Computing the Parameterized Burrows–Wheeler Transform Online

Daiki Hashimoto[1], Diptarama Hendrian[1], Dominik Köppl[2], Ryo Yoshinaka[1], and Ayumi Shinohara[1]

[1]Tohoku University, Japan
[2]Tokyo Medical and Dental University, Japan

Abstract

Parameterized strings are a generalization of strings in that their characters are drawn from two different alphabets, where one is considered to be the alphabet of static characters and the other to be the alphabet of parameter characters. Two parameterized strings are a parameterized match if there is a bijection over all characters such that the bijection transforms one string to the other while keeping the static characters (i.e., it behaves as the identity on the static alphabet). Ganguly et al. [SODA 2017] proposed the parameterized Burrows–Wheeler transform (pBWT) as a variant of the Burrows–Wheeler transform for space-efficient parameterized pattern matching. In this paper, we propose an algorithm for computing the pBWT online by reading the characters of a given input string one-by-one from right to left. Our algorithm works in $O(|\Pi| \log n / \log \log n)$ amortized time for each input character, where n and Π denote the size of the input string and the alphabet of the parameter characters, respectively.

1 Introduction

The *parameterized matching problem* (*p-matching problem*) [2] is a generalization of the classic pattern matching problem in the sense that we here consider two disjoint alphabets, the set Σ of *static characters* and the set Π of *parameter characters*. We call a string over $\Sigma \cup \Pi$ a *parameterized string (p-string)*. Two equal-length p-strings X and Y are said to *parameterized match* (*p-match*) if there is a bijection that renames the parameter characters in X so X becomes equal to Y. The *p-matching problem* is, given a text p-string T and pattern p-string P, to output the positions of all substrings of T that p-match P. The p-matching problem is motivated by applications in the software maintenance [1, 2], the plagiarism detection [5], the analysis of gene structures [17], and so on. There exist indexing structures that support p-matching, such as parameterized suffix trees [1, 17], parameterized suffix arrays [7, 10], and so on [4, 6, 13, 14]; see also [12] for a survey. A drawback of these indexing structures is that they have high space requirements.

A more space-efficient indexing structure, the *parameterized Burrows–Wheeler transform (pBWT)*, was proposed by Ganguly et al. [8]. The pBWT is a variant of the *Burrows–Wheeler transform (BWT)* [3] that can be used as an indexing structure for p-matching using only $o(n \log n)$ bits of space. Later on, Kim and Cho [11] improved this indexing structure by changing the encoding of p-strings used for defining the pBWT. Recently, Ganguly et al. [9] augmented this index with capabilities of a suffix tree while keeping the space within $o(n \log n)$ bits. However, as far as we are aware of, none research related to the pBWT [8, 11, 9, 18] has discussed how to construct their pBWT-based data structures in detail. Their construction algorithms mainly rely on the parameterized suffix tree. Given the parameterized suffix tree of a p-string T of length n, the pBWT of T can be constructed in $O(n \log(|\Sigma| + |\Pi|))$ time offline.

In this paper, we propose an algorithm for constructing pBWTs and related data structures used for indexing structures of p-matching. Our algorithm constructs the data structures directly in an online manner by reading the input text from right to left. The algorithm uses the dynamic array data structures of Navarro and Nekrich [15] to maintain our growing arrays. For each character read, our algorithm takes $O(|\Pi| \log n/ \log\log n)$ amortized time, where n is the size of input string. Therefore, we can compute pBWT of a p-string T of length n in $O(n|\Pi| \log n/ \log\log n)$ time in total. In comparison, computing the standard BWT on a string T (i.e., the pBWT on a string having no parameter characters) can be done in $O(n \log n/ \log\log n)$ time with the dynamic array data structures [15] (see [16] for a description of this online algorithm). Looking at our time complexity, the factor $|\Pi|$ also appears in the time complexity of an offline construction algorithm of parameterized suffix arrays [7] as $O(n|\Pi|)$ and a right-to-left online construction algorithm of parameterized suffix trees [13] as $O(n|\Pi| \log(|\Pi|+|\Sigma|))$. This suggests it would be rather hard to improve the time complexity of the online construction of pBWT to be independent of $|\Pi|$.

2 Preliminaries

We denote the set of nonnegative integers by $\mathbb{N}$ and let $\mathbb{N}_+ = \mathbb{N} \setminus \{0\}$ and $\mathbb{N}_\infty = \mathbb{N}_+ \cup \{\infty\}$. The set of strings over an alphabet A is denoted by A^*. The empty string is denoted by ε. The length of a string $W \in A^*$ is denoted by $|W|$. For a subset $B \subseteq A$, the set of elements of B occurring in $W \in A^*$ is denoted by $B \upharpoonright W$. We count the number of occurrences of characters of B in a string W by $|W|_B$. So, $|W|_A = |W|$. When B is a singleton of b, i.e., $B = \{b\}$, we often write $|W|_b$ instead of $|W|_{\{b\}}$. When W is written as $W = XYZ$, X, Y, and Z are called *prefix*, *factor*, and *suffix* of W, respectively. The i-th character of W is denoted by $W[i]$ for $1 \le i \le |W|$. The factor of W that begins at position i and ends at position j is $W[i : j]$ for $1 \le i \le j \le |W|$. For convenience, we abbreviate $W[1 : i]$ to $W[\,: i]$ and $W[i : |W|]$ to $W[i :\,]$ for $1 \le i \le |W|$. Let $\mathit{Rot}(W, 0) = W$ and $\mathit{Rot}(W, i+1) = \mathit{Rot}(W, i)[|W|]\mathit{Rot}(W, i)[: |W| - 1]$ be the i-th right rotation of W. Note that $\mathit{Rot}(W, i) = \mathit{Rot}(W, i+|W|)$. For convenience we denote $W_i = \mathit{Rot}(W, i)$. Let $\mathit{Left}_W(a)$ and $\mathit{Right}_W(a)$ be the leftmost and rightmost positions of a character $a \in A$ in W, respectively. If a does not occur in W, define $\mathit{Left}_W(a) = \mathit{Right}_W(a) = 0$.

