Theoretical Computer Science 710 (2018) 35–43

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

An $O(n^2 \log m)$-time algorithm for the boxed-mesh permutation pattern matching problem ☆

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

[a] Department of Computer Science and Engineering, Inha University, Incheon 22212, South Korea
[b] Department of Computer Science and Engineering, Sejong University, Seoul 05006, South Korea

ARTICLE INFO

Article history:
Received 16 June 2016
Received in revised form 27 January 2017
Accepted 14 February 2017
Available online 2 March 2017

Keywords:
Permutation pattern matching
Order-isomorphism
Boxed-mesh pattern

ABSTRACT

Given a text T of length n and a pattern P of length m over a numeric alphabet Σ, the boxed-mesh permutation pattern matching problem is to find all boxed-subsequences of T whose relative order between all characters is the same as that of P. In this paper, we propose an $O(n^2 \log m)$-time algorithm for the boxed-mesh permutation pattern matching problem based on interesting properties of boxed subsequences, using preprocessed information on P and order-statistic trees.

1. Introduction

Permutation patterns have been studied vigorously due to their applications in time series data analysis [2,4,7,10–13]. Given a text T and a pattern P over a numeric alphabet Σ, the permutation pattern matching problem[1] is to find every subsequence of T whose relative order between all characters (numbers) is the same as that of P [3,5]. For example, when $P = (5, 3, 4, 8, 9, 6, 7)$ and $T = (10, 6, 2, 7, 15, 16, 12, 19, 13, 11, 3)$ are given, P has the same relative order as those of two subsequences of T, i.e., $T' = (10, 6, 7, 15, 16, 12, 13)$ and $T'' = (10, 2, 7, 15, 16, 12, 13)$. The first character '10' in T' (resp. in T'') is the 3rd smallest character as '5' in P, the second character '6' in T' (resp. '2' in T'') is the smallest character as '3' in P, and so on.

The above (classical) permutation pattern matching problem is shown to be NP-hard [3] and other various types of permutation patterns and corresponding problems have been introduced and have been studied. For example, when certain characters in the pattern should be adjacent in the text, we call it *vincular permutation pattern matching* problem which is also shown to be NP-hard [5]. When all the characters in the pattern should be adjacent in the text, we call it *order-preserving pattern matching* problem (also known as *consecutive permutation pattern matching* problem). For the order-preserving pattern matching problem, Kubica et al. [13] and Cho et al. [7] gave $O(n + m \log m)$-time algorithms for a general alphabet and gave $O(n + m)$-time algorithms for an integer alphabet when $|T| = n$ and $|P| = m$.

In this paper, we focus on the boxed-mesh permutation pattern matching (BPPM for short) problem, another variation of permutation pattern matching problem. In BPPM, the ith character c in a (numeric) string x can be represented as

☆ A preliminary version of this paper appeared in CPM 2015 [8].

* Corresponding author.
E-mail addresses: csukhyeun@inha.edu (S. Cho), jcna@sejong.ac.kr (J.C. Na), jssim@inha.ac.kr (J.S. Sim).

[1] Note that this terminology was also used in different problems by other group.

http://dx.doi.org/10.1016/j.tcs.2017.02.022
0304-3975/

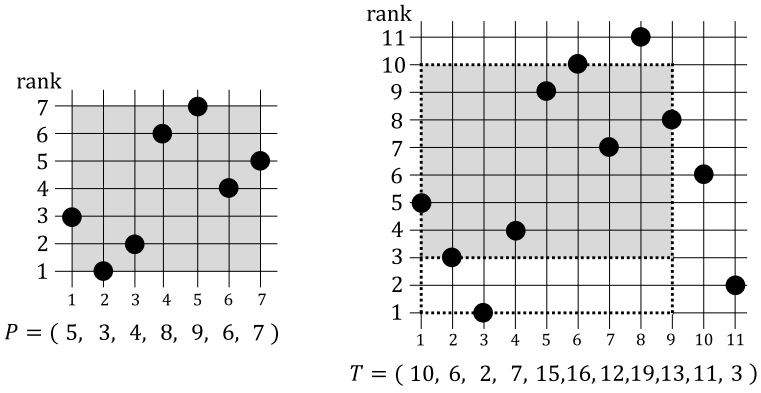

Fig. 1. Representations of $P = (5, 3, 4, 8, 9, 6, 7)$ and $T = (10, 6, 2, 7, 15, 16, 12, 19, 13, 11, 3)$ in two-dimensional planes, where the x-axis is labeled with indexes of characters and the y-axis is labeled with ranks of characters.

a point of coordinate (i, c) on a two-dimensional plane. Consider a rectangle R whose coordinates of four apexes are $(h_1, v_1), (h_1, v_2), (h_2, v_1)$ and (h_2, v_2). Assume R includes k number of points each of which represents a character of x. Then we can construct a subsequence x' of x ($|x'| = k$) by concatenating all the characters represented as points in R from left to right. In this case, we say that R represents x' and x' is a *boxed subsequence* of x. Note that not every subsequence of x is a boxed subsequence. For the previous example, all characters of P and those of T can be represented in two-dimensional planes, respectively, as shown in Fig. 1. All the points of $T' = (10, 6, 7, 15, 16, 12, 13)$ are included in the shaded rectangle of the right plane and there are no other points in the shaded rectangle. Thus, T' is a boxed subsequence of T. But for another subsequence $T'' = (10, 2, 7, 15, 16, 12, 13)$, no rectangle can include all the points of T'' without including any other points of T. Thus, T'' is not a boxed subsequence of T. Avgustinovich et al. [2] firstly introduced the concept of boxed-mesh patterns and studied pattern avoidance, i.e., counting the number of permutations not containing a given pattern. Bruner et al. [5] introduced the BPPM problem and showed that it can be solved in $O(n^3)$ time when $|T| = n$ as follows. First, they fix two characters c_1 and c_2 ($c_1 < c_2$) in T. Then, they check whether the boxed subsequence including c_1 and c_2 as the smallest character and the largest character, respectively, is an occurrence of P in T.

In this paper, we propose an $O(n^2 \log m)$-time algorithm for the BPPM problem using $O(m)$ space. Our algorithm fixes a position i and then finds all the occurrences of P in T whose first character is the ith character of T based on interesting properties of boxed subsequences. To speed up pattern matching, our algorithm uses preprocessed information on P and order-statistic trees [9]. (Recently, independent work for the BPPM problem was performed by Amit et al. [1]. Their algorithm runs in $O(n^2)$ time using $O(n)$ space.)

This paper is organized as follows. In Section 2, we give some definitions and describe the previous works related to the BPPM problem. In Section 3, we present our $O(n^2 \log m)$-time algorithm for the BPPM problem, and then we present an algorithm for preprocessing P in Section 4. Finally, we conclude in Section 5.

