problem is to characterize if-and-only-if conditions on the data structures. It is also of interest for design methods for the data structures.

Reverse problems on strings have been considered for various data structures. Franek et al. [8] initiated a linear time algorithm for testing whether an integer array is the Border Table of a string on unbounded size alphabet (infinite alphabet). Duval et al. [7] solves the same question for a bounded-size alphabet (finite alphabet). I et al. [10] considered for parametrized border array. Bannai et al. [1] solved three other data structures: Directed Acyclic Subsequence Graph, Directed Acyclic Word Graph, and Suffix Array. All of their three testing algorithms run in linear time. Clement et al. [3] showed a linear time algorithm for solving the reverse problem for Prefix Table. The reverse problem for the Longest Previous Factor Table is an open question.

Repetitions is one of most fundamental property of strings, it is important both theoretical and practical point of view. Detecting repetition in strings is an important element of several questions: pattern matching, text compression.

Kucherov and Kolpakov showed that considering maximal repetitions, or runs, the number of runs in any string of length n is $O(n)$. Although they were not able to give bounds for the constant factor, there have been several works to this end [13,14,12,4,2,9,5,15]. The known results in the topic and a deeper description of the motivation can be found in a survey by Crochemore et al. [6]. The currently known best upper bound[1] and lower bound are as follows:

$$0.944575 \leq \frac{\rho(n)}{n} \leq 1.029$$

* Supported in part by Grant-in-Aid for JSPS Fellows

[1] Presented on the website http://www.csd.uwo.ca/faculty/ilie/runs.html

Wataru Matsubara, Akira Ishino, Ayumi Shinohara: Inferring Strings from Runs, pp. 150–160 .
Proceedings of PSC 2010, Jan Holub and Jan Žďárek (Eds.), ISBN 978-80-01-04597-8

The upper bound obtained by calculations based on the proof technique of [4,5]. The technique bounds the number of runs for each string by considering runs in two parts: runs with long periods, and runs with short periods. The former is more sparse and easier to bound. The latter is bounded by an exhaustive calculation concerning how runs of different periods can overlap in an interval of some length. Considering all possibility, they show that a string of length n contains at most $0.93n$ runs with period up to 60.

We are inspired by their work, and we tackle to the reverse problem of detecting all runs in a string. That is, for a given set S of runs, we will find a string whose runs are consitent with S. We consider the following two situations.

1. Inferring strings whose runs are equal to S (the perfect input problem).
2. Inferring strings whose runs subsume S (the imperfect input problem).

We show that inferring strings over infinite alphabet is tractable in both settings. We show that it is also tractable for binary alphabet in the perfect input setting. On the other hand, the inferring string over ternary alphabet is intractable. For alphabet size k, we show that the perfect input problem is NP-hard for $k \geq 4$ and the inperfect input problem is NP-hard for $k \geq 3$.

2 Preliminary

2.1 Notations on Strings

Let $\mathcal{N} = \{0, 1, 2, \ldots\}$ be the set of natural numbers. Let Σ be a set of symbols. An element of Σ^* is called a *string*. Σ^n denotes the set of strings of length n. The length of a string w is denoted by $|w|$. The i-th character of a string w is denoted by $w[i]$ for $1 \leq i \leq |w|$, and the substring of a string w that begins at position i and ends at position j is denoted by $w[i : j]$ for $1 \leq i \leq j \leq |w|$. A string w has period p if $w[i] = w[i + p]$ for $1 \leq i \leq |w| - p$. A string w is called *primitive* if w cannot be written as u^k, where k is a positive integer, $k \geq 2$.

Definition 1. *A* run *(also called a maximal repetition) of period p in a string w is a substring $w[i : j]$ such that:*

(1) $w[i : j]$ *has period p and satisfies $j - i + 1 \geq 2p$,*
(2) $w[i - 1] \neq w[i + p - 1]$ *(if $w[i - 1]$ is defined),*
$w[j + 1] \neq w[j - p + 1]$ *(if $w[j + 1]$ is defined), and*
(3) $w[i : i + p - 1]$ *is primitive.*

We denote the run $u = w[i : j]$ of period p in w by a triple $\langle i, j, p \rangle \in \mathcal{N}^3$ consisting of the begin position i, the end position j and the minimul period p of u. For a string w, we define that $Runs(w) = \{\langle i, j, p \rangle \mid w[i : j]$ is a run of period p in $w\}$. For instance, $Runs(\texttt{ababcbcca}) = \{\langle 1, 4, 2 \rangle, \langle 4, 7, 2 \rangle, \langle 7, 8, 1 \rangle\}$.

Theorem 2 ([11]). *Given a string w of length n, the set of all runs in string w can be calculated in $O(n)$ time.*

3 Easiness Results

At first, we give the definition of the problem.

