Information Processing Letters 115 (2015) 397–402

Contents lists available at ScienceDirect

Information Processing Letters

www.elsevier.com/locate/ipl

A fast algorithm for order-preserving pattern matching ☆

Sukhyeun Cho [a], Joong Chae Na [b], Kunsoo Park [c], Jeong Seop Sim [a,*]

[a] Department of Computer and Information Engineering, Inha University, Incheon 402-751, South Korea
[b] Department of Computer Science and Engineering, Sejong University, Seoul 143-747, South Korea
[c] School of Computer Science and Engineering, Seoul National University, Seoul 151-742, South Korea

ARTICLE INFO

Article history:
Received 8 April 2014
Received in revised form 27 September 2014
Accepted 29 October 2014
Available online 1 November 2014
Communicated by Tsan-sheng Hsu

Keywords:
Analysis of algorithms
Order-preserving pattern matching
Order-isomorphism
Horspool algorithm
KMP algorithm

ABSTRACT

Given a text T and a pattern P, the order-preserving pattern matching (OPPM) problem is to find all substrings in T which have the same relative orders as P. The OPPM has been studied in the fields of finding some patterns affected by relative orders, not by their absolute values. In this paper, we present a method of deciding the order-isomorphism between two strings even when there are same characters. Then, we show that the bad character rule of the Horspool algorithm for generic pattern matching problems can be applied to the OPPM problem and we present a space-efficient algorithm for computing shift tables for text search. Finally, we combine our bad character rule with the KMP-based algorithm to improve the worst-case running time. We give experimental results to show that our algorithm is about 2 to 6 times faster than the KMP-based algorithm in reasonable cases.

1. Introduction

Given a text T and a pattern P, the order-preserving pattern matching (OPPM for short) problem is to find all substrings in T which have the same relative orders as P. For example, when $P = (35, 40, 23, 40, 40, 28, 30)$ and $T = (10, 20, 15, 28, 32, 12, 32, 32, 20, 25, 15, 25)$ are given, P has the same relative orders as the substring $T' = (28, 32, 12, 32, 32, 20, 25)$ of T. In T' (resp. P), the first character 28 (resp. 35) is the 4-th smallest, the second character 32 (resp. 40) is the 5-th smallest, the third character 12 (resp. 23) is the smallest, and so on. See Fig. 1. The OPPM has been studied in the fields of finding some patterns affected by relative orders, not by their absolute values. For example, it can be applied to time series analysis like share prices on stock markets and to musical melody matching of two musical scores [2].

Recently, several results were presented on the OPPM problem. For the OPPM problem, the order-isomorphism must be defined. Kim et al. [2] defined the order-isomorphism as the equivalence of permutations converted from strings with an assumption that all the characters in a string are distinct. Given T ($|T| = n$) and P ($|P| = m$), they proposed an algorithm for the OPPM problem running in $O(n + m \log m)$ time based on the Knuth–Morris–Pratt (KMP) algorithm [3]. Meanwhile, Kubica et al. [4] defined the order-isomorphism as the equivalence of all relative orders between two strings, and presented a method of deciding the order-isomorphism of two strings even when there are same characters. They independently proposed an algorithm for the OPPM problem based on the KMP algorithm running in $O(n + m \log m)$ time for a general alphabet and $O(n + m)$ time for an integer alphabet whose characters can be sorted in linear time. More recently, Crochemore et al. [5] introduced order-preserving suffix trees, and they suggested an algorithm finding all occurrences of P in T running in $O(m + z)$ time where z is the number of occurrences.

☆ A preliminary version of this paper appeared in COCOA 2013 [1].

* Corresponding author. Tel.: +82 32 860 7455.
E-mail address: jssim@inha.ac.kr (J.S. Sim).

http://dx.doi.org/10.1016/j.ipl.2014.10.018
0020-0190/

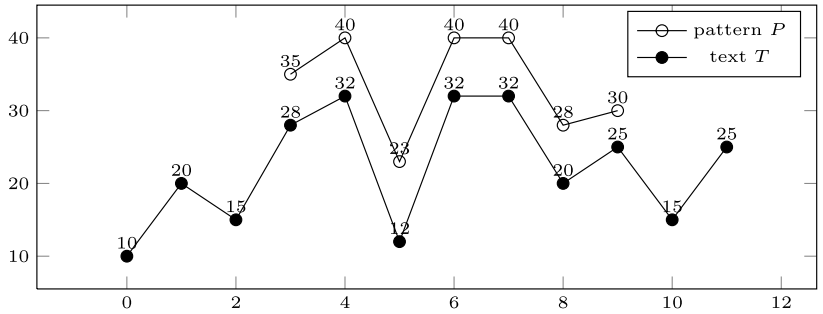

Fig. 1. An OPPM example for $P = (35, 40, 23, 40, 40, 28, 30)$ and $T = (10, 20, 15, 28, 32, 12, 32, 32, 20, 25, 15, 25)$.