2. Preliminaries

First, we give some basic definitions and notations on strings. Let Σ denote the set of characters such that a comparison of two characters can be done in constant time. A string x over the alphabet Σ is a sequence of characters derived from the alphabet Σ. For simplicity, we assume that the characters of x are all distinct. We denote the length of x by $|x|$ and the ith character by $x[i]$ ($1 \le i \le |x|$). A sequence $(x[i_1], x[i_2], \ldots, x[i_k])$ of characters in x is called a *subsequence* of x when $1 \le i_1 < i_2 < \cdots < i_k \le |x|$. A *substring* of x denoted by $x[i..j]$ is a subsequence of consecutive characters $(x[i], x[i+1], \ldots, x[j])$ in x. A *prefix* of x is a substring of x starting at the first position. For a string x, we denote by $\min(x)$ (resp. $\max(x)$) the smallest (resp. largest) character in x. For a string x and a character c, we denote by $x \oplus c$ the concatenation of x and c. The *rank* of a character c for a string x is defined as $\textsc{Rank}(c, x) = 1 + |\{i : x[i] < c,\ 1 \le i \le |x|\}|$.

We formally define the order-isomorphism [13]. Two strings x and y of the same length over Σ are called *order-isomorphic*, written $x \approx y$, if

$$x[i] \le x[j] \Leftrightarrow y[i] \le y[j] \text{ for all } 1 \le i, j \le |x|.$$

If two strings x and y are not order-isomorphic, we write $x \not\approx y$. For a string x, the *prefix representation* μ_x is defined as $\mu_x[i] = \textsc{Rank}(x[i], x[1..i])$ ($1 \le i \le |x|$) and it can be computed in $O(|x| \log |x|)$ time using an order-statistic tree [9], which is our main tool.[2] For example, the prefix representation for P in Fig. 1 is $\mu_P = (1, 1, 2, 4, 5, 4, 5)$ as shown in Table 1. We can check order-isomorphism of two strings using their prefix representations. For two strings x and y over Σ, $x \approx y$ if and

[2] On the Word RAM model, μ_x can be computed in $O(|x|\sqrt{\log |x|})$ time [6].

Table 1
μ_P and π for $P = (5, 3, 4, 8, 9, 6, 7)$.

i	1	2	3	4	5	6	7
$P[i]$	5	3	4	8	9	6	7
$\mu_P[i]$	1	1	2	4	5	4	5
$\pi[i]$	0	1	2	3	4	4	5

only if $\mu_x = \mu_y$ [12]. Since we mainly use the prefix representation μ_P of the given pattern P in our algorithm, we omit the subscript P in μ_P if not confusing.

Given a text $T[1..n]$ and a pattern $P[1..m]$, we say that P occurs at (i, j) of T if there exists a boxed subsequence T' of T such that $T[i]$ and $T[j]$ are the first and last characters of T', respectively, and $T' \approx P$. Also, we say that T' is an occurrence of P at (i, j) in T. In the previous example shown in Fig. 1, $T' = (10, 6, 7, 15, 16, 12, 13)$ is an occurrence of P since T' is a boxed subsequence of $T = (10, 6, 2, 7, 15, 16, 12, 19, 13, 11, 3)$ and $T' \approx P$. Though the relative order between characters of $T'' = (10, 2, 7, 15, 16, 12, 13)$ is the same as that of P, T'' is not an occurrence since it is not a boxed subsequence of T. Then the BPPM problem can be defined as follows.

Definition 1. Given a text T and a pattern P over Σ, the BPPM problem is to find all occurrences of P in T.

3. Boxed-mesh permutation pattern matching algorithms

3.1. Pivotal subsequences

We first give some basic notions and properties for our algorithm. For a boxed subsequence x' of a string x, x' is called a *full-width boxed* (for short, *f-boxed*) subsequence of x if x' includes only and all the characters c of x such that $\min(x') \le c \le \max(x')$. For example, a subsequence $T' = (T[1], T[2], T[4], T[7]) = (10, 6, 7, 12)$ is an f-boxed subsequence of $T[1..9]$ in Fig. 1. However, T' is not an f-boxed subsequence of $T[1..10]$, since $\min(T') = 6 \le T[10] = 11 \le 12 = \max(T')$ but $T[10]$ is not a character in T'. Furthermore, for an f-boxed subsequence x' of a string x, we define the lower boundary and the upper boundary of x' for x as follows.

- The lower boundary of x' for x, denoted by $lb(x', x)$, is the value of the largest character in x less than $\min(x')$. If no such character exists, $lb(x', x)$ is $-\infty$.
- The upper boundary of x' for x, denoted by $ub(x', x)$, is the value of the smallest character in x greater than $\max(x')$. If no such character exists, $ub(x', x)$ is ∞.

For the example in Fig. 1, $lb(T', T[1..9]) = 2$ and $ub(T', T[1..9]) = 13$ where $T' = (10, 6, 7, 12)$. Note that $lb(T', T[1..9]) \neq 3$ since '3' $(= T[11])$ is not a character in $T[1..9]$. For another string $T[1..8]$, the upper boundary of T' is $ub(T', T[1..8]) = 15$.

Given a pattern P, a boxed subsequence x' of a string x is called a *pivotal subsequence* of x if the following three conditions are satisfied:

C1. x' includes the first character of x,
C2. x' is an f-boxed subsequence of x, and
C3. x' is order-isomorphic to the prefix $P[1..|x'|]$ of P, i.e., $x' \approx P[1..|x'|]$.

For the example in Fig. 1, $T' = (T[1], T[2], T[4], T[7]) = (10, 6, 7, 12)$ is a pivotal subsequence of $T[1..9]$ since T' includes $T[1] = 10$, T' is an f-boxed subsequence of $T[1..9]$, and $T' = (10, 6, 7, 12) \approx P[1..4] = (5, 3, 4, 8)$. As another example, $T'' = (T[1], T[2], T[4], T[10]) = (10, 6, 7, 11)$ is a pivotal subsequence of $T[1..11]$. Note that the last character of x may not be included in x' and the shortest pivotal subsequence is $x[1]$. The following lemmas show the pivotal subsequences of x are all of distinct lengths.

Lemma 1. *For fixed k and r, there exists at most one f-boxed subsequence x' of a string x such that $|x'| = k$, $x'[1] = x[1]$, and* $\textsc{Rank}(x'[1], x') = r$.