2.1 Parameterized Burrows–Wheeler transform

Throughout this paper, we fix two disjoint ordered alphabets Σ and Π. We call elements of Σ *static characters* and those of Π *parameter characters*. Elements of Σ^* and $(\Sigma \cup \Pi)^*$ are called *static strings* and *parameterized strings* (or *p-strings* for short), respectively.

Two p-strings S and T of the same length are a *parameterized match (p-match)*, denoted by $S \approx T$, if there is a bijection f on $\Sigma \cup \Pi$ such that $f(a) = a$ for any $a \in \Sigma$ and $f(S[i]) = T[i]$ for all $1 \le i \le |T|$ [2]. We use Kim and Cho's version of p-string encoding [11], which replaces 0 in Baker's encoding [1] by ∞. The *prev-encoding* $\langle T \rangle$ of T is the string over $\Sigma \cup \mathbb{N}_\infty$ of length $|T|$ defined by

$$\langle T \rangle[i] = \begin{cases} T[i] & \text{if } T[i] \in \Sigma, \\ \infty & \text{if } T[i] \in \Pi \text{ and } \mathit{Right}_{T[:i-1]}(T[i]) = 0, \\ i - \mathit{Right}_{T[:i-1]}(T[i]) & \text{if } T[i] \in \Pi \text{ and } \mathit{Right}_{T[:i-1]}(T[i]) \neq 0 \end{cases}$$

for $1 \le i \le |T|$. When $T[i] \in \Pi$, $\langle T \rangle[i]$ represents the distance between i and the previous occurrence position of the same parameter character. If $T[i]$ does not occur before the position i, the distance is assumed to be ∞. We call a string $W \in (\Sigma \cup \mathbb{N}_\infty)^*$ a *pv-string* if $W = \langle T \rangle$ for some p-string T. For any p-strings S and T, $S \approx T$ if and only if $\langle S \rangle = \langle T \rangle$ [2]. For example, given $\Sigma = \{\mathtt{a}, \mathtt{b}\}$ and $\Pi = \{\mathtt{u}, \mathtt{v}, \mathtt{x}, \mathtt{y}\}$, $S = \mathtt{uvvauvb}$ and $T = \mathtt{xyyaxyb}$ are a p-match by f with $f(\mathtt{u}) = \mathtt{x}$ and $f(\mathtt{v}) = \mathtt{y}$, where $\langle S \rangle = \langle T \rangle = \infty\infty 1\mathtt{a}43\mathtt{b}$.

For defining pBWT, we use another encoding $[\![T]\!]$ given by

$$[\![T]\!][i] = \begin{cases} T[i] & \text{if } T[i] \in \Sigma, \\ |\Pi \upharpoonright T_{n-i}[1 : \mathit{Left}_{T_{n-i}}(T[i])]| & \text{if } T[i] \in \Pi \end{cases}$$

Table 1: The pBWT $pBWT(T) = \mathsf{L}_T = \texttt{a33131\$22aa}$ of the example string $T = \texttt{xayzzazyza\$}$ with related arrays, where $\Sigma = \{\texttt{a}\}$ and $\Pi = \{\texttt{x}, \texttt{y}, \texttt{z}\}$.

i	T_i	$\langle T_i \rangle$	$RA_T[i]$	$\mathsf{LCP}^\infty_T[i]$	$\langle T_{RA_T[i]} \rangle$	$\mathsf{F}_T[i]$	$[\![T_{RA_T[i]}]\!]$	$\mathsf{L}_T[i]$
1	$xayzzazyza	$∞a∞∞1a252a	1	0	$∞a∞∞1a252a	$	$3a211a233a	a
2	a$xayzzazyz	a$∞a∞∞1a252	2	0	a$∞a∞∞1a252	a	a$3a211a233	3
3	za$xayzzazy	∞a$∞a∞61a25	10	2	a∞∞1a252a$∞	a	a211a233a$3	3
4	yza$xayzzaz	∞∞a$∞a661a2	6	0	a∞∞2a$∞a661	a	a233a$3a211	1
5	zyza$xayzza	∞∞2a$∞a661a	3	1	∞a$∞a∞61a25	3	3a$3a211a23	3
6	azyza$xayzz	a∞∞2a$∞a661	7	1	∞a2∞2a$∞a66	1	1a233a$3a21	1
7	zazyza$xayz	∞a2∞2a$∞a66	11	1	∞a∞∞1a252a$	3	3a211a233a$	$
8	zzazyza$xay	∞1a2∞2a$∞a6	8	1	∞1a2∞2a$∞a6	1	11a233a$3a2	2
9	yzzazyza$xa	∞∞1a252a$∞a	4	2	∞∞a$∞a661a2	3	33a$3a211a2	2
10	ayzzazyza$x	a∞∞1a252a$∞	9	2	∞∞1a252a$∞a	2	211a233a$3a	a
11	xayzzazyza$	∞a∞∞1a252a$	5	0	∞∞2a$∞a661a	2	233a$3a211a	a

for $1 \le i \le |T|$. When $T[i] \in \Pi$, $[\![T]\!][i]$ counts the number of distinct parameter characters in T between i and the next occurrence of $T[i]$, if $T[i]$ occurs after i. If i is the rightmost occurrence position of $T[i]$, then we continue counting parameter characters from the left end to the right until we find $T[i]$. Since $T[i]$ occurs in T_{n-i} as the last character, $[\![T]\!][i]$ cannot be zero. Note that $[\![Rot(T, i)]\!] = Rot([\![T]\!], i)$ by definition. It is not hard to see that for any p-strings S and T, $S \approx T$ if and only if $[\![S]\!] = [\![T]\!]$ (see Proposition 1 in the appendix). For example, the two strings S and T given above are encoded as $[\![S]\!] = [\![T]\!] = \texttt{212a22b}$.