Problem 3. Inferring strings of alphabet size k from runs (k-InvRunsEq).
Input: $S \subseteq \mathcal{N}^3$ and $n \in \mathcal{N}$.
Output: A string $w \in \Sigma_k^n$ such that $Runs(w) = S$ if any, and *None* otherwise.

Problem 4. Inferring consistent strings from runs (k-InvRunsSubset).
Input: $S \subset \mathcal{N}^3$ and $n \in \mathcal{N}$.
Output: A string $w \in \Sigma_k^n$ such that $Runs(w) \supseteq S$ if any, and *None* otherwise.

For convenience, ∞-InvRunsEq and ∞-InvRunsSubset denotes the problem for infinite alphabet. In this section, we show k-InvRunsEq can be solved in polynomial time if either $k \leq 2$ or $k = \infty$.

Theorem 5. *2-InvRunsEq is solvable in linear time.*

Proof. First we give a simple observation on the relationship between consecutive two symbols $w[k]$, $w[k+1]$ of string w and runs of period 1 in $Runs(w)$. If $w[k] = w[k+1]$ for some k, then the interval $(k, k+1)$ must be included in some run $\langle i, j, 1\rangle \in Runs(w)$ such that $i \leq k$ and $k+1 \leq j$. The converse is also holds. As a result, we can determine whether $w[k] = w[k+1]$ or not for every $1 \leq k \leq n-1$, by simply checking the intervals in all runs of period 1 in $Runs(w)$.

For binary alphabet $\Sigma_2 = \{\mathtt{a}, \mathtt{b}\}$, after choosing the first symbol $w[1]$ either `a` or `b` arbitrarily, we can uniquely determine the next symbol one by one consecutively, depending on whether $w[k] = w[k+1]$ or not, for each $k = 1, 2, \ldots, n-1$. It can be done in $O(n)$ time. The resulting string w is the only possible candidate for the solution of a given set S of runs, up to isomorphism. We can verify that $Runs(w) = S$ holds or not, in $O(n)$ time. If it holds, return w as a solution; otherwise, return *None*. □

Let $G = (V, E)$ be an unordered graph, where V is a set of nodes and $E \subseteq V \times V$ is a set of edges. A proper graph coloring on G is an assignment of colors to its nodes such that no two adjacent nodes receive the same color.

Problem 6. [k-Color problem]
Input: Graph G
Decide: The nodes of G can be colored with k colors such that no two nodes jointed by an edge have the same color.

We can regard the problem as identifying an equivalence relation over n elements $V = \{1, 2, \ldots, n\}$, so that it is consistent with the structural information of runs in S. Some constraints of equivalence and inequivalence are easily extracted from each run in S. For example, if $\langle 4, 7, 2\rangle$ is a run of string w of length 9, we know that $w[4] = w[6]$ and $w[5] = w[7]$ from condition (1) in Definition 1, as well as $w[3] \neq w[5]$ and $w[6] \neq w[8]$ from condition (2). Based on these observations, we will reduce the problem to the graph-coloring problem of a graph (V_S, E_S), where V_S is an equivalence class of V and E_S represents the inequivalence relations, as we will show the details below.

We define a binary relation R over V by

$$R = \{(k, k+p) \mid i \leq k \leq j-p \quad \text{for } \langle i, j, p\rangle \in S\},$$

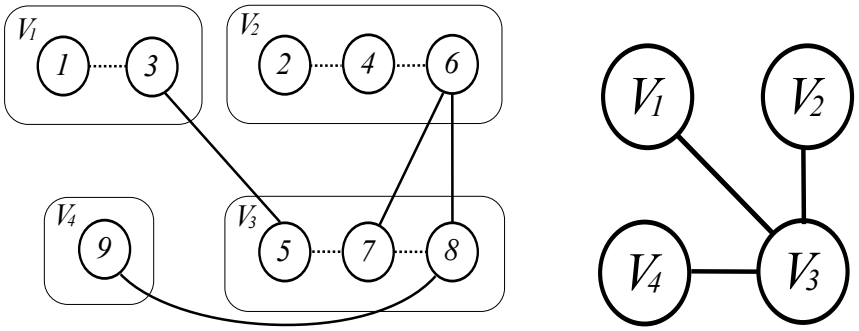

Figure 1. Graph G_S which represents $S = \{\langle 1, 4, 2\rangle, \langle 4, 7, 2\rangle, \langle 7, 8, 1\rangle\}$.

which stands for the equivalence $w[k] = w[k + p]$ due to the condition (1) in Definition 1. Let $R^{\equiv}$ be the reflexive transitive symmetric closure of R. That is, $R^{\equiv}$ is the smallest equivalence relation over V containing R. We define $V_S = \{[v]_{R^{\equiv}} \mid v \in V\}$, where $[v]_{R^{\equiv}}$ denotes the equivalence class of v in V with respect to $R^{\equiv}$.