Proof. Since $x'[1] = x[1]$ and $\textsc{Rank}(x'[1], x') = r$, the number of characters in x' smaller than $x[1]$ is $r - 1$. Also, since $|x'| = k$, the number of characters in x' larger than $x[1]$ is $k - r$. Furthermore, since x' is an f-boxed subsequence of x, x' consists of $x[1]$, the $r - 1$ largest characters in x smaller than $x[1]$ and the $k - r$ smallest characters in x larger than $x[1]$. Therefore, x' is unique (if exists). □

Lemma 2. *For a string x and a fixed k, there exists at most one pivotal subsequence of x of length k.*

Proof. Let x' be a pivotal subsequence of x of length k. By definition of the pivotal subsequence, $x'[1] = x[1]$ and x' is an f-boxed subsequence of x. Since $x' \approx P[1..k]$, $\textsc{Rank}(x'[1], x') = \textsc{Rank}(P[1], P[1..k])$. Therefore, x' is unique by Lemma 1. □

From Lemma 2, we can get the following corollary on the uniqueness of an occurrence at (i, j).

Corollary 1. *For fixed i and j ($1 \le i < j \le n$), there exists at most one boxed subsequence T' of T such that $T[i]$ and $T[j]$ are the first and the last characters of T', respectively, and $T' \approx P$.*

Proof. Since $T[i]$ and $T[j]$ are the first character and the last character in T', respectively, T' is an f-boxed subsequence of $T[i..j]$. Thus, T' is a unique pivotal subsequence of $T[i..j]$ of length m by Lemma 2. □

3.2. Algorithm

We present our algorithm for finding all the occurrences of a pattern $P[1..m]$ in a text $T[1..n]$. Our algorithm consists of $n - m + 1$ phases and in each Phase i ($1 \le i \le n - m + 1$), it finds all the occurrences of P whose first character is $T[i]$. Since each phase is completely independent of the other phases, we devote our attention to a fixed phase (Phase 1).

Phase 1 consists of $n - 1$ steps, from Step 2 to Step n. In Step j, the algorithm finds an occurrence of P (if exists) whose first character is $T[1]$ and the last character is $T[j]$ using pivotal subsequences. Let $\mathcal{Z}_j$ be the set of all the pivotal subsequences of $T[1..j]$ ($1 \le j \le n$) and let lz_j be the longest subsequence in $\mathcal{Z}_j$ such that $|lz_j| < m$. (lz_1 is simply $T[1]$ since $T[1]$ is the only pivotal subsequence of $T[1]$.) For P and $T[1..9]$ in Fig. 1, $\mathcal{Z}_9 = \{(10), (10, 7), (10, 6, 7), (10, 6, 7, 12), (10, 6, 7, 12, 13), (10, 6, 7, 15, 16, 12, 13)\}$ and $lz_9 = (10, 6, 7, 12, 13)$ (note that the longest subsequence $(10, 6, 7, 15, 16, 12, 13)$ in $\mathcal{Z}_9$ is not lz_9 since its length is equal to $m = 7$). In Step j ($2 \le j \le n$), given lz_{j-1}, we do the following.

- First, we decide whether P occurs at $(1, j)$ or not. Assume that there exists an occurrence (say z') of P whose first character is $T[1]$ and the last character is $T[j]$. Let z be the sequence obtained by deleting $T[j]$ from z', i.e., $z' = z \oplus T[j]$. Then, z' is a pivotal subsequence of $T[1..j]$ of length m and z is a pivotal subsequence of $T[1..j-1]$ of length $m - 1$. Note that z is lz_{j-1} in this case. Thus, the occurrence z' can be found simply by determining whether $lz_{j-1} \oplus T[j]$ is a pivotal subsequence of $T[1..j]$ or not.
- Second, we compute lz_j for the next step. Assume that we have $\mathcal{Z}_{j-1}$. From $\mathcal{Z}_{j-1}$, we can compute $\mathcal{Z}_j$ by checking if, for every $z \in \mathcal{Z}_{j-1}$, $z \oplus T[j]$ and z are pivotal subsequences of $T[1..j]$, which will be proven later (Lemma 3). Notice that our goal is to compute lz_j but not $\mathcal{Z}_j$. Thus, for $z \in \mathcal{Z}_{j-1}$ in decreasing order of length, we repeat the check until $z \oplus T[j]$ or z is a pivotal subsequence of $T[1..j]$.

To compute lz_j in Step j, we need $\mathcal{Z}_{j-1}$. However, we do not maintain $\mathcal{Z}_{j-1}$ explicitly. Instead, we compute the pivotal subsequences $z \in \mathcal{Z}_{j-1}$ in decreasing order of length from lz_{j-1}. The computation can be done efficiently using the π-function defined as follows for the given pattern P. Let $\mathcal{P}_q$ be the set of all the pivotal subsequences of $P[1..q]$ ($1 \le q \le m$). Then, $\pi[q]$ is the length of the longest subsequence x in $\mathcal{P}_q$ such that $|x| < q$. Obviously, $P[1..q]$ is the longest pivotal subsequence of $P[1..q]$ and thus, $\pi[q]$ is the length of the second longest sequence in $\mathcal{P}_q$. For $P[1..6] = (5, 3, 4, 8, 9, 6)$ in Fig. 1, $\mathcal{P}_6 = \{(5), (5, 4), (5, 3, 4), (5, 3, 4, 6), (5, 3, 4, 8, 9, 6)\}$ and thus $\pi[6] = 4$. (See Table 1 in Section 2 for the entire π-function.) Assume that there exists a pivotal subsequence of length 6 in $\mathcal{Z}_{j-1}$. Then, the value 4 of $\pi[6]$ informs us that there exists no pivotal subsequence of length 5 but there exists a pivotal subsequence of length 4 in $\mathcal{Z}_{j-1}$.

Algorithm 1 shows a pseudocode of our algorithm. The first for loop (line 3) represents the phases and the second for loop (line 5) represents the steps in each phase. In Phase 1, we initialize $z = lz_1$ (i.e., $T[1]$), $B_{lb} = lb(z, T[1]) = -\infty$, and $B_{ub} = ub(z, T[1]) = \infty$, respectively (line 4). In Step j ($2 \le j \le n$), given lz_{j-1}, $lb(lz_{j-1}, T[1..j-1])$ and $ub(lz_{j-1}, T[1..j-1])$, we repeat the while loop (lines 6–17) until lz_j is computed. In an iteration of the while loop, we consider a pivotal subsequence $z \in \mathcal{Z}_{j-1}$ in decreasing order of its length by maintaining the following loop invariant:

> At the beginning of each iteration of the while loop, z is a pivotal subsequence of $T[1..j-1]$, $B_{lb} = lb(z, T[1..j-1])$ and $B_{ub} = ub(z, T[1..j-1])$.

Note that $z = lz_{j-1}$ at the first iteration. Each iteration of the while loop consists of the following three stages. Let $r = \textsc{Rank}(T[j], z)$.