Hereafter in this section, we fix a p-string T of length n which ends with a special static character \$ which occurs nowhere else in T. We extend the linear order over Σ to $\Sigma \cup \mathbb{N}_\infty$ by letting $\$ < a < i < \infty$ for any $a \in \Sigma \setminus \{\$\}$ and $i \in \mathbb{N}_+$. The order over $\mathbb{N}_+$ coincides with the usual numerical order.

The pBWT of T is defined through sorting $[\![T_p]\!]$ for $p = 1, \ldots, n$ using $\langle T_p \rangle$ as keys.

Definition 1 (Parameterized rotation array)**.** *The parameterized rotation array RA_T of T is an array of size n such that $RA_T[i] = p$ with $1 \le p \le n$ if and only if $\langle T_p \rangle$ is the i-th lexicographically smallest string in $\{ \langle T_p \rangle \mid 1 \le p \le n \}$. We denote its inverse by RA_T^{-1}, i.e., $RA_T^{-1}[p] = i$ iff $RA_T[i] = p$.*

Note that RA_T and RA_T^{-1} are well-defined and bijective due to the presence of \$ in T. Here, we have $RA_T[i] = n - pSA_T[i] + 1$, where pSA_T refers to the suffix array $\mathtt{pSA}_\infty$ in [11]. The array gives an $n \times n$ square matrix $([\![T_{RA_T[i]}]\!])_{i=1}^n$, which we call the *rotation sort matrix* of T, whose (i, p) entry is $[\![T_{RA_T[i]}]\!][p]$. The pBWT of T is formed by the characters in the last column of the matrix.

Definition 2 (pBWT [11])**.** *The parameterized Burrows–Wheeler transform (pBWT) of a p-string T, denoted by $pBWT(T)$, is a string of length n such that $pBWT(T)[i] = [\![T_{RA[i]}]\!][n]$.*

An example pBWT can be found in Table 1. We will use L_T as a synonym of $pBWT(T)$, since it represents the *last* column of the matrix $([\![T_{RA_T[i]}]\!])_{i=1}^n$. When picking up the characters from the *first* column, we obtain another array F_T. That is, $\mathsf{F}_T[i] = [\![T_{RA_T[i]}]\!][1]$ for all $i \in \{1, \ldots, n\}$. Those arrays L_T and F_T are "linked" by the following mapping.

Definition 3 (LF mapping)**.** *The LF mapping $LF_T : \{1, \ldots, n\} \to \{1, \ldots, n\}$ for T is defined as $LF_T(i) = j$ if $T_{RA[i]+1} = T_{RA[j]}$.*

By rotating T_p to the right by one, the last character moves to the first position in T_{p+1}. Roughly speaking, $\mathsf{L}_T[i]$ and $\mathsf{F}_T[LF_T(i)]$ "originate" in the same character occurrence of T, which implies $\mathsf{L}_T[i] = \mathsf{F}_T[LF_T(i)]$ in particular. One can recover $[\![T]\!]$ as $[\![T]\!][p] = \mathsf{L}_T[LF^{-p}(k_T)]]$ for $1 \le p \le n$ where $k_T = RA_T^{-1}[n]$. L_T, F_T, and LF_T are used for pattern matching based on pBWT. See [11] for the details.

Our pBWT construction algorithm maintains neither RA_T nor LF_T, but involves some helper data structures in addition to L_T and F_T. Among those, the array LCP^∞_T is worth explaining before going into the algorithmic details. For two pv-strings X and Y, let $lcp^\infty(X,Y) = |W|_\infty$ be the number of ∞'s in the longest common prefix W of X and Y. The following array counts the number of ∞'s in the longest common prefixes of two adjacent rows in $(\langle T\rangle_{RA_T[i]})_{i=1}^{n}$.

Definition 4 (∞-LCP array)**.** The ∞-LCP array LCP_T^∞ *of T is an array of size n such that* $\mathsf{LCP}_T^\infty[n] = 0$ *and* $\mathsf{LCP}_T^\infty[i] = lcp^\infty(\langle T_{RA_T[i]}\rangle, \langle T_{RA_T[i+1]}\rangle)$ *for $1 \le i < n$.*

Table 1 shows an example of a pBWT and related (conceptual) data structures. We can compute $lcp^\infty(\langle T_{RA_T[i]}\rangle, \langle T_{RA_T[j]}\rangle)$ using LCP_T^∞ as follows.

Lemma 1. *For $1 \le i < j \le n$, $lcp^\infty(\langle T_{RA_T[i]}\rangle, \langle T_{RA_T[j]}\rangle) = \min_{i \le k < j} \mathsf{LCP}_T^\infty[k]$.*

Kim and Cho [11] showed some basic relations among L_T, LF_T, and lcp^∞. We rephrase Lemma 3 of [11] into a form convenient for our discussions.