We also define a binary relation D by

$$D = \{(i - 1, i - 1 + p) \mid 1 < i,\ \langle i, j, p\rangle \in S\} \cup \{(j + 1, j + 1 - p) \mid j < n,\ \langle i, j, p\rangle \in S\}.$$

D represents the inequalities $w[i - 1] \neq w[i - 1 + p]$ and $w[j + 1] \neq w[j + 1 - p]$ that come from condition (2) in Definition 1. We now define the set $E_S \subseteq V_S \times V_S$ of edges by

$$E_S = \{([v_1]_{R^{\equiv}}, [v_2]_{R^{\equiv}}) \mid (v_1, v_2) \in D\}.$$

Using a graph (V_S, E_S), we give a following theorem.

Theorem 7. ∞-InvRunsEq *and* ∞-InvRunsSubset *are solvable in* $O(n^2)$ *time.*

Proof. Since the equivalence relation $R^{\equiv}$ is disjoint with D, if G_S contains a self loop, then there exists no string w that satisfies $Runs(w) \supseteq S$. Otherwise, let ψ be a coloring of G_S. Since the number of colors is unbounded, it is straightforward to get such ψ; we may associate a different color to each node in V_S. By using ψ, we construct a string $w = \psi([1])\psi([2]) \ldots \psi([n])$, where we abbreviated $[i]_{R^{\equiv}}$ by $[i]$. It is not hard to verify that $Runs(w)$ satisfy the problem condition. We now consider the time complexity. The graph G_S can be constructed in $O(n^2)$ time, and we can check whether G_S contains a self loop or not in linear time. Therefore, ∞-InvRunsEq and ∞-InvRunsSubset are solvable in $O(n^2)$ time. □

Example 8. Let us consider an instance $S = \{\langle 1, 4, 2\rangle, \langle 4, 7, 2\rangle, \langle 7, 8, 1\rangle\}$ and 9 for ∞-InvRunsEq. We will find a string w of length 9 over infinite alphabet satisfying $Runs(w) = S$. We construct the graph $G_S = (V_S, E_S)$ from S as follows: Since $R = \{(1, 3), (2, 4), (4, 6), (5, 7), (7, 8)\}$, we have a set of nodes $V_S = \{V_1, V_2, V_3, V_4\}$ with $V_1 = \{1, 3\}$, $V_2 = \{2, 4, 6\}$, $V_3 = \{5, 7, 8\}$, $V_4 = \{9\}$. Moreover, since $D = \{(3, 5), (6, 7), (6, 8), (8, 9)\}$, we have a set of edges $E_S = \{(V_1, V_3), (V_2, V_3), (V_3, V_4)\}$. (Figure 1 shows G_S.) Since G_S contains no self loop, it is always colorable if the number of colors is unlimited. By considering a trivial coloring function ϕ such that $\phi(V_1) = \mathtt{a}$, $\phi(V_2) = \mathtt{b}$, $\phi(V_3) = \mathtt{c}$, and $\phi(V_4) = \mathtt{d}$, we get the string $w = \psi([1])\psi([2]) \ldots \psi([9]) = \mathtt{ababcbccd}$, and we can verify that $Runs(w) = S$, indeed.

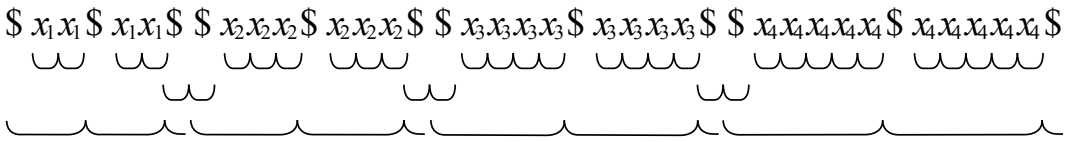

Figure 2. The string v representing the node set $V = \{1, 2, 3, 4\}$. Bows indicate all runs in v.

Remark that in Example 8, the graph G_S actually can be colored by 3 colors as $\phi(V_1) = \phi(V_4) = \mathtt{a}$, $\phi(V_2) = \mathtt{b}$, and $\phi(V_3) = \mathtt{c}$, so that we have another solution string $w = \mathtt{ababcbcca}$ over three symbols. In this way, k-InvRunsEq problem is reduced to k-Color problem. However, unfortunately, k-Color is NP-complete for finite $k \geq 3$ so that it is intractable. In the next section, we show an opposite reduction.

4 Hardness result for perfect input

In this section we show the NP-completeness of k-InvRunsEq for any fixed $k \geq 4$.

4.1 Instance transformation

Let $G = (V, E)$ be an input of the 3-Color problem, where $V = \{1, 2, \ldots, m\}$. We will construct an instance $\langle S, n\rangle$ for 4-InvRunsEq.