1. First, we check if $z \oplus T[j] \in \mathcal{Z}_j$ (line 8). If $B_{lb} < T[j] < B_{ub}$ and $r = \mu[|z| + 1]$, then $z \oplus T[j] \in \mathcal{Z}_j$ (Lemma 4). When $z \oplus T[j] \in \mathcal{Z}_j$, we have two subcases.
   - If $|z| < m - 1$ (line 9), then $lz_j = z \oplus T[j]$. Hence, we append $T[j]$ to z (the Append operation) and terminate Step j (break in line 9). In this case, since $B_{lb} < T[j] < B_{ub}$ and $T[j]$ is included in lz_j, $lb(lz_j, T[1..j])$ and $ub(lz_j, T[1..j])$ for the next step are equal to the current B_{lb} and B_{ub}, respectively.

Algorithm 1 Boxed-mesh permutation pattern matching.

```
1: Compute μ and π for P;
2: m ← |P|, n ← |T|;
3: for i ← 1 to n − m + 1 do
4:     z ← (T[i]), B_lb ← −∞, B_ub ← ∞;                                  ▷ Initialization
5:     for j ← i + 1 to n do
6:         while TRUE do                                                  ▷ for z ∈ Z_{j−1} in decreasing order of length
7:             r ← RANK(T[j], z);
8:             if B_lb < T[j] < B_ub and r = μ[|z| + 1] then              ▷ when z ⊕ T[j] ∈ Z_j
9:                 if |z| < m − 1 then APPEND(z, T[j]), break;            ▷ z ← z ⊕ T[j]
10:                else print "P occurs at (i, j)";
11:            if r = 1 then B_lb ← max(B_lb, T[j]), break;               ▷ when z ∈ Z_j
12:            else if r = |z| + 1 then B_ub ← min(B_ub, T[j]), break;
13:            else ℓ ← π[|z|];                                           ▷ when z ⊕ T[j] ≠ lz_j and z ≠ lz_j
14:                repeat
15:                    if P[1] > P[|z|] then B_lb ← EXTRACTMIN(z);
16:                    else B_ub ← EXTRACTMAX(z);
17:                until |z| = ℓ
```

- If $|z| = m - 1$ (line 10), then $z \oplus T[j]$ is an occurrence at (i, j) but it is not lz_j due to the length restriction that $|lz_j| < m$. Hence, we continue to compute lz_j.

For example, at the beginning Step 7, we are given $z = lz_6 = (10, 6, 7, 15, 16)$, $B_{lb} = lb(z, T[1..6]) = T[3] = 2$, and $B_{ub} = ub(z, T[1..6]) = \infty$. Since $B_{lb} < T[7] = 12 < B_{ub}$ and $r = \text{RANK}(T[7], z) = \text{RANK}(12, z) = 4$ is equal to $\mu[|z| + 1] = \mu[6] = 4$, $z \oplus T[7] = (10, 6, 7, 15, 16, 12)$ is a pivotal subsequence of $T[1..7]$. It is also lz_7 since $|z| = 5 < m - 1 = 6$.

2. Next, when $z \oplus T[j]$ is not lz_j, we check if $z \in \mathcal{Z}_j$ (lines 11–12). If $r = 1$ or $r = |z| + 1$, i.e., $T[j] < \min(z)$ or $T[j] > \max(z)$, then $z \in \mathcal{Z}_j$ (Lemma 5). In this case, $lz_j = z$ since $|z| < m$ and, for the next step, B_{lb} is updated to $lb(z, T[1..j])$ from $lb(z, T[1..j-1])$ and B_{ub} is updated to $ub(z, T[1..j])$ from $ub(z, T[1..j-1])$, respectively. Hence, we set $B_{lb} = \max(B_{lb}, T[j])$ (if $r = 1$) and $B_{ub} = \min(B_{ub}, T[j])$ (if $r = |z| + 1$), and terminate Step j (break in lines 11–12).
For example, consider Step 8, where $z = lz_7 = (10, 6, 7, 15, 16, 12)$, $B_{lb} = 2$, and $B_{ub} = \infty$. (In this case, $z \oplus T[8] \notin \mathcal{Z}_8$.) Since r is equal to $|z| + 1 = 7$ (i.e., $T[8] = 19 > \max(z) = 16$), z is a pivotal subsequence of $T[1..8]$ and $lz_8 = z$. Since $T[8]$ is not included in lz_8, B_{ub} for the next step is $ub(lz_8, T[1..8]) = \min(\infty, 19) = 19$.
3. When $z \oplus T[j] \neq lz_j$ and $z \neq lz_j$, we compute a new pivotal subsequence for the next iteration of the while loop, i.e., the longest sequence z' in $\mathcal{Z}_{j-1}$ whose length is less than $|z|$ (lines 13–17). Let $\ell = \pi[|z|]$. The new pivotal subsequence z' can be obtained by deleting characters one by one from z until $|z'| = \ell$ (Lemma 6). We delete $\min(z)$ if $P[1] > P[|z|]$, and $\max(z)$ otherwise. Also, for the next iteration, we update B_{lb} $(= lb(z', T[1..j-1]))$ when $\min(z)$ is deleted, and we update B_{ub} $(= ub(z', T[1..j-1]))$ when $\max(z)$ is deleted.
For example, consider Step 9, where $z = lz_8 = (10, 6, 7, 15, 16, 12)$, $B_{lb} = 2$, and $B_{ub} = 19$. (Note that not only $z \oplus T[9] \neq lz_9$ since $|z| = m - 1$ (lines 8–10) but also $z \neq lz_9$ since $r = \text{RANK}(T[9], z) = \text{RANK}(13, z) = 5$ (lines 11–12).) Since $\pi[6] = 4$, $P[6] > P[1]$, and $P[5] > P[1]$, $z' = (10, 6, 7, 12)$ is obtained by deleting the two largest characters from z, i.e., $T[6] = 16$ and $T[5] = 15$. Also, B_{ub} is updated to $ub(z', T[1..8]) = T[5] = 15$.

3.3. Analysis

We analyze the correctness of Algorithm 1. Lemma 3 shows that only a pivotal subsequence z of $T[1..j-1]$ or $z \oplus T[j]$ can be a pivotal subsequence of $T[1..j]$.

Lemma 3. *For $z' \in \mathcal{Z}_j$ $(2 \le j \le n)$, if z' does not include $T[j]$, then $z' \in \mathcal{Z}_{j-1}$; otherwise (if z' includes $T[j]$), $z \in \mathcal{Z}_{j-1}$ where z is the sequence obtained by deleting $T[j]$ from z', i.e., $z' = z \oplus T[j]$.*

Proof. First, consider the case when z' does not include $T[j]$. Since z' is an f-boxed subsequence of $T[1..j]$ and z' does not include $T[j]$, z' is also an f-boxed subsequence of $T[1..j-1]$. Since $z' \in \mathcal{Z}_j$, z' includes $T[1]$ and $z' \approx P[1..|z'|]$. Hence, z' is a pivotal subsequence of $T[1..j-1]$.