Lemma 2. *Consider i and j with $1 \le i < j \le n$ and $T_{RA_T[i]}[n], T_{RA_T[j]}[n] \in \Pi$. Then, $LF_T(i) < LF_T(j)$ iff $\min\{\mathsf{L}_T[i] - 1,\ lcp^\infty(\langle T_{RA_T[i]}\rangle, \langle T_{RA_T[j]}\rangle)\} < \mathsf{L}_T[j]$.*

Corollary 1 ([11])**.** *If $i < j$ and $\mathsf{L}_T[i] = \mathsf{L}_T[j]$, then $LF_T(i) < LF_T(j)$.*

To maintain L_T, F_T, and LCP_T^∞ dynamically, our algorithm uses the data structure for dynamic arrays by Navarro and Nekrich [15] that supports the following operations on an array Q of size m in $O(\frac{\log m}{\log\log m})$ amortized time.

1. $\mathsf{access}(Q, i)$: returns $Q[i]$ for $1 \le i \le m$;
2. $\mathsf{rank}_a(Q, i)$: returns $|Q[\,:i]|_a$ for $1 \le i \le m$;
3. $\mathsf{select}_a(Q, i)$: returns i-th occurrence position of a for $1 \le i \le \mathsf{rank}_a(Q, m)$;
4. $\mathsf{insert}_a(Q, i)$: inserts a between $Q[i-1]$ and $Q[i]$ for $1 \le i \le m+1$;
5. $\mathsf{delete}(Q, i)$: deletes $Q[i]$ from Q for $1 \le i \le m$.

Corollary 1 implies that we can compute $LF_T(i)$ and its inverse $LF_T^{-1}(j)$ by

$$LF_T(i) = \mathsf{select}_x(\mathsf{F}_T, \mathsf{rank}_x(\mathsf{L}_T, i)) \quad \text{where} \quad x = \mathsf{L}_T[i],$$
$$LF_T^{-1}(j) = \mathsf{select}_y(\mathsf{L}_T, \mathsf{rank}_y(\mathsf{F}_T, j)) \quad \text{where} \quad y = \mathsf{F}_T[j].$$

3 Computing pBWT online

This section introduces our algorithm computing $pBWT_T$ in an online manner by reading a p-string T from right to left. Let $T = cS$ for $c \in \Sigma \cup \Pi \setminus \{\$\}$ and $n = |S| \ge 1$. We consider updating L_S to L_T. Hereafter, we assume that Σ is known and $|\Sigma| \le |T|$ as in [16]. Among the rows of the rotation matrices of S and T, the rows of $[\![S]\!] = [\![S_n]\!]$ and $[\![T]\!] = [\![T_{n+1}]\!]$ play important roles when updating. Let $k_S = RA_S^{-1}[n]$ and $k_T = RA_T^{-1}[n+1]$. We note that $\mathsf{L}_S[k_S] = \mathsf{L}_T[k_T] = \$$.

First, we observe RA_T is obtained from RA_S just by "inserting" $n+1$ at k_T.

Lemma 3. *For $1 \le i \le n+1$,*

$$RA_T[i] = \begin{cases} RA_S[i] & \text{if } i < k_T, \\ n+1 & \text{if } i = k_T, \\ RA_S[i-1] & \text{if } i > k_T. \end{cases}$$

Algorithm 1: PBWT update algorithm

1 **Function** $\mathsf{UpdateAll}(c, n, \mathsf{L}, \mathsf{F}, \mathsf{Right}, \mathsf{Left}, \mathsf{RM}, \mathsf{C}, \mathsf{LCP}^\infty)$
2 $\quad k = \mathsf{select}_\$(\mathsf{L}, 1)$; $\qquad // = k_S$
3 $\quad \mathsf{L}, \mathsf{F}, \mathsf{Right}, \mathsf{Left}, \mathsf{RM} = \mathsf{UpdateLF}(c, n, \mathsf{L}, \mathsf{F}, \mathsf{Right}, \mathsf{Left}, \mathsf{RM}, k)$;
$\quad // = \mathsf{L}^\circ_T, \mathsf{F}^\circ_T, \mathsf{Right}_T, \mathsf{Left}_T, \mathsf{RM}^\circ_T$
4 $\quad \mathsf{L}, \mathsf{F}, \mathsf{C}, k' = \mathsf{InsertRow}(n, \mathsf{L}, \mathsf{F}, \mathsf{C}, k)$; $\qquad // = \mathsf{L}_T, \mathsf{F}_T, \mathsf{C}_T, k_T$
5 $\quad$ **foreach** $a \in \Pi$ **do**
6 $\qquad$ **if** $\mathsf{RM}[a] \geq k'$ **then** $\mathsf{RM}[a] = \mathsf{RM}[a] + 1$; $\qquad // = \mathsf{RM}_T[a]$
7 $\quad x = \mathsf{UpdateLCP}(\mathsf{L}, \mathsf{F}, \mathsf{LCP}^\infty, k')$; $\qquad // = \mathsf{LCP}^\infty_T[k_T]$
8 $\quad \mathsf{LCP}^\infty[k'-1] = \mathsf{UpdateLCP}(\mathsf{L}, \mathsf{F}, \mathsf{LCP}^\infty, k'-1)$; $\qquad // = \mathsf{LCP}^\infty_T[k_T - 1]$
9 $\quad \mathsf{insert}_x(\mathsf{LCP}^\infty, k')$;
10 $\quad$ **return** $n + 1, \mathsf{L}, \mathsf{F}, \mathsf{Right}, \mathsf{Left}, \mathsf{RM}, \mathsf{C}, \mathsf{LCP}^\infty$;

Table 2: An example of our update step for $S = \texttt{xayzzazyza\$}$ and $T = \texttt{y}S$. The updated and inserted values are highlighted. In the arrays $\langle T_{RA_S[i]} \rangle$, updated/inserted values appear only after \$. Lemmas 3 and 8 are immediate consequences of this observation.