In this paper, we propose fast algorithms for the OPPM problem based on the Horspool algorithm [6–8]. Experimental results show that our algorithms are about 2 to 6 times faster than the KMP-based algorithm in reasonable cases. Our contributions are as follows.

- We present a method of deciding the order-isomorphism between two strings even when there are same characters. We show that Kubica et al.'s method [4] may decide it incorrectly when there are same characters.
- We show that the bad character rule can be applied to the OPPM problem by defining a group of characters as one character. Kim et al. [2] mentioned the hardness of applying the Boyer–Moore algorithm [9] to the OPPM problem. The good suffix rule could be well-defined but the bad character rule could not be directly applied to the OPPM problem.
- We present a space-efficient algorithm computing the shift table for text search based on a factorial number system. Let q be a size of the group of characters and $|\Sigma|$ be the size of an alphabet. Then, our algorithm uses $O(q!)$ space for the shift table while the algorithms of [6,7] for the generic pattern matching problem use $O(|\Sigma|^q)$ space for the shift table.
- We also show that our bad character rule can be combined with the KMP-based algorithm to improve the worst-case running time of [1]. The combined algorithm guarantees $O(n + m \log m)$ time for a general alphabet and $O(n + m)$ time for an integer alphabet in the worst case when q is a constant.

2. Preliminaries

Let Σ denote an alphabet and $\sigma = |\Sigma|$. Let $|x|$ denote the length of a string x. A string x is described by a sequence of characters $(x[0], x[1], \ldots, x[|x| - 1])$.

Now, we formally define the order-isomorphism and the order-preserving pattern matching problem. Two strings x and y of the same length over Σ are called *order-isomorphic*, written $x \approx y$, if

$$x[i] \le x[j] \quad \Leftrightarrow \quad y[i] \le y[j] \quad \text{for } 0 \le i, j < |x| \text{ [4].}$$

If two strings x and y are not order-isomorphic, we write $x \not\approx y$. Given a text $T[0..n-1]$ and a pattern $P[0..m-1]$, we say that T *matches* P at position i if $T[i-m+1..i] \approx P$. In the previous example shown in Fig. 1, T matches P at position 9 because $T[3..9] \approx P$. The *order-preserving pattern matching problem* is to find all positions of T matched with P.

Let us define a *prefix table* μ_x of string x:

$$\mu_x[i] = \left|\left\{j : x[j] \le x[i] \text{ for } 0 \le j < i\right\}\right|.$$

See Table 1 for an example.

Table 1
μ_P, $LMax_P$, $LMin_P$, π_P for $P = (35, 40, 23, 40, 40, 28, 30)$.

i	0	1	2	3	4	5	6
$P[i]$	35	40	23	40	40	28	30
$\mu_P[i]$	0	1	0	3	4	1	2
$LMax_P[i]$	−1	0	−1	1	3	2	5
$LMin_P[i]$	−1	−1	0	1	3	0	0
$\pi_P[i]$	0	1	1	2	1	1	2

Lemma 1. *For two strings x and y, if $x \approx y$, then $\mu_x = \mu_y$.*

Proof. By the assumption that $x \approx y$, $x[i] \le x[j] \Leftrightarrow y[i] \le y[j]$ for $0 \le i < j < |x|$. Hence, $\mu_x = \mu_y$. □

Lemma 2. *Assume that $x[0..t] \approx y[0..t]$. For all $0 \le i, j \le t$, if $x[i] < x[j]$, then $y[i] < y[j]$, and if $x[i] = x[j]$, then $y[i] = y[j]$.*

Proof. We first prove by contradiction the first proposition (when $x[i] < x[j]$). Suppose that $y[i] \ge y[j]$. Then, by the definition of order-isomorphism, $x[i] \ge x[j]$, which contradicts the assumption that $x[i] < x[j]$.

Next, consider the case when $x[i] = x[j]$. Then, since $x[i] \le x[j]$, $y[i] \le y[j]$ by the definition of order-isomorphism. Moreover, since $x[j] \le x[i]$, $y[j] \le y[i]$. Since $y[i] \le y[j]$ and $y[j] \le y[i]$, $y[i] = y[j]$. □

Kubica et al. [4] used location tables called *LMax* and *LMin* for the order information of prefixes of P: Given a string x, for $i = 0, \ldots, |x| - 1$,

$$LMax_x[i] = j$$
$$\text{if } x[j] = \max\left\{x[k] : k \in [0, i-1], x[k] \le x[i]\right\} \quad \text{and}$$
$$LMin_x[i] = j$$
$$\text{if } x[j] = \min\left\{x[k] : k \in [0, i-1], x[k] \ge x[i]\right\}.$$