Next, consider the case when z' includes $T[j]$. Since $z' = z \oplus T[j]$ is an f-boxed subsequence of $T[1..j]$, z is also an f-boxed subsequence of $T[1..j-1]$. Moreover, $z \approx P[1..|z|]$ since $z' = z \oplus T[j] \approx P[1..|z|+1]$. Obviously, z includes $T[1]$. Hence, z is a pivotal subsequence of $T[1..j-1]$. □

By definition of $\mathcal{Z}_j$, $\mathcal{Z}_j$ includes the occurrence of P (if exists) whose first and last characters are $T[1]$ and $T[j]$, respectively. Thus, we can get the following corollary from Lemma 3, which shows Algorithm 1 finds correctly the occurrence $(1, j)$ (if exists) in Step j of Phase 1.

Corollary 2. *For an occurrence z' of P at $(1, j)$ in T, let z be the sequence obtained by deleting $T[j]$ from z', i.e., $z' = z \oplus T[j]$. Then, $z \in \mathcal{Z}_{j-1}$.*

The following two lemmas show conditions for $z \oplus T[j]$ and z to be pivotal subsequences of $T[1..j]$ when $z \in \mathcal{Z}_{j-1}$.

Lemma 4. *For $z \in \mathcal{Z}_{j-1}$ $(2 \le j \le n)$ such that $|z| < m$, let $B_{lb} = lb(z, T[1..j-1])$, $B_{ub} = ub(z, T[1..j-1])$, and $r = \textsc{Rank}(T[j], z)$. Then, $z \oplus T[j] \in \mathcal{Z}_j$ if and only if $B_{lb} < T[j] < B_{ub}$ and $r = \mu[|z|+1]$.*

Proof. Let $z' = z \oplus T[j]$. We first consider the case when $B_{lb} < T[j] < B_{ub}$ and $r = \mu[|z|+1] = \mu[|z'|]$. Obviously, z' includes $T[1]$ and since $B_{lb} < T[j] < B_{ub}$, z' is an f-boxed subsequence of $T[1..j]$. By definition of the rank, $\textsc{Rank}(T[j], z') = \textsc{Rank}(T[j], z) = r$. Since $z \approx P[1..|z|]$ and $\textsc{Rank}(T[j], z') = \textsc{Rank}(P[|z'|], P[1..|z'|])$ by the condition $r = \mu[|z'|]$, $z' \approx P[1..|z'|]$. Hence, $z' \in \mathcal{Z}_j$ in this case.

Next, we consider the case when $T[j] \le B_{lb}$ or $T[j] \ge B_{ub}$ or $r \ne \mu_P[|z'|]$. By definitions of B_{lb} and B_{ub}, there exist characters in $T[1..j-1]$ whose values are B_{lb} and B_{ub}. Thus, if $T[j] \le B_{lb}$ or $T[j] \ge B_{ub}$, z' is not an f-boxed subsequence of $T[1..j]$. Moreover, if $r \ne \mu_P[|z'|]$, then $\textsc{Rank}(T[j], z') \ne \textsc{Rank}(P[|z'|], P[1..|z'|])$ and thus $z' \not\approx P[1..|z'|]$. Hence, $z' \notin \mathcal{Z}_j$ in this case. □

Lemma 5. *For $z \in \mathcal{Z}_{j-1}$ $(2 \le j \le n)$, $z \in \mathcal{Z}_j$ if and only if $r = 1$ or $r = |z| + 1$, where $r = \textsc{Rank}(T[j], z)$.*

Proof. By definition of the rank, $r = 1$ if and only if $T[j] < \min(z)$, and $r = |z| + 1$ if and only if $T[j] > \max(z)$. (Note that $T[j]$ cannot be equal to $\min(z)$ and $\max(z)$ by the assumption that the characters of T are all distinct.) If $T[j] < \min(z)$ or $T[j] > \max(z)$, the f-boxed subsequence z of $T[1..j-1]$ is also an f-boxed subsequence of $T[1..j]$. Obviously, z includes $T[1]$ and $z \approx P[1..|z|]$. Thus, in this case, $z \in \mathcal{Z}_j$. If $\min(z) \le T[j] \le \max(z)$, z is not an f-boxed subsequence of $T[1..j]$ and thus $z \notin \mathcal{Z}_j$. □

Lemma 3, Lemma 4, and Lemma 5 show that Algorithm 1 computes correctly lz_j when pivotal subsequences $z \in \mathcal{Z}_{j-1}$ are given in decreasing order of length.

Next, we show that Algorithm 1 can compute correctly pivotal subsequences $z \in \mathcal{Z}_{j-1}$ in decreasing order of length in Step j of Phase 1.

Lemma 6. *Given a pivotal subsequence $z \in \mathcal{Z}_{j-1}$ $(2 \le j \le n)$, the repeat-until loop (lines 14–17) computes correctly the longest sequence in $\mathcal{Z}_{j-1}$ whose length is less than $|z|$.*

Proof. Let $\ell = \pi[|z|]$. We first prove that the sequence z' of length ℓ computed by the repeat-until loop satisfies all the conditions for being a pivotal subsequence of $T[1..j-1]$. Let $r_\ell = \textsc{Rank}(P[1], P[1..\ell])$. Then, since $z \approx P[1..|z|]$ and $\min(z)$ (resp. $\max(z)$) is deleted if $P[1] > P[|z|]$ (resp. $P[1] < P[|z|]$) in the repeat-until loop, $\textsc{Rank}(z'[1], z'[1..\ell])$ is r_ℓ and thus z' always includes $z[1] = T[1]$. Moreover, since z' is obtained by deleting some largest characters and some smallest characters from the f-boxed subsequence z of $T[1..j-1]$, z' is also an f-boxed subsequence of $T[1..j-1]$. Finally, we show that $z' \approx P[1..\ell]$. Assume that $z' = (z[p_1], \ldots, z[p_\ell])$. Let $P' = (P[p_1], \ldots, P[p_\ell])$. Then, $z' \approx P'$ since $z \approx P[1..|z|]$. Moreover, P' is an f-boxed subsequence of $P[1..|z|]$ such that $|P'| = \ell$, $P'[1] = P[1]$, and $\textsc{Rank}(P'[1], P') = r_\ell$ (recall that r_ℓ is equal to $\textsc{Rank}(z'[1], z'[1..\ell])$). Meanwhile, by definition of $\pi[|z|]$, there exists a pivotal subsequence P'' of $P[1..|z|]$ of length ℓ. Also, P'' is an f-boxed subsequence of $P[1..|z|]$ such that $|P''| = \ell$, $P''[1] = P[1]$, and $\textsc{Rank}(P''[1], P'') = r_\ell$. That is, P'' and P' satisfy the same conditions and thus P'' is the same as P' by Lemma 1. Therefore, $P' = P'' \approx P[1..\ell]$ and thus $z' \approx P' \approx P[1..\ell]$. Hence, z' is a pivotal subsequence of $T[1..j-1]$.