$\mathsf{F}_S[i]$	$\langle S_{RA_S[i]} \rangle$	$\mathsf{L}_S[i]$
\$	\$∞a∞∞1a252a	a
a	a\$∞a∞∞1a252	3
a	a∞∞1a252a\$∞	3
a	a∞∞2a\$∞a661	1
3	∞a\$∞a∞61a25	3
1	∞a2∞2a\$∞a66	1
3	∞a∞∞1a252a\$	\$
1	∞1a2∞2a\$∞a6	2
3	∞∞a\$∞a661a2	2
2	∞∞1a252a\$∞a	a
2	∞∞2a\$∞a661a	a

$\mathsf{F}^\circ_T[i]$	$\langle T_{RA_S[i]} \rangle$	$\mathsf{L}^\circ_T[i]$
\$	\$∞∞a3∞1a252a	a
a	a\$∞∞a3∞1a252	3
a	a∞∞1a252a\$4∞	3
a	a∞∞2a\$4∞a361	1
3	∞a\$∞∞a361a25	2
1	∞a2∞2a\$4∞a36	1
3	∞a∞∞1a252a\$4	2
1	∞1a2∞2a\$4∞a3	2
2	∞∞a\$4∞a361a2	2
2	∞∞1a252a\$4∞a	a
2	∞∞2a\$4∞a361a	a

$\mathsf{F}_T[i]$	$\langle T_{RA_T[i]} \rangle$	$\mathsf{L}_T[i]$
\$	\$∞∞a3∞1a252a	a
a	a\$∞∞a3∞1a252	3
a	a∞∞1a252a\$4∞	3
a	a∞∞2a\$4∞a361	1
3	∞a\$∞∞a361a25	2
1	∞a2∞2a\$4∞a36	1
3	∞a∞∞1a252a\$4	2
1	∞1a2∞2a\$4∞a3	2
2	∞∞a\$4∞a361a2	2
2	∞∞a3∞1a252a\$	\$
2	∞∞1a252a\$4∞a	a
2	∞∞2a\$4∞a361a	a

In the BWT, where S and T have no parameter characters, this implies that $\mathsf{L}_T[i] = T_{RA_T[i]}[n+1] = S_{RA_S[i]}[n] = \mathsf{L}_S[i]$ for $i < k_T$ and $\mathsf{L}_T[i+1] = T_{RA_T[i+1]}[n+1] = S_{RA_S[i]}[n] = \mathsf{L}_S[i]$ for $i > k_T$, except when $i = k_S$. Therefore, for computing L_T from L_S, we only need to update $\mathsf{L}_S[k_S] = \$$ to c and to find the position $k_T = RA_T^{-1}[n+1]$ where \$ should be inserted. However in the pBWT, $RA_T[i] = RA_S[i]$ does not necessarily imply that the values $\mathsf{L}_T[i] = [\![T_{RA_T[i]}]\!][n+1]$ and $\mathsf{L}_S[i] = [\![S_{RA_S[i]}]\!][n]$ coincide, since it is not always true that $[\![S]\!] = [\![T]\!][2:]$. So we also need to update the values of the encoding.

Algorithm 1 shows our update procedure, which maintains the array F and other auxiliary data structures in addition to L. After getting the key position k_S as the unique occurrence position of \$ in L_S at Line 2, to update the values of L and F from L_S and F_S to L_T and F_T, respectively, we compute intermediate arrays L°_T and F°_T of length n, which satisfy

$$\mathsf{L}^\circ_T[i] = [\![T_{RA_S[i]}]\!][n+1] \quad \text{and} \quad \mathsf{F}^\circ_T[i] = [\![T_{RA_S[i]}]\!][1]$$

for $1 \leq i \leq n$ using $\mathsf{UpdateLF}$ at Line 3. In other words, L°_T and F°_T are extracted from the last and the first columns of the $n \times (n+1)$ matrix $([\![T_{RA_S[i]}]\!])_{i=1}^n$, respectively, which can conceptionally be obtained by deleting the k_T-th row of the rotation sort matrix of T. We then find the other key position k_T and inserts appropriate values into L°_T and F°_T at k_T to turn them into L_T and F_T, respectively, by $\mathsf{InsertRow}$ at Line 4. The rest of the algorithm is devoted to maintaining some of the helper arrays. Particularly, a dedicated function $\mathsf{UpdateLCP}$ is used to update the ∞-LCP array. In the remainder of this section, we will explain those functions and involved auxiliary data structures in respective subsections. Table 2 shows an example of our 2-step update.

Algorithm 2: Computing L°_T and F°_T

```
Function UpdateLF(c, n, L, F, Right, Left, RM, k)
    if c ∈ Σ then L[k] = c;
    else
        foreach a ∈ Π with Left[a] ≠ 0 do
            // Computing L°_T[RM[a]] = F°_T[LF_S(RM[a])]
            i = RM[a];
            j = select_{L[i]}(F, rank_{L[i]}(L, i));                // j = LF_S(i)
            if a = c then
                cnt = 0;
                foreach b ∈ Π with Left[b] ≠ 0 do
                    if Left[a] ≥ Right[b] then cnt = cnt + 1;
            else
                cnt = L[i];
                if Left[c] = 0 or Left[a] > Left[c] ≥ Right[c] > Right[a] then
                    cnt = cnt + 1;
            L[i] = cnt; F[j] = cnt;
        // Computing L°_T[k_S] = ⟦T⟧[1]
        cnt = 1;
        if Left[c] = 0 then
            foreach a ∈ Π with Left[a] ≠ 0 do cnt = cnt + 1;
            Right[c] = n + 1; Left[c] = n + 1; RM[c] = k;
        else
            foreach a ∈ Π with Left[a] > Left[c] do cnt = cnt + 1;
            Left[c] = n + 1;
        L[k] = cnt;
    return L, F, Right, Left, RM;
```

3.1 Step 1: UpdateLF computes $\mathsf{L}^\circ_T[i]$ and $\mathsf{F}^\circ_T[i]$

When $c \in \Sigma$, computing L°_T and F°_T from L_S and F_S, respectively, is easy.