At first, we construct a string g over $\Delta = \{x_1, x_2, \ldots, x_m, \$\}$ which represents G as follows. For nodes V, let v be the string

$$v = v_1 v_2 \ldots v_k \ldots v_m,$$

where each substring v_k corresponding to node k is defined by

$$v_k = \$x_k^{(k+1)}\$x_k^{(k+1)}\$.$$

For example, Figure 2 shows the string v for the case $m = 4$.

Next we give the transformation which represents the edges E. For each $1 \leq k \leq n$, we define ℓ_k and r_k by

$$\ell_k = v_1 v_2 \ldots v_{k-1}\$x_k^{k+1},$$
$$r_k = x_k^{k+1}\$v_{k+1} \ldots v_m.$$

You see that ℓ_k is a prefix of v, and r_k is a suffix of v, so that $\ell_k\$r_k = v$. String e_{ij} which represents the edge $(i, j) \in E$ is defined as $e_{ij} = \ell_i r_j$.

Using the above gadgets, we give the function *enc* which encodes the graph G to the string:

$$g = B_1 B_2 \ldots B_{|E|+1},$$

where blocks B_t's are defined by $B_1 = vv\ell_{i_1}, B_{|E|+1} = r_{j_{|E|}}vv$, and for $t = 2 \ldots |E|$, $B_t = r_{j_{t-1}}vv\ell_{i_t}$ if t is odd $r_{j_{t-1}}vvv\ell_{i_t}$ otherwise. For example, Figure 3 shows the string g for the case $V = \{1, 2, 3, 4\}, E = \{(1, 2), (1, 3), (2, 3), (3, 4)\}$.

Critical points of the construction B_t are the following (see Figure 4):

(B1) Each B_t has period $|v|$ since ℓ_k (r_k, resp.) is a prefix (suffix, resp.) of v.

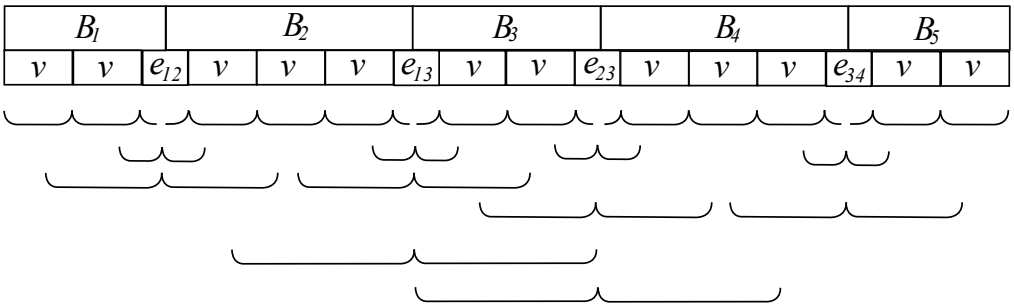

Figure 3. The string g representing the graph $G = (V = \{1, 2, 3, 4\}, E = \{(1, 2), (1, 3), (2, 3), (3, 4)\})$. Bows indicate some of runs in g.

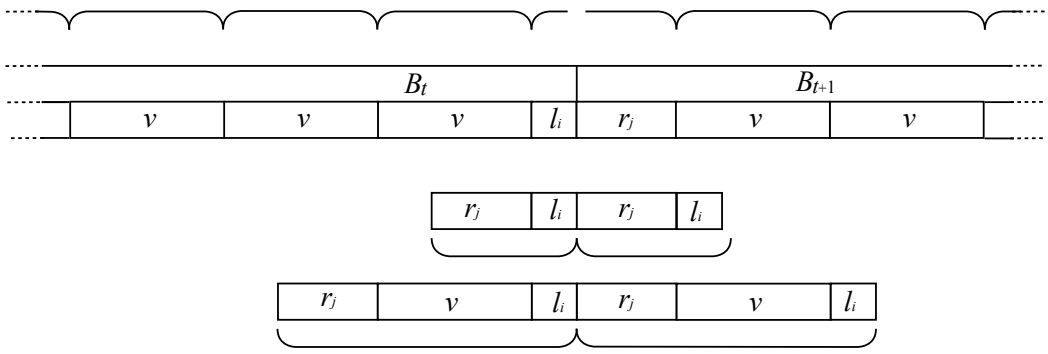

Figure 4. The substring of g represents the edge $E_t = (i, j)$. The top bows indicate that both B_t and B_{t+1} have period $|v|$, although these two periodicities are disconnected.

(B2) On the border between B_t and B_{t+1} there exists a run of period $|r_j v \ell_i|$ (the bottom bow in Figure 4), as well as a run of period $|r_j \ell_i|$ (the middle bow in Figure 4).

Therefore, string g is a concatenation of e_{i_k,j_k}'s and v's. Using this string, we can add some appropriate restrictions to the substitution φ depending on the input graph G. Finally, we calculate the set of runs and the length of string g, and output the instance $\langle Runs(g), |g| \rangle$ of the inverse runs problem.