Next, we prove by contradiction that there exists no sequence z' of length ℓ' $(\ell < \ell' < |z|)$ in $\mathcal{Z}_{j-1}$. Suppose that such sequence z' exists. Since both z' and z are pivotal subsequences of $T[1..j-1]$ and $|z'| < |z|$, z' is a subsequence of z (otherwise, z' is not an f-boxed subsequence of $T[1..j-1]$). Assume that $z' = (z[p_1], \ldots, z[p_{\ell'}])$. Let P' be the subsequence $(P[p_1], \ldots, P[p_{\ell'}])$ of $P[1..q]$. Then, similarly to the above, it can be shown that P' is a pivotal subsequence of $P[1..|z|]$, which contradicts the definition of $\pi[|z|]$ since $|P'| = \ell' > \ell = \pi[|z|]$. Hence, there is no such sequence z'. □

Therefore, we get the following lemma.

Lemma 7. *Given the π-function, Algorithm 1 solves the BPPM problem correctly.*

Now we analyze the time complexity of Algorithm 1 except for computing the π-function. In each iteration of the while loop (lines 6–17), all statements run at most once except for the repeat-until loop (lines 14–17). Only when the repeat-until loop is executed, the while loop is repeated. Moreover, in each iteration of the repeat-until loop, either ExtractMin or ExtractMax (lines 15–16) is called. Let us consider how many times ExtractMax and ExtractMin are called in each phase. Whenever ExtractMax or ExtractMin is called, the length of z decreases by one. Moreover, the length of z increases at

Table 2
R_{min}, R_{max} and $\tilde{\mu}$ for $P = (5, 3, 4, 8, 9, 6, 7)$.

i	1	2	3	4	5	6	7
$P[i]$	5	3	4	8	9	6	7
R_{min}	0	−1	−2	−2	−2	−2	−2
R_{max}	0	0	0	1	2	3	4
$\tilde{\mu}$	0	−1	−1	1	2	1	2

Algorithm 2 Compute the π-function.

```
1: Compute the arrays R_min, R_max, and μ̃ for P;
2: m ← |P|, π[1] ← 0;
3: for q ← 2 to m do
4:     ℓ ← q − 1, r ← μ̃[q];                          ▷ z = P[1..ℓ] and r = rRank(P[ℓ + 1], P[1..ℓ + 1])
5:     while TRUE do
6:         if r = μ̃[ℓ + 1] and ℓ < q − 1 then        ▷ when z ⊕ T[j] ∈ P_q
7:             π[q] = ℓ + 1, break;
8:         else if r < R_min[ℓ] or r > R_max[ℓ] then π[q] = ℓ, break;   ▷ when z ∈ P_q
9:         else ℓ ← π[ℓ];                            ▷ length of the next pivotal subsequence in P_{q−1}
```

most by one in each step (by Append in line 9) and thus it increases at most by n in each phase. Hence, ExtractMax and ExtractMin runs $O(n)$ times in each phase. Furthermore, each operation on z (Rank, Append, ExtractMax, and ExtractMin) can be performed in $O(\log m)$ time by maintaining z in an order-statistic tree [9]. Therefore, each phase takes $O(n \log m)$ time. Finally, the prefix representation μ for P can also be computed in $O(m \log m)$ time as mentioned in Section 2. Hence, we get the following lemma.

Lemma 8. *Given the π-function, the BPPM problem can be solved in $O(n^2 \log m)$ time.*

4. Computing the π-function

In this section, we describe how to compute the π-function. In our preliminary version [8], an $O(m^2 \log m)$-time algorithm has been proposed. We improve it to run in $O(m^2)$ time. In our algorithm for computing the π-function, we use an *r-rank* defined as follows: For a character c and a string x, the r-rank of c for x, denoted by $r\text{Rank}(c, x)$, is the rank relative to the first character of x, i.e., $r\text{Rank}(c, x) = \text{Rank}(c, x) - \text{Rank}(x[1], x)$. Note that $r\text{Rank}(c, x)$ is positive if $c > x[1]$, negative if $c < x[1]$, and 0 if $c = x[1]$.

Our algorithm for computing the π-function is similar to Phase 1 of Algorithm 1. Recall that $\pi[q]$ $(1 \le q \le m)$ is the length of the longest sequence in $\mathcal{P}_q$ whose length is less than q, and $\mathcal{P}_q$ (the set of the pivotal subsequences of $P[1..q]$) is defined similarly to $\mathcal{Z}_j$ used in Phase 1 of Algorithm 1. Thus, for computing $\pi[q]$ $(2 \le q \le m)$, we use the subsequences in $\mathcal{P}_{q-1}$ (by definition, $\pi[1] = 0$). That is, for subsequences $z \in \mathcal{P}_{q-1}$ in decreasing order of length, we check if $z \oplus P[q]$ and z are pivotal subsequences of $P[1..q]$. Differently from Algorithm 1, however, we do not compute explicitly each pivotal subsequence z, its lower boundary B_{lb}, and its upper boundary B_{ub}.

For each subsequence $z \in \mathcal{P}_{q-1}$, we only maintain the length of z but not z itself. Note that a subsequence $z_\ell \in \mathcal{P}_{q-1}$ of a fixed length ℓ is unique (Lemma 2). Thus, the subsequence z_ℓ can be specified by only its length. Let a_ℓ and b_ℓ be the numbers of characters in $P[1..\ell]$ less and greater than $P[1]$, respectively. Then, since $z_\ell \approx P[1..\ell]$, z_ℓ consists of $P[1]$, the a_ℓ largest characters in $P[1..q-1]$ less than $P[1]$, and the b_ℓ smallest characters in $P[1..q-1]$ greater than $P[1]$. For example, assume $\ell = 2$ and $q = 6$ in Fig. 1. Then, $a_2 = 1$ and $b_2 = 0$, and thus $z_2 \in \mathcal{P}_6$ consists of $P[1]$ and $P[3]$ since $P[3] = 4$ is the largest character in $P[1..5]$ less than $P[1] = 5$. Let us define two arrays R_{min} and R_{max} as $R_{min}[\ell] = -a_\ell$ and $R_{max}[\ell] = b_\ell$ $(1 \le \ell \le m)$. Then, R_{min} and R_{max} can be computed using r-ranks as follows:

- $R_{min}[\ell] = r\text{Rank}(c_{min}, P[1..\ell])$ where $c_{min} = \min(P[1..\ell])$, and
- $R_{max}[\ell] = r\text{Rank}(c_{max}, P[1..\ell])$ where $c_{max} = \max(P[1..\ell])$.

