Theoretical Computer Science 525 (2014) 68–79

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Order-preserving matching

Jinil Kim [a], Peter Eades [b], Rudolf Fleischer [c,d], Seok-Hee Hong [b], Costas S. Iliopoulos [e,f], Kunsoo Park [a,*], Simon J. Puglisi [g], Takeshi Tokuyama [h]

[a] Department of Computer Science and Engineering, Institute of Computer Technology, Seoul National University, Seoul, Republic of Korea
[b] School of Information Technologies, University of Sydney, Australia
[c] SCS and IIPL, Fudan University, Shanghai, China
[d] Department of Applied Information Technology, German University of Technology in Oman, Muscat, Oman
[e] Department of Informatics, King's College London, London, United Kingdom
[f] Digital Ecosystems and Business Intelligence Institute, Curtin University, Australia
[g] Department of Computer Science, University of Helsinki, Finland
[h] Graduate School of Information Sciences, Tohoku University, Japan

ARTICLE INFO

Keywords:
String matching
Numeric string
Order relation
Multiple pattern matching
KMP algorithm
Aho–Corasick algorithm

ABSTRACT

We introduce a new string matching problem called *order-preserving matching* on numeric strings, where a pattern matches a text if the text contains a substring of values whose relative orders coincide with those of the pattern. Order-preserving matching is applicable to many scenarios such as stock price analysis and musical melody matching in which the order relations should be matched instead of the strings themselves. Solving order-preserving matching is closely related to the *representation of order relations* of a numeric string. We define the *prefix representation* and the *nearest neighbor representation* of the pattern, both of which lead to efficient algorithms for order-preserving matching. We present efficient algorithms for single and multiple pattern cases. For the single pattern case, we give an $O(n \log m)$ time algorithm and optimize it further to obtain $O(n + m \log m)$ time. For the multiple pattern case, we give an $O(n \log m)$ time algorithm.

1. Introduction

String matching is a fundamental problem in computer science and has been extensively studied. Sometimes a string consists of numeric values instead of characters in an alphabet, and we are interested in some *trends* in the text rather than specific patterns. For example, in a stock market, analysts may wonder whether there is a period when the share price of a company dropped consecutively for 10 days and then went up for the next 5 days. In such cases, the changing patterns of share prices are more meaningful than the absolute prices themselves. Another example is melody matching between two musical scores. A musician may be interested in whether her new song has a melody similar to well-known songs. As many variations are possible in a melody where the relative heights of pitches are preserved but the absolute pitches can be changed, it would be reasonable to match relative pitches instead of absolute pitches to find similar musical phrases.

An *order-preserving matching* can be helpful in both examples, because a pattern is matched with the text if the text contains a substring of values whose relative orders coincide with those of the pattern. For example, in Fig. 1, pattern

* Corresponding author. Tel.: +82 2 880 1828; fax: +82 2 885 3141.
E-mail addresses: jikim@theory.snu.ac.kr (J. Kim), peter.eades@sydney.edu.au (P. Eades), rudolf@fudan.edu.cn (R. Fleischer), seokhee.hong@sydney.edu.au (S.-H. Hong), c.iliopoulos@kcl.ac.uk (C.S. Iliopoulos), kpark@theory.snu.ac.kr (K. Park), puglisi@cs.helsinki.fi (S.J. Puglisi), tokuyama@dais.is.tohoku.ac.jp (T. Tokuyama).

0304-3975/$ – see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.tcs.2013.10.006