Lemma 4. *If $c \in \Sigma$, then for any $i \in \{1, \dots, n\}$, $\mathsf{F}^\circ_T[i] = \mathsf{F}_S[i]$ and $\mathsf{L}^\circ_T[i] = \mathsf{L}_S[i]$ except for $\mathsf{L}^\circ_T[k_S] = c$.*

Concerning the case $c \in \Pi$, first let us express the values of $[\![T]\!]$ using $[\![S]\!]$.

Lemma 5. *Suppose $c \in \Pi$.*

$$[\![T]\!][1] = \begin{cases} |\Pi \upharpoonright S| + 1 & \text{if } \mathit{Left}_S(c) = 0, \\ |\Pi \upharpoonright S[1 : \mathit{Left}_S(c)]| & \text{otherwise.} \end{cases}$$

For $1 \le p \le n$, if $S[p] \in \Sigma$ or $p \neq \mathit{Right}_S(S[p])$, then $[\![T]\!][p+1] = [\![S]\!][p]$. If $S[p] = a \in \Pi$ and $p = \mathit{Right}_S(a)$, then

$$[\![T]\!][p+1] = \begin{cases} |\Pi \upharpoonright S[p+1 : n]| + 1 & \text{if } a = c, \\ [\![S]\!][p] + 1 & \text{if } \mathit{Left}_S(c) = 0 \text{ or} \\ & \quad \mathit{Left}_S(a) < \mathit{Left}_S(c) \le \mathit{Right}_S(c) < \mathit{Right}_S(a), \\ [\![S]\!][p] & \text{otherwise.} \end{cases}$$

Based on Lemmas 4 and 5, Algorithm 2 computes $\mathsf{F}^\circ_T[i]$ and $\mathsf{L}^\circ_T[i]$ from $\mathsf{F}_S[i]$ and $\mathsf{L}_S[i]$, as well as other auxiliary data structures. Note that, since the intermediate matrix $(T_{RA_S[i]})_{i=1}^n$ misses a row corresponding to $[\![T]\!]$, the value $[\![T]\!][1]$ does not matter for F°_T, whereas it appears as $\mathsf{L}^\circ_T[k_S] = \mathsf{L}^\circ_T[RA_S^{-1}[n]]$. When $c \in \Pi$, Lemma 5 implies that, other than $\mathsf{L}^\circ_T[k_S] = [\![T]\!][1]$, we only need to update the values at the positions in L and F corresponding to the rightmost occurrence position $p = \mathit{Right}_S(a)$ of each parameter character $a \in \Pi$ in S. By rotating S to the right by $n - p$, that occurrence comes to the right end and appears in the pBWT. That is, the array L needs to be updated only at i such that $RA_S[i] = n - \mathit{Right}_S(a)$. The algorithm maintains such position i as $\mathsf{RM}_S[a]$ for each $a \in \Pi \upharpoonright S$, i.e. $\mathsf{RM}_S[a] = RA_S^{-1}[n - \mathit{Right}_S(a)]$. Similarly, we only need to update F at $LF_S(\mathsf{RM}_S[a])$, where $\mathsf{F}^\circ_T[LF_S(\mathsf{RM}_S[a])] = \mathsf{L}^\circ_T[\mathsf{RM}_S[a]]$. In our algorithm, as alternatives of Left_S and Right_S, we maintain two arrays Left and Right that store the leftmost and rightmost occurrence positions of parameter characters counting *from the right end*, respectively, i.e., $\mathsf{Left}_S[a] = \mathit{Right}_{\overline{S}}(a)$ and $\mathsf{Right}_S[a] = \mathit{Left}_{\overline{S}}(a)$ for each $a \in \Pi$, where $\overline{S}$ is the reverse of S.

Algorithm 2 also updates RM to RM°_T, which indicates the row of L°_T corresponding to the rightmost occurrence of each parameter character in T. That is, $\mathsf{RM}^\circ_T[a] = i$ iff $RA_S[i] = \mathsf{Right}_T[a]$, as long as a occurs in T. When $c \in \Pi$ and it appears in the text for the first time, we have $\mathsf{RM}^\circ_T[c] = k_S$ (Line 19). Other than that, $\mathsf{RM}^\circ_T[a] = \mathsf{RM}_S[a]$ for every $a \in \Pi$.

Lemma 6. *Algorithm 2 computes* $\mathsf{L}^\circ_T[i]$, $\mathsf{F}^\circ_T[i]$, Right_T, Left_T, *and* RM°_T *in* $O(|\Pi|\frac{\log n}{\log\log n})$ *amortized time.*

3.2 Step 2: $\mathsf{InsertRow}$ computes L_T and F_T

To transform F°_T and L°_T into F_T and L_T, we insert the values $[\![T]\!][1]$ and $[\![T]\!][n+1]$ at the position k_T, respectively. We know those values as $[\![T]\!][1] = \mathsf{L}^\circ_T[k_S]$ and $[\![T]\!][n+1] = \$$. Therefore, it is enough to discuss how to find the position k_T.

In the case $c \in \Sigma$, the position k_T can be calculated similarly to the case of BWT for static strings thanks to Corollary 1. Define $\Sigma_{<b} = |\{\, a \in \Sigma \mid a < b \,\}|$.

Lemma 7. *If* $c \in \Sigma$, $k_T = |T|_{\Sigma_{<c}} + |\{\, i \mid \mathsf{L}^\circ_T[i] = c,\ 1 \le i \le k_S \,\}|$.