Then, $z_\ell \in \mathcal{P}_{q-1}$ consists of the characters c in $P[1..q-1]$ such that $R_{min}[\ell] \le r\text{Rank}(c, P[1..q-1]) \le R_{max}[\ell]$. (Note that $R_{min}[\ell]$ and $R_{max}[\ell]$ are not dependent on q.) Also, we use $\tilde{\mu}$ instead of μ, where $\tilde{\mu}$ is the r-rank version of the prefix representation μ of P, i.e., $\tilde{\mu}[\ell] = r\text{Rank}(P[\ell], P[1..\ell])$ $(1 \le \ell \le m)$. See Table 2 for an example.

Algorithm 2 shows a pseudocode of our algorithm for computing the π-function. First, we compute the arrays R_{min}, R_{max}, and $\tilde{\mu}$, i.e., $\tilde{\mu}[\ell] = r\text{Rank}(P[\ell], P[1..\ell])$ $(1 \le \ell \le m)$. Our algorithm consists of $m - 1$ steps, from Step 2 to Step m. Initially, we set $\pi[1] = 0$ (line 2). In Step q $(2 \le q \le m)$, we maintain the length ℓ of a sequence $z \in \mathcal{P}_{q-1}$ and compute $\pi[q]$. Let lz_q $(1 \le q \le m)$ be the longest subsequence in $\mathcal{P}_q$ and lz'_q be the longest subsequence in $\mathcal{P}_q$ such that $|lz'_q| < q$. Since lz_q is simply $P[1..q]$, we do not need to compute lz_q for the next step. Thus, at the beginning of Step q, we set $\ell = q - 1$ (i.e., $z = P[1..q-1]$) and repeat the while loop (lines 5–9) until lz'_q is computed (the while loop always terminates, since for $q > 1$ and $\ell = 1$ the condition in line 8 is always true). In an iteration of the while loop, we consider a pivotal subsequence $z \in \mathcal{P}_{q-1}$ in decreasing order of length. Let r be $\tilde{\mu}[q] = r\text{Rank}(P[q], P[1..q])$. Each iteration consists of three stages.

1. First, we check if whether $z \oplus P[q] \in \mathcal{P}_q$ (line 6). Since the definition of $\mathcal{P}_q$ is the same as that of $\mathcal{Z}_j$ in Section 3, we can check it in a similar way as in Algorithm 1. Since r is a rank for $P[1..q]$ but not for z, however, we can avoid the comparison on the lower and upper boundaries of z (in line 8 of Algorithm 1). If $\ell < q-1$, then $z \oplus P[q]$ is lz'_q, and thus we set $\pi[q] = |lz'_q| = \ell + 1$ and terminate Step q.
2. Next, we check if $z \in \mathcal{P}_q$ (line 8), which can be done by comparing ranks as in Algorithm 1. If $r < R_{min}[\ell]$ or $r > R_{max}[\ell]$, then $P[q] < \min(z)$ or $P[q] > \max(z)$, and thus $z \in \mathcal{P}_q$. Since z is lz'_q, we set $\pi[q] = |z| = \ell$ and terminate Step q.
3. If $z \oplus P[q] \neq lz'_q$ and $z \neq lz'_q$, we compute the length of a new pivotal subsequence for the next iteration of the while loop, i.e., the longest sequence $z' \in \mathcal{P}_{q-1}$ whose length is less than ℓ. The length of z' is simply $\pi[\ell]$ as in Algorithm 1. Note that $\pi[\ell]$ is known in Step q since $\ell < q$, and we do not compute z' explicitly.

Now we consider the correctness of Algorithm 2. Lemmas for $\mathcal{Z}_j$ presented in Section 3.3 are also satisfied for $\mathcal{P}_q$. Since we use ranks for $P[1..q]$ but not for a pivotal subsequence z, however, we need the following two lemmas on conditions for $z \oplus P[q]$ and z to be pivotal subsequences of $P[1..q]$ when $z \in \mathcal{P}_{q-1}$.

Lemma 9. *For $z \in \mathcal{P}_{q-1}$ $(2 \le q \le m)$, $z \oplus P[q] \in \mathcal{P}_q$ if and only if $r = \tilde{\mu}[\ell + 1]$, where $r = r\textsc{Rank}(P[q], P[1..q])$ and $\ell = |z|$.*

Proof. Let $z' = z \oplus P[q]$. We first prove that $z' \in \mathcal{P}_q$ if $r = \tilde{\mu}[\ell + 1]$. Obviously, z' includes $P[1]$. Let $B_{lb} = lb(z, P[1..q-1])$. Since z is a pivotal subsequence of $P[1..q-1]$ of length ℓ, $r\textsc{Rank}(min(z), P[1..q-1]) = R_{min}[\ell]$ and $r\textsc{Rank}(B_{lb}, P[1..q-1]) = R_{min}[\ell] - 1$. Moreover, $r = r\textsc{Rank}(P[q], P[1..q]) = r\textsc{Rank}(P[q], P[1..q-1])$ by definition of the r-rank. Thus, if $P[q] < B_{lb}$, then $r < R_{min}[\ell] - 1$ and thus r cannot be equal to $\tilde{\mu}[\ell + 1]$ since $\tilde{\mu}[\ell + 1] \ge R_{min}[\ell + 1] \ge R_{min}[\ell] - 1$. Therefore, $B_{lb} < P[q]$ and similarly $P[q] < B_{ub}$ where $B_{ub} = ub(z, P[1..q-1])$, and thus z' is an f-boxed subsequence of $P[1..q]$. Finally, since z' is an f-boxed subsequence of $P[1..q]$, $r\textsc{Rank}(P[q], z') = r\textsc{Rank}(P[q], P[1..q]) = r$ and, since $r = \tilde{\mu}[\ell + 1]$, $r\textsc{Rank}(P[q], z') = r\textsc{Rank}(P[\ell + 1], P[1..\ell + 1])$. Since $z \approx P[1..\ell]$ and $r\textsc{Rank}(z'[\ell + 1], z') = r\textsc{Rank}(P[\ell + 1], P[1..\ell + 1])$, $z' \approx P[1..\ell + 1]$ (note that $z'[\ell + 1] = P[q]$). Hence, $z' \in \mathcal{P}_q$.