4.2 Correctness of the reduction

For a given graph G, let $\langle S, n \rangle$ be the instance generated by the above reduction. By reconstructing a graph G_S from S and n in the same way in Section 3, we will show the relation between a coloring function for G and a string w of length n satisfying $Runs(w) = S$.

Let $V' = \{1, 2, \ldots, n\}$ be the set of positions of w. Because of the conditions (B1) and (B2), for any position $i \in V'$, there exists position $j \in \{1, \ldots, |v|\}$ such that $[i]_{R^{\equiv}} = [j]_{R^{\equiv}}$. We consider V_S as the quotient set of V by the equivalence relation $R^{\equiv}$, that is represented using the base string g as $V_S = \{V_{x_1}, V_{x_2}, \ldots, V_{x_{|V|}}, V_{\$}\}$ where $V_c = \{i \mid g[i] = c\}$ for $c \in \Delta$.

Next we consider the edges $E_S \subset V_S \times V_S$ yielded by the binary relation D representing inequivalences extracted from S. We show $E_S = \{(V_{x_i}, V_{x_j}) \mid (i,j) \in E\} \cup \{(V_{x_t}, V_\$), \mid 1 \le t \le |V|\}$ by enumerating the all runs in the string g. That is, we show the substring $g[i-1 : j+1]$ for each run $\langle i, j, p\rangle \in \mathit{Runs}(g)$:

- period 1
  Since $\$x_k^{k+1}\$$ is a substring of g, we have $(V_\$, V_{x_k}) \in E_S$ for each $k = 1 \dots m$.
- period 1
  Since $\$x_i^{i+1}x_j$ and $x_i x_j^{j+1}\$$ are substrings of g, we have
  $$\{(V_{x_j}, V_\$), (V_{x_i}, V_\$), (V_{x_i}, V_{x_j})\} \subset E_S$$
  for each $(i,j) \in E$.
- period $(k+2)$
  Since $\$\$x_k^{k+1}\$x_k^{k+1}\$\$$ is a substring of g, we have $(V_{x_k}, V_\$) \in E_S$ for each $k = 1 \dots m$.
- period $|r_j v^t \ell_j|$ for $t = 0, 1$
  Since $\$r_j v^t \ell_i r_j v^t \ell_i\$$ is a substring of g, we have
  $$\{(V_{x_j}, V_\$), (V_{x_i}, V_\$)\} \subset E_S$$
  for each $(i,j) \in E$,
- period $|r_j v v \ell_j|$
  Since $x_j(r_j v v \ell_i)^2\$$ and $\$(r_j v v \ell_i)^2 x_i$ are substrings of g, we have
  $$\{(V_{x_j}, V_\$), (V_{x_i}, V_\$), (V_{x_i}, V_{x_j})\} \subset E_S$$
  for each $(i,j) \in E$,

For this graph G_S, we have the following lemma.

Lemma 9. *The following three propositions are equivalent for any integer $k \ge 1$:*

(1) G is k-Colorable.
(2) G_S is $(k+1)$-Colorable.
(3) A string $w \in \Sigma_{k+1}^n$ exists such that $\mathit{Runs}(w) = S$.

Proof. (1) $\Leftrightarrow$ (2) From the definition of G_S, the induced subgraph $G' = (V_S - \{V_\$\}, E'_S)$ of G_S is isomorphic to G. Moreover $V_\$$ is connected with all the other nodes in V_S. Therefore $(k+1)$-coloring for G_S is a k-coloring for G'. It means (2) $\Rightarrow$ (1). On the other hand, by assigning a new color to the node $V_\$$, we obtain the $(k+1)$-coloring for G_S from k-coloring for G. It means (1) $\Rightarrow$ (2). See Figure 5.

(2) $\Rightarrow$ (3) Assume that $\psi : V_S \to \{1, \dots, k\}$ is a k coloring function of G. We can construct the substitution $\varphi : \{x_1, \dots, x_n, \$\} \to \{\mathtt{a}_1, \dots, \mathtt{a}_k, \$\}$ as $\varphi(c) = \mathtt{a}_{\psi(V_c)}$ for $c \in \Delta$. Since it satisfies $\mathit{Runs}(\varphi(g)) = S$, there exists a string $w = \varphi(g)$ such that $\mathit{Runs}(w) = S$.