In the case $c \in \Pi$, we will use Lemma 2 for finding k_T in Lemma 9 below. We first observe that one can use LCP^∞_S to calculate $lcp^\infty(\langle T_p\rangle, \langle T_q\rangle)$ for most cases.

Lemma 8. *For* $1 \le p < q \le n$, $lcp^\infty(\langle T_p\rangle, \langle T_q\rangle) = lcp^\infty(\langle S_p\rangle, \langle S_q\rangle)$.

Lemma 9. *Suppose* $c \in \Pi$. *Let* $\ell_i = lcp^\infty(\langle S_{RA_S[i]}\rangle, \langle S_{RA_S[k_S]}\rangle)$ *for* $1 \le i \le n$. *Then,*

$$
\begin{aligned}
k_T = 1 + |T|_\Sigma & \\
&+ |\{\, i \mid 1 \le \mathsf{L}^\circ_T[i] \le \mathsf{L}^\circ_T[k_S],\ 1 \le i < k_S \,\}| && (1)\\
&+ |\{\, i \mid \ell_i < \mathsf{L}^\circ_T[k_S] < \mathsf{L}^\circ_T[i],\ 1 \le i < k_S \,\}| && (2)\\
&+ |\{\, i \mid 1 \le \mathsf{L}^\circ_T[i] \le \min\{\mathsf{L}^\circ_T[k_S] - 1, \ell_i\},\ k_S < i \le n \,\}|\,. && (3)
\end{aligned}
$$

Proof. By definition,

$$k_T = 1 + |T|_\Sigma + |\{\, j \mid \mathsf{F}_T[j] \in \mathbb{N}_+,\ 1 \le j < k_T \,\}|\,,$$

of which we focus on the last term. Let $h = LF_T^{-1}(k_T)$ and $m_i = lcp^\infty(\langle T_{RA_T[i]}\rangle, \langle T_{RA_T[h]}\rangle)$ for $1 \le i \le n+1$. By Lemma 2, $\mathsf{F}_T[j] \in \mathbb{N}_+$ and $1 \le j < k_T$ iff for $i = LF_T^{-1}(j)$, either

1. $1 \le i < h$ and $1 \le \mathsf{L}_T[i] \le \mathsf{L}_T[h]$,
2. $1 \le i < h$ and $m_i < \mathsf{L}_T[h] < \mathsf{L}_T[i]$, or
3. $h < i \le n+1$ and $1 \le \mathsf{L}_T[i] \le \min\{\mathsf{L}_T[h] - 1,\, m_i\}$.

Those three cases are mutually exclusive. Let $m^\circ_i = lcp^\infty(\langle T_{RA_S[i]}\rangle, \langle T_{RA_S[k_S]}\rangle)$. Counting each of the above cases is equivalent to counting i such that

Algorithm 3: Inserting $[\![T]\!][1]$ to F and $[\![T]\!][n+1]$ to L

1 **Function** $\mathsf{InsertRow}(n, \mathsf{L}, \mathsf{F}, \mathsf{C}, k)$
2    $x = \mathsf{L}[k]$; $\quad$ *// $= [\![T]\!][1]$*
3   **if** $x \in \Sigma$ **then**
4      $k' = \mathsf{select}_x(\mathsf{C}, 1) - |\Sigma_{<x}| - 1 + \mathsf{rank}_x(\mathsf{L}, k)$; $\quad$ *// $k_T = |S|_{\Sigma_{<c}} + |\{\, i \mid \mathsf{L}^\circ_T[i] = c,\ 1 \le i \le k_S \,\}|$*
5      $\mathsf{insert}_x(\mathsf{C}, \mathsf{select}_x(\mathsf{C}, 1))$;
6   **else**
7      $k' = 1 + |\mathsf{C}| - |\Sigma|$; $\quad$ *// $= 1 + |S|_\Sigma$*
8   **for** $y = 1$ **to** x **do** $k' = k' + \mathsf{rank}_y(\mathsf{L}, k-1)$; $\quad$ *// Term (1)*
9      $j = 0$;
10   **for** $y = 0$ **to** $x - 1$ **do**
11   **if** $\mathsf{rank}_y(\mathsf{LCP}^\infty, k-1) \neq 0$ **then**
12          $j = \max\{j, \mathsf{select}_y(\mathsf{LCP}^\infty, \mathsf{rank}_y(\mathsf{LCP}^\infty, k-1))\}$;
          // $j = \max(\{j\} \cup \{\, i \mid \mathsf{LCP}^\infty[i] = y$ and $1 \le i < k_S \,\})$
13   **for** $y = x + 1$ **to** $|\Pi|$ **do** $k' = k' + \mathsf{rank}_y(\mathsf{L}, j)$; $\quad$ *// Term (2)*
14      $j = n$; $\quad$ *// $j_0 = n$*
15   **for** $y = 1$ **to** $x - 1$ **do** $\quad$ *// Term (3)*
16   **if** $\mathsf{rank}_{y-1}(\mathsf{LCP}^\infty, k-1) < \mathsf{rank}_{y-1}(\mathsf{LCP}^\infty, n)$ **then**
17          $j = \min\{j, \mathsf{select}_{y-1}(\mathsf{LCP}^\infty, \mathsf{rank}_{y-1}(\mathsf{LCP}^\infty, k-1) + 1)\}$;
          // $j_y = \min(\{j_{y-1}\} \cup \{\, i \mid \mathsf{LCP}^\infty[i] = y - 1$ and $k_S \le i \le n \,\})$
18          $k' = k' + \mathsf{rank}_y(\mathsf{L}, j-1) - \mathsf{rank}_y(\mathsf{L}, k)$;
19    $\mathsf{insert}_\$(\mathsf{L}, k')$; $\mathsf{insert}_x(\mathsf{F}, k')$;
20   **return** $\mathsf{L}, \mathsf{F}, \mathsf{C}, k'$;

1. $1 \le i < k_S$ and $1 \le \mathsf{L}^\circ_T[i] \le \mathsf{L}^\circ_T[k_S]$,
2. $1 \le i < k_S$ and $m^\circ_i < \mathsf{L}^\circ_T[k_S] < \mathsf{L}^\circ_T[i]$, or
3. $k_S < i \le n$ and $1 \le \mathsf{L}^\circ_T[i] \le \min\{\mathsf{L}^\circ_T[k_S] - 1, m^\circ_i\}$.