Next, we prove that $z' \notin \mathcal{P}_q$ if $r \neq \tilde{\mu}[\ell + 1]$. We only consider the case when z' is an f-boxed subsequence of $P[1..q]$ (otherwise, $z' \notin \mathcal{P}_q$). In this case, $r\textsc{Rank}(P[q], z') = r\textsc{Rank}(P[q], P[1..q]) = r$. Since $r \neq \tilde{\mu}[\ell + 1]$, $r\textsc{Rank}(z'[\ell + 1], z') \neq r\textsc{Rank}(P[\ell + 1], P[1..\ell + 1])$ and thus $z' \not\approx P[1..\ell + 1]$. Hence, $z' \notin \mathcal{P}_q$. □

Lemma 10. *For $z \in \mathcal{P}_{q-1}$ $(2 \le q \le m)$, $z \in \mathcal{P}_q$ if and only if $r < R_{min}[\ell]$ or $r > R_{max}[\ell]$, where $r = r\textsc{Rank}(P[q], P[1..q])$ and $\ell = |z|$.*

Proof. By definition of the rank, $r\textsc{Rank}(P[q], P[1..q]) = r\textsc{Rank}(P[q], P[1..q-1])$ and, since z is an f-boxed subsequence of $P[1..q-1]$, $r\textsc{Rank}(P[q], P[1..q-1]) = r\textsc{Rank}(P[q], z)$. Thus, $r = r\textsc{Rank}(P[q], z)$. Since $z \approx P[1..\ell]$, $r\textsc{Rank}(\min(z), z) = R_{min}[\ell]$, and thus $P[q] < \min(z)$ if and only if $r = r\textsc{Rank}(P[q], z) < R_{min}[\ell]$. Similarly, $P[q] > \max(z)$ if and only if $r > R_{max}[\ell]$. If and only if $P[q] < \min(z)$ or $P[q] > \max(z)$, $z \in \mathcal{P}_q$ as already shown in the proof of Lemma 5. Therefore, $z \in \mathcal{P}_q$ if and only if $r < R_{min}[\ell]$ or $r > R_{max}[\ell]$. □

We analyze the time complexity of Algorithm 2. Since $\tilde{\mu}$ can be computed in $O(m)$ time from the prefix representation μ for P which can be computed in $O(m \log m)$ time, $\tilde{\mu}$ can be computed in $O(m \log m)$ time. The arrays R_{min} and R_{max} can be computed in $O(m)$ time by counting the number of characters less than and greater than $P[1]$, respectively. The running time of each step is bounded by the while loop. In Step q, the length ℓ of z is initialized to $q-1$ and it decreases at each iteration of the while loop. Thus, each step takes $O(m)$ time and therefore Algorithm 2 takes $O(m^2)$ time since it consists of $m-1$ steps.

Lemma 11. *The π-function for P of length m can be computed in $O(m^2)$ time.*

From Lemma 8 and Lemma 11, we can get the following theorem.

Theorem 1. *Given a text of length n and a pattern of length m, the BPPM problem can be solved in $O(n^2 \log m)$ time.*

5. Concluding remarks

In this paper, we have proposed a general framework for solving the BPPM problem which finds an occurrence by fixing its first and last characters. The key of the framework is the notion of the pivotal subsequences and their relationship. A naive implementation of the framework leads to an $O(n^2 m)$-time algorithm, which was given in our preliminary version [8]. In this paper, we have presented an $O(n^2 \log m)$-time algorithm (Algorithm 1) for the BPPM problem, which makes use of order-statistic trees and the preprocessed information on a pattern P, called the π-function. The π-function can be computed in $O(m^2 \log m)$ time by applying Algorithm 1 for the pattern, which was also given in our preliminary version [8]. We have presented an improved algorithm (Algorithm 2) for computing the π-function which runs in $O(m^2)$ time.

Acknowledgements

Joong Chae Na was supported by the MSIP (Ministry of Science, ICT & Future Planning), Korea, under National program for Excellence in Software program (the SW oriented college support program) (R7718-16-1005) supervised by the IITP (Institute for Information & communications Technology Promotion), and by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT & Future Planning (2014R1A1A1004901). Jeong Seop Sim was supported by the Genome Program for Fostering New Post-Genome industry of the National Research Foundation (NRF) funded by the Korean government (MSIP) (NRF-2014M3C9A3064706), and by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIP) (No. 2014R1A2A1A11050337).

References

[1] M. Amit, P. Bille, P.H. Cording, I.L. Gørtz, H.W. Vildhøj, Boxed permutation pattern matching, in: 27th Annual Symposium on Combinatorial Pattern Matching, CPM 2016, June 27–29, 2016, Tel Aviv, Israel, 2016, pp. 20:1–20:11.
[2] S.V. Avgustinovich, S. Kitaev, A. Valyuzhenich, Avoidance of boxed mesh patterns on permutations, Discrete Appl. Math. 161 (1–2) (2013) 43–51.
[3] P. Bose, J.F. Buss, A. Lubiw, Pattern matching for permutations, Inform. Process. Lett. 65 (5) (1998) 277–283.
[4] P. Brändén, A. Claesson, Mesh patterns and the expansion of permutation statistics as sums of permutation patterns, Electron. J. Combin. 18 (2) (2011) P5.
[5] M. Bruner, M. Lackner, The computational landscape of permutation patterns, CoRR, arXiv:1301.0340 [abs], 2013.
[6] T.M. Chan, M. Patrascu, Counting inversions, offline orthogonal range counting, and related problems, in: Proceedings of the Twenty-First Annual ACM–SIAM Symposium on Discrete Algorithms, SODA 2010, Austin, Texas, USA, January 17–19, 2010, 2010, pp. 161–173.
[7] S. Cho, J.C. Na, K. Park, J.S. Sim, A fast algorithm for order-preserving pattern matching, Inform. Process. Lett. 115 (2) (2015) 397–402.
[8] S. Cho, J.C. Na, J.S. Sim, Improved algorithms for the boxed-mesh permutation pattern matching problem, in: Combinatorial Pattern Matching – 26th Annual Symposium, Proceedings, CPM 2015, Ischia Island, Italy, June 29 – July 1, 2015, 2015, pp. 138–148.
[9] T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein, Introduction to Algorithms, 3rd ed., MIT Press, 2009.
[10] M. Crochemore, C.S. Iliopoulos, T. Kociumaka, M. Kubica, A. Langiu, S.P. Pissis, J. Radoszewski, W. Rytter, T. Walen, Order-preserving incomplete suffix trees and order-preserving indexes, in: String Processing and Information Retrieval – 20th International Symposium, Proceedings, SPIRE 2013, Jerusalem, Israel, October 7–9, 2013, 2013, pp. 84–95.
[11] S. Elizalde, M. Noy, Consecutive patterns in permutations, Adv. in Appl. Math. 30 (1) (2003) 110–125.
[12] J. Kim, P. Eades, R. Fleischer, S. Hong, C.S. Iliopoulos, K. Park, S.J. Puglisi, T. Tokuyama, Order-preserving matching, Theoret. Comput. Sci. 525 (2014) 68–79.
[13] M. Kubica, T. Kulczynski, J. Radoszewski, W. Rytter, T. Walen, A linear time algorithm for consecutive permutation pattern matching, Inform. Process. Lett. 113 (12) (2013) 430–433.