(3) $\Rightarrow$ (2) Assume that string w satisfies $\mathit{Runs}(w) = S$. There exists a substitution $\varphi : \{x_1, \dots, x_n, \$\} \to \{\mathtt{a}_1, \dots, \mathtt{a}_k, \$\}$ and it holds $w = \varphi(g)$. Then, we can construct a coloring function ψ of G as follows; for each $c \in \{x_1, \dots, x_n\}$, $\psi(V_c) = i$ such that $\varphi(c) = \mathtt{a}_i$. Therefore G is k-colorable. □

Theorem 10. *4-*InvRunsEq *is NP-complete.*

Proof. From the above section, for any fixed $k \ge 1$, k-Color is polynomial time reducible to $(k+1)$-InvRunsEq. Since 3-Color is NP-Complete, 4-InvRunsEq is also NP-complete. □

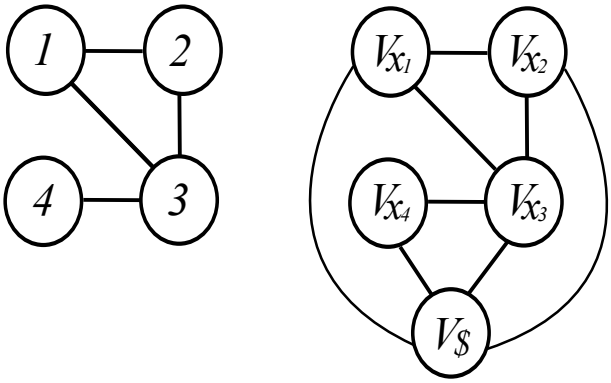

Figure 5. Input graph G and reconstructed graph G_S

5 Hardness result for Imperfect input

In this section we consider a variant of k-InvRunsEq. We consider the problem to infer a string from a set of runs that are given *imperfectly*.

We show the NP-completeness of 3-InvRunsSubset.

5.1 Instance reduction

Let $G = (V, E)$ be an input of the k-Color problem, where $V = \{1, 2, \ldots, m\}$. we construct the instance for the k-InvRunsSubset problem.

At first, we construct a string g over $\Delta' = \{x_1, x_2, \ldots x_m, \$_1, \$_2, \ldots, \$_m\}$ which represents G as follows: for nodes V, let v be the string,

$$v = x_1 x_1 \$_1 x_2 x_2 \$_2 \ldots x_m x_m \$_m.$$

The different point from k-InvRunsEq is that we cannot use a single symbol \$ as a separator. Instead, we use m separator symbols $\$_1, \$_2, \ldots, \$_m$, and we construct a string on Δ' and substitution $\varphi : \Delta' \to \{\mathtt{a}_1, \ldots, \mathtt{a}_k\}$.

For all $k = 2 \ldots m$, it satisfies $\varphi(x_{k-1}) \neq \varphi(\$_k) \neq \varphi(x_{k+1})$ and $\varphi(x_m) \neq \varphi(\$_m) \neq \varphi(x_1)$.

Since the alphabet size is three or more, for any substitution on $\{x_1, x_2, \ldots, x_m\}$, we can choose a substitution on $\{\$_1, \ldots, \$_m\}$. We can use the variables $\{\$_1, \ldots, \$_m\}$ as a separator.

Next we give the transformation which represents the edge of graph. For each $k = 1 \ldots m$, we define ℓ_k and r_k as follows:

$$\ell_k = x_1 x_1 \$_1 x_2 x_2 \$_2 \ldots x_{k-1} x_{k-1} \$_{k-1} x_k,$$
$$r_k = x_k \$_k x_{k+1} x_{k+1} \$_{k+1} \ldots x_m x_m \$_m.$$

You can see that ℓ_k is a prefix of v and r_k is a suffix v. The string e_{ij} represents the edge $(i, j) \in E$, where $e_{ij} = \ell_i r_j$.

Using the above gadgets, we give the definition of string which represents graph G by

$$g = B_1 B_2 \ldots B_{|E|+1},$$

where blocks B_t's are defined by $B_1 = vv\ell_{i_1}, B_{|E|+1} = r_{j_{|E|}} vv$, and for $t = 2 \ldots |E|$, $B_t = r_{j_{t-1}} vv \ell_{i_t}$.

By picking up runs from $Runs(g)$ we construct S as follows:

$$\begin{aligned} S = &\{\langle 3t-2, 3t-1, 1\rangle \mid 1 \le t \le m\} \\ &\cup \{\langle pos_t + 1, pos_{t+1}, |v|\rangle \mid 0 \le t \le |E|\} \\ &\cup \{\langle pos_t - p_t + 1, pos_t + p_t, p_t\rangle \mid 1 \le t \le |E|\}, \end{aligned}$$

where $pos_t = |B_1 B_2 \dots B_t|$ and $p_t = |r_{i_t} v \ell_{j_t}|$. Note that $S \subset Runs(g)$.

We output $\langle S, |g|\rangle$ as the instance of k-InvRunsSubset.

5.2 Correctness of the reduction

For a given a graph G, let $\langle S, n\rangle$ be the instance generated by the above reduction. By reconstructing a graph G_S from S and n in the same way in Section 3 and Section 4, we will show the relation between a coloring function for G and a string w of length n satisfying $Runs(w) \supseteq S$.

