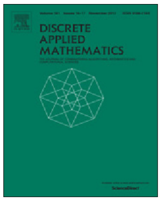
Contents lists available at ScienceDirect

Discrete Applied Mathematics

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



Prefix and suffix reversals on strings[☆],[☆☆]

Guillaume Fertin, Loïc Jankowiak, Géraldine Jean [*]

Laboratoire des Sciences du Numérique de Nantes, UMR CNRS 6004, Université de Nantes, 2 rue de la Houssinière, 44322 Nantes Cedex 3, France

ARTICLE INFO

Article history:
Received 29 January 2016
Received in revised form 2 April 2017
Accepted 26 July 2017
Available online xxxx

Keywords:
Prefix and suffix reversals
Strings
Algorithmic complexity
Approximation algorithms
Fixed-parameterized tractability

ABSTRACT

The Sorting by Prefix Reversals problem consists in sorting the elements of a given permutation π using a minimum number of prefix reversals, i.e. reversals that always involve the leftmost element of π. A natural extension of this problem is to consider strings, in which any letter may appear several times, rather than permutations. In strings, three different types of problems arise: *grouping* (given a string S, transform it so that all identical letters are consecutive), *sorting* (a constrained version of grouping, in which the target string must be lexicographically ordered) and *rearranging* (given two strings S and T, transform S into T). In this paper, we study these three problems, under an algorithmic viewpoint, in the setting where two operations, rather than one, are allowed: namely, *prefix and suffix* reversals — where a suffix reversal must always involve the rightmost element of the string. We first compare the "prefix reversals only" case to our case, before presenting a series of algorithmic results on these three problems concerning polynomiality, constant ratio approximation algorithms, NP-hardness and fixed-parameterized tractability. These results depend on the size k of the alphabet on which the strings are built, with a particular focus on small-sized alphabet instances (i.e., $k = O(1)$) and big-sized alphabet instances (i.e. $n - k = O(1)$, where n is the length of the input string(s)).

1. Introduction and notations

A usual task in comparative genomics consists in comparing pairs of genomes, in order to define how (dis)similar they are. Usually, a genome is modeled as a permutation π or a string S, and the pairwise comparison of genomes is done by computing a distance between them. Such a distance is generally defined based on so-called "rearrangements", which are large scale operations involving contiguous segments of π (resp. S). The most famous example of such rearrangements is *reversals* (see e.g. [3,9]), which take a contiguous segment of π (resp. S), reverse its order, and reincorporate this new segment at the same location. More generally, given a set $\mathcal{S}$ of possible rearrangements, the distance between two permutations π and π' (resp. two strings S and T) is defined as the minimum number of operations from $\mathcal{S}$ that are required to obtain π' starting from π (resp. to obtain T starting from S). Many studies are concerned with the special case where $|\mathcal{S}| = 1$, i.e. only one type of rearrangement is authorized. There is a very abundant literature on the subject, and we refer the reader to [7] for a survey on the topic. In this paper, our object of study is strings, a generalization of permutations, in which any letter may occur several times. The set of rearrangements we authorize here is composed of two operations: *prefix reversals*,

[☆] Supported by GRIOTE project, funded by Région Pays de la Loire.

[☆☆] A preliminary version of this work appeared in Proceedings of the 22nd edition of the International Symposium on String Processing and Information Retrieval (SPIRE 2015) Fertin et al. (2015) [6].

[*] Corresponding author.
E-mail addresses: guillaume.fertin@univ-nantes.fr (G. Fertin), loic.jankowiak@etu.univ-nantes.fr (L. Jankowiak), geraldine.jean@univ-nantes.fr (G. Jean).

http://dx.doi.org/10.1016/j.dam.2017.07.031
0166-218X/© 2017 Elsevier B.V. All rights reserved.

which necessarily involve the first letter of the string, and *suffix reversals*, which necessarily involve the last letter of the string. When dealing with strings, three types of problems are usually considered: *grouping* (given a string S, transform it so that all identical letters are consecutive), *sorting* (a constrained version of grouping, in which the target string must be lexicographically ordered) and *rearranging* (given two strings S and T, transform S into T). In this paper, we study these three problems, under an algorithmic viewpoint.

Terminology and basic properties.