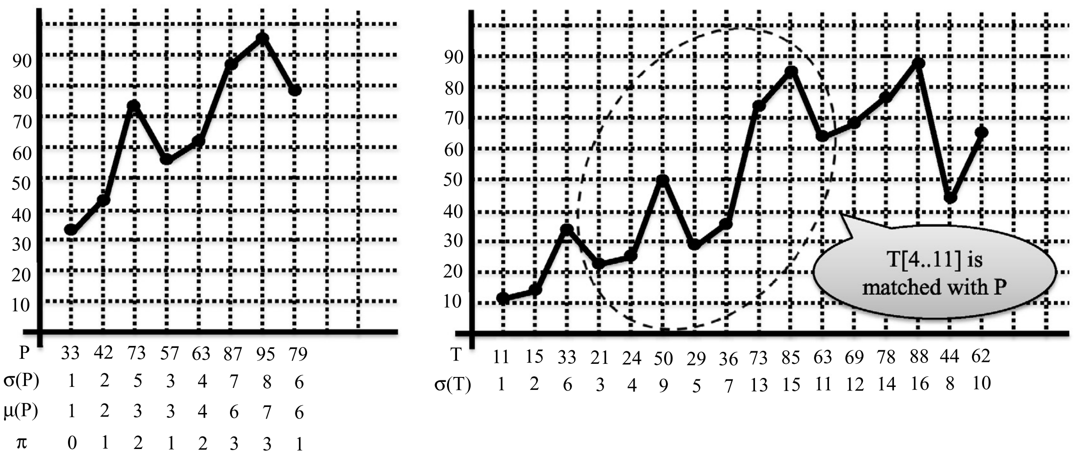

Fig. 1. Example of pattern and text.

$P = (33, 42, 73, 57, 63, 87, 95, 79)$ is matched with text T since the substring $(21, 24, 50, 29, 36, 73, 85, 63)$ in the text has the same relative orders as the pattern. In both strings, the first characters 33 and 21 are the smallest, the second characters 42 and 24 are the second smallest, the third characters 73 and 50 are the 5-th smallest, and so on. If we regard prices of shares, or absolute pitches of musical notes, as numeric characters of the strings, both examples above can be modeled as order-preserving matching.

Solving order-preserving matching is closely related to *representations of order relations* of a numeric string. If we replace each character in a numeric string by its rank in the string, then we can obtain a (natural) representation of order relations. But this natural representation is not amenable to developing efficient algorithms because the rank of a character depends on the substring in which the rank is computed. Hence, we define the *prefix representation* of order relations, which leads to an $O(n \log m)$ time algorithm for order-preserving matching, where n and m are the lengths of the text and the pattern, respectively. Surprisingly, however, there is an even better representation, called the *nearest neighbor representation*, with which we were able to develop an $O(n + m \log m)$ time algorithm.

In this paper, we define a new class of string matching problem, called *order-preserving matching*, and present efficient algorithms for single and multiple pattern cases. For the single pattern case, we propose an $O(n \log m)$ algorithm based on the Knuth–Morris–Pratt (KMP) algorithm [18,20], and optimize it further to obtain $O(n + m \log m)$ time. For the multiple pattern case, we present an $O(n \log m)$ algorithm based on the Aho–Corasick algorithm [1].

Related work: Norm matching and (δ, γ)-matching have been studied to search for similar patterns of numeric strings. In norm matching [8,25,2,28], each text substring and the pattern is matched if the L_p distance is less than the predefined value for some given p. In (δ, γ)-matching [12,19,16,15,23,24,26], two parameters δ and γ are given, and two numeric strings of the same length are matched if the maximum difference of the corresponding characters is at most δ and the total sum of differences is at most γ. Several variants were studied to allow for *don't care* symbols [17], transposition-invariant [23] and gaps [13,14,21]. On the other hand, some generalized matching problems such as parameterized matching [10,7], less than matching [6], swapped matching [3,27], overlap matching [5], and function matching [4,9] are studied extensively where *matching* relations are defined differently so that some properties of two strings are matched instead of exact matching of characters. However, none of this prior work addresses the *order relations*, which we focus on in this paper.

2. Problem formulation

2.1. Notations

Let Σ denote the set of numbers such that a comparison of two numbers can be done in constant time, and let Σ^* denote the set of strings over the alphabet Σ. Let $|x|$ denote the length of a string x. A string x is described by either a concatenation of characters $x[1] \cdot x[2] \cdot \cdots \cdot x[|x|]$ or as a sequence of characters $(x[1], x[2], \ldots, x[|x|])$ interchangeably. For a string x, let a substring $x[i..j]$ be $(x[i], x[i+1], \ldots, x[j])$ and the prefix x_i be $x[1..i]$. The rank of a character c in string x is defined as $rank_x(c) = 1 + |\{i \colon x[i] < c \text{ for } 1 \leqslant i \leqslant |x|\}|$. For simplicity, we assume that all the numbers in a string are *distinct*. When a number occurs more than once in a string, we can extend our character definition to a pair (character, index) so that the characters in the string become distinct.

2.2. Natural representation of order relations

For a string x, the *natural representation* of the order relations can be defined as $\sigma(x) = rank_x(x[1]) \cdot rank_x(x[2]) \cdot \cdots \cdot rank_x(x[|x|])$.