Let $V' = \{1, 2, \dots n\}$ be a set of positions. At first we construct V_S from using the equivalence relation R of S. Similar to the case of k-InvRunsEq, because the conditions (B1) and (B2) in Section 4, for any position $i \in V'$, there exists position $j \in \{1, \dots, |v|\}$ such that $[i]_{R^{\equiv}} = [j]_{R^{\equiv}}$. We consider V_S as the quotient set of V by the equivalence relation $R^{\equiv}$, that is represented using the base string g as $V_S = \{V_{x_1}, \dots, V_{x_m}, V_{\$_1}, \dots, V_{\$_m}\}$ where $V_c = \{i \mid g[i] = c\}$ for $c \in \Delta'$.

Next we consider the edges $E_S \subset V_S \times V_S$ yielded by the binary relation D representing inequivalences extracted from S. We show $E_S = \{(V_{x_i}, V_{x_j} \mid (i,j) \in E\} \cup \{(V_{x_t}, V_{\$_{t-1}}), (V_{x_t}, V_{\$_t}) \mid 1 \le t \le |E|\}$:

- Since $\langle 3t-2, 3t-1, 1\rangle \in S$, and the fact that $g[3t-3] = \$_{t-1}, g[3t-2] = x_t$ and $g[3t-1] = x_t, g[3t] = \$_t$, we have $(V_{\$_{t-1}}, V_{x_t}), (V_{\$_t}, V_{x_t}) \in E_S$ for $t = 1 \dots m$.
- Since $\langle pos_t + 1, pos_{t+1}, |v|\rangle \in S$, and the fact that $g[pos_t] = x_{j_t}, g[pos_t + |v|] = x_{i_t}$ and $g[pos_t + 1] = x_{i_t}, g[pos_t + 1 - |v|] = x_{j_t}$, we have $(V_{x_{i_t}}, V_{x_{j_t}}) \in E_S$ for $t = 1 \dots |E|$.
- Since $\langle pos_t - p_t + 1, pos_t + p_t, p_t\rangle \in S$ and the fact that $g[pos_t - p_t] = x_{j_t}, g[pos_t] = x_{i_t}$ and $g[pos_t + 1] = x_{j_t}$ and $g[pos_t + p_t + 1] = x_{i_t}$, we have $(V_{x_{i_t}}, V_{x_{j_t}}) \in E_S$ for $t = 1 \dots |E|$.

For this graph G_S, we have the following lemma.

Lemma 11. *The following three observation are equivalent for any fixed $k \ge 3$:*

(1) G is k-Colorable.
(2) G_S is k-Colorable.
(3) A string $w \in \Sigma_k^n$ exists such that $Runs(w) \supseteq S$.

Proof. (1) $\Leftrightarrow$ (2) From the definition of G_S, the induced subgraph $G' = (V'_S, E'_S)$ of G_S is isomorphic to G, where $V'_S = V_S - \{V_{\$_1}, \dots, V_{\$_m}\}$. Therefore k-coloring for G_S is a k-coloring for G'. It means (2) $\Rightarrow$ (1). On the other hand, Since each nodes $\{V_{\$_1}, \dots, V_{\$_m}\}$ is connected to two nodes in G_S, we can assign an another color for any fixed $k \ge 3$

Therefore we obtain the k-coloring for V_S from k-coloring for G, where $k \ge 3$. It means (1) $\Rightarrow$ (2). See Figure 6.

(2) $\Rightarrow$ (3) Assume that $\psi : V_S \to \{1 \dots k\}$ is k coloring function of G. We can construct the substitution $\varphi : \Delta' \to \{\mathtt{a}_1, \dots, \mathtt{a}_k\}$ as $\varphi(c) = \mathtt{a}_{\psi(V_c)}$ for $c \in \Delta'$. Since it satisfies $Runs(\varphi(g)) \supseteq S$, there exists the string $w = \varphi(g)$ such that $Runs(w) \supseteq S$.

(3) ⇒ (2) Assume that string w satisfies $Runs(w) \supseteq S$. there exists some substitution $\varphi : \Delta' \to \{\mathtt{a}_1, \ldots, \mathtt{a}_k\}$ and it holds $Runs(w) \supseteq Runs(\varphi(g))$. And then, we can construct the coloring function ψ of G as follows: For each $c \in \{x_1, \ldots, x_n\}$, $\psi(V_c) = i$ such that $\varphi(c) = \mathtt{a}_i$. Therefore G is k-colorable.