Throughout the paper, and otherwise stated, any string S is a sequence of letters built on an alphabet Σ of cardinality k, is *fully k-ary* (i.e. each letter of Σ appears at least once in S), and is of length n. A letter l in S is said to be *lonely* whenever it appears only once in S, and *abundant* otherwise. When $k = n$, i.e. in the special case of permutations, the identity string is denoted I_n. Two strings S and T are said to be *compatible* if they contain the same multiset of letters. If S is the concatenation of two strings S_1 and S_2, it is denoted $S = S_1.S_2$, or S_1S_2 if clear from the context. If S is the concatenation of p occurrences of the same string X, this will be denoted $S = X^p$. For any $1 \leq i \leq j \leq n$, $S[i..j]$ denotes the substring of S starting at position i and ending at position j. A *block* in S is a maximal substring in which all letters are equal. Let $b(S)$ be the number of blocks in S. Given two compatible strings S and T, we let $b_{max}(S, T) = \max\{b(S), b(T)\}$, which we will write b_{max} in the rest of the paper if clear from the context. Given two integers $1 \leq i \leq j \leq n$, a *reversal* $\rho(i, j)$ of a string S is the operation that reverses $S[i..j]$ and reincorporates it at the same location, i.e. $S = s_1s_2 \ldots s_{i-1}\underline{s_is_{i+1} \ldots s_{j-1}s_j}s_{j+1} \ldots s_n$ is transformed into $S' = s_1s_2 \ldots s_{i-1}\underline{s_js_{j-1} \ldots s_{i+1}s_i}s_{j+1} \ldots s_n$. For readability, the substring $S[i..j]$ to be reversed will sometimes be underlined, as shown in the above example. If a reversal $\rho(1, n)$ is applied to a string S, we will say it is a *complete* reversal, and the resulting string will be written $\overline{S}$. A *prefix reversal* (resp. *suffix reversal*) is a reversal $\rho(1, j)$ with $1 \leq j \leq n$ (resp. $\rho(i, n)$ with $1 \leq i \leq n$), i.e. the first (resp. last) letter of S is involved in the reversal. We will often use the terms *p-reversal* (resp. *s-reversal*) to denote a reversal that is prefix (resp. suffix).

We say that a fully k-ary string S is *grouped* if it contains exactly k blocks, i.e. if, for any letter l in Σ, all occurrences of l are consecutive. We say that S is *sorted* if S is both grouped and lexicographically ordered. Given a string S, sort(S) will denote its *corresponding sorted string*, i.e. the unique string compatible with S that is lexicographically ordered. Given two compatible strings S and T, and given a set $\mathcal{S}$ of possible operations on strings, the *distance* between S and T is defined as the minimum number of operations from $\mathcal{S}$ that are required to obtain T, starting from S. In this paper, the two operations we allow are prefix reversals and suffix reversals. We are now ready to formally describe the three problems we are interested in, in their decision version.

Grouping Strings by Prefix and Suffix Reversals (GPSR)
Instance : A fully k-ary string S, an integer r.
Question : Is there a sequence of at most r p- and s-reversals that transforms S into a grouped string?

Sorting Strings by Prefix and Suffix Reversals (SPSR)
Instance : A fully k-ary string S, an integer r.
Question : Is there a sequence of at most r p- and s-reversals that transforms S into its corresponding sorted string sort(S)?

Rearranging Strings by Prefix and Suffix Reversals (RPSR)
Instance : Two compatible fully k-ary strings S and T, an integer r.
Question : Is there a sequence of at most r p- and s-reversals that transforms S into T?

Given a string S, the least integer r for which the answer to GPSR is positive is called the *grouping distance*, and is denoted $d^g_{PS}(S)$. Similarly, the sorting and rearranging distances are denoted $d^s_{PS}(S)$ and $d^r_{PS}(S, T)$. Note that when $k = n$ (i.e. strings are in fact permutations), the grouping problem is trivial (S is already grouped). Moreover, the sorting and rearranging problems become equivalent (up to a relabeling of the input permutations in the case of rearranging).

Note also that when dealing with the grouping (resp. sorting) problem, it is easy to show, by contradiction, that there always exists an optimal grouping (resp. sorting) algorithm for S that never cuts a block (otherwise, a more efficient algorithm would exist). Thus, when we look at GPSR or SPSR, the lengths of the blocks play no particular role, and it will thus be more convenient to assume that each block of S is of length 1. We will call such a string a *normalized* string. Consequently, any (prefix or suffix) reversal whose effect is to make two identical letters consecutive reduces the length of a normalized string by 1, and such a reversal is called a *1-flip*. A reversal that is not a 1-flip is called a *0-flip*.

Known results.

Comparing pairs of strings by means of their rearrangement distance has mostly been studied in the case where only one type of operation is allowed, such as reversals [5,14,1], transpositions [5,14] or translocations [4]. When prefix reversals only are allowed, the main results are the following: first, the grouping problem is polynomial when $k = 2$ and $k = 3$, and admits

Please cite this article in press as: G. Fertin, et al., Prefix and suffix reversals on strings, Discrete Applied Mathematics (2017), http://dx.doi.org/10.1016/j.dam.2017.07.031.