□

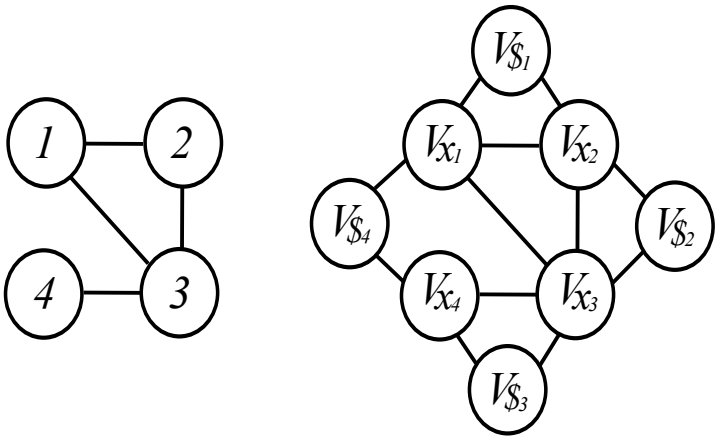

Figure 6. Input graph G and reconstructed graph G_S

Theorem 12. *3-*InvRunsSubset *is NP-complete.*

Proof. From above section, for any $k \geq 3$, k-Color is polynomial time reducible to k-InvRunsSubset. Since 3-Color is NP-Complete, 3-InvRunsSubset is also NP-complete. □

6 Conclusion

In this paper, we considered reverse problems of detecting all runs in a string. We showed that the computational complexity depends on the alphabet size of the output string. we also consider a variant of the problem, where the information on runs is incomplete. The result is summarized as the following table.

alphabet size k	k-InvRunsEq	k-InvRunsSubset
2	$O(n)$	open
3	open	NP-Complete
≥ 4	NP-Complete	NP-Complete
∞	$O(n^2)$	$O(n^2)$

It would be interesting to find out whether 3-InvRunsEq and 2-InvRunsSubset are NP-complete or in P. For 3-InvRunsEq, it is needed to improve the reduction from 3-Color without using the separator symbol "$". On the other hand, for 2-InvRunsSubset, it seems that we should develop a reduction from another NP-Complete problem.

References

1. H. Bannai, S. Inenaga, A. Shinohara, and M. Takeda: *Inferring strings from graphs ans arrays*, in Proc. 28th International Symposium on Mathematical Foundations of Computer Science (MFCS 2003), vol. 2747 of LNCS, Springer, 2003, pp. 208–217.
2. P. Baturo, M. Piatkowski, and W. Rytter: *The number of runs in Sturmian words*, in Proc. 13th International Conference on Implementation and Application of Automata (CIAA 2008), vol. 5148 of LNCS, Springer, 2008, pp. 252–261.
3. J. Clément, M. Crochemore, and G. Rindone: *Reverse engineering prefix tables*, in 26th International Symposium on Theoretical Aspects of Computer Science (STACS 2009), Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2009, pp. 289–300.
4. M. Crochemore and L. Ilie: *Maximal repetitions in strings.* J. Comput. Syst. Sci., 74 2008, pp. 796–807.
5. M. Crochemore, L. Ilie, and L. Tinta: *Towards a solution to the "runs" conjecture*, in Proc. 19th Annual Symposium on Combinatorial Pattern Matching (CPM 2008), vol. 5029 of LNCS, Springer, 2008, pp. 290–302.
6. M. Crochemore, W. Rytter, and L. Ilie: *Repetitions in strings: Algorithms and combinatorics.* Theoretical Computer Science, 410(50) 2009, pp. 5227–5235.
7. J. Duval, T. Lecroq, and A. Lefevre: *Border array on bounded alphabet*, in Proc. Prague Stringology Conference 2002, 2002, pp. 28–35.
8. F. Franek, S. Gao, W. Lu, P. J. Ryan, W. F. Smyth, Y. Sun, and L. Yang: *Verifying a border array in linear time.* J. Comb. Math. Comb. Comput, 42 2000, pp. 223–236.
9. M. Giraud: *Not so many runs in strings*, in Proc. 2nd International Conference on Language and Automata Theory and Applications (LATA 2008), vol. 5196 of LNCS, Springer, 2008, pp. 245–252.
10. T. I, S. Inenaga, H. Bannai, and M. Takeda: *Counting parameterized border arrays for a binary alphabet*, in Proc. 3nd International Conference on Language and Automata Theory and Applications (LATA 2009), vol. 5457 of LNCS, Springer, 2009, pp. 422–433.
11. R. Kolpakov and G. Kucherov: *Finding maximal repetitions in a word in linear time*, in Proc. 40th Annual Symposium on Foundations of Computer Science (FOCS 1999), 1999, pp. 596–604.
12. S. J. Puglisi, J. Simpson, and W. F. Smyth: *How many runs can a string contain?* Theoretical Computer Science, 401(1–3) 2008, pp. 165–171.
13. W. Rytter: *The number of runs in a string: Improved analysis of the linear upper bound*, in Proc. 23rd Annual Symposium on Theoretical Aspects of Computer Science (STACS 2006), vol. 3884 of LNCS, Springer, 2006, pp. 184–195.
14. W. Rytter: *The number of runs in a string.* Inf. Comput., 205(9) 2007, pp. 1459–1469.
15. J. Simpson: *Modified padovan words and the maximum number of runs in a word.* The Australasian Journal of Combinatorics, 42 2010, pp. 129–145.