This is because the matrix $([\![T_{RA_S[i]}]\!])_{i=1}^n$ can conceptionally be obtained by removing the k_T-th row of the matrix of $([\![T_{RA_T[i]}]\!])_{i=1}^{n+1}$, where the row k_S of $([\![T_{RA_S[i]}]\!])_{i=1}^n$ corresponds to the row h of $([\![T_{RA_T[i]}]\!])_{i=1}^{n+1}$ in particular ($RA_S[k_S] = RA_T[h] = n$), and $i = k_T = RA_T^{-1}[n+1]$ is not counted due to $T[n+1] = \$ \in \Sigma$.

Lemma 8 implies $m^\circ_i = \ell_i$, which completes the proof. □

Based on Lemmas 7 and 9, Algorithm 3 finds the key position k_T.

For handling the case $c \in \Sigma$, we maintain a dynamic array C by which one can obtain the value $|T|_{\Sigma_{<c}} = |S|_{\Sigma_{<c}}$ quickly. The array C_S can be seen as a string of the form $\mathsf{C}_S = a_1^{|S|_{a_1}+1} \ldots a_\sigma^{|S|_{a_\sigma}+1}$, where $a_1, \ldots, a_\sigma$ enumerate the static characters of Σ in the lexicographic order ($\sigma = |\Sigma|$) and a^s denotes the sequence of a of length s. Then, $|T|_{\Sigma_{<c}} = \mathsf{select}_c(\mathsf{C}_S, 1) - |\Sigma_{<c}| - 1$. The other term $|\{\, i \mid \mathsf{L}^\circ_T[i] = c,\ 1 \le i \le k_S \,\}|$ in Lemma 7 is calculated as $\mathsf{rank}_c(\mathsf{L}^\circ_T, k_S)$. We remark C_S has $a^{|S|_a+1}$ rather than $a^{|S|_a}$ so that $\mathsf{select}_c(\mathsf{C}, 1)$ is always defined.

Suppose $c \in \Pi$. The term $|T|_\Sigma$ of the equation of Lemma 9 is calculated as $|T|_\Sigma = |S|_\Sigma = |\mathsf{C}| - |\Sigma|$. Let $x = \mathsf{L}^\circ_T[k_S]$. Term (1) is obtained at Line 8 by

$$(1) = \textstyle\sum_{y=1}^{x} \mathsf{rank}_y(\mathsf{L}^\circ_T, k_S - 1)\,.$$

Concerning Term (2), we first find the range of $i < k_S$ satisfying $\ell_i < x$. By Lemma 1, $\ell_i = \min_{i \le j < k_S} \mathsf{LCP}^\infty_S[j]$. Thus, for any $i < k_S$, $\ell_i < x$ iff $i \le j_* = \max\{\, j \mid \mathsf{LCP}^\infty_S[j] < x,\ j < k_S \,\}$.

The **for** loop of Line 10 computes such j_*. Then, (2) is computed at Line 13 as

$$(2) = |\{\, i \mid x < \mathsf{L}^{\circ}_T[i],\ 1 \le i \le j_* \,\}| = \textstyle\sum_{y=x+1}^{|\Pi|} \mathsf{rank}_y(\mathsf{L}^{\circ}_T, j_*)\,.$$

We compute Term (3) by summing up the numbers of positions $i > k_S$ such that $\mathsf{L}^{\circ}_T[i] = y \le \ell_i$ for all $y = 1, \dots, x-1$ in the **for** loop of Line 15. To this end, we find the range of $i > k_S$ such that $\ell_i \ge y$. By Lemma 1, $\ell_i = \min_{k_S \le j < i} \mathsf{LCP}^{\infty}_S[j]$. Thus, for every $i > k_S$, $\ell_i \ge y$ iff $i < j_y = \min\{\, j \mid \mathsf{LCP}^{\infty}_S[j] < y,\ k_S \le j \le n \,\}$. Note that $j_y = \min(\{j_{y-1}\} \cup \{\, j \mid \mathsf{LCP}^{\infty}_S[j] = y-1,\ k_S \le j \le n \,\})$ for any $y \ge 1$ assuming $j_0 = n$. Line 17 computes j_y as the first occurrence of $y-1$ after those in $\mathsf{LCP}^{\infty}[1 : k_S - 1]$. Then, (3) is calculated by

$$\begin{aligned}(3) &= |\{\, i \mid 1 \le \mathsf{L}^{\circ}_T[i] \le x-1,\ k_S < i < j_y \,\}| \\ &= \textstyle\sum_{y=1}^{x-1} \left(\mathsf{rank}_y(\mathsf{L}^{\circ}_T, j_y - 1) - \mathsf{rank}_y(\mathsf{L}^{\circ}_T, k_S)\right)\,.\end{aligned}$$

Lemma 10. *Algorithm 3 computes L_T, F_T, C_T, and k_T in $O(|\Pi|\frac{\log n}{\log\log n})$ amortized time.*

3.3 Step 3: Updating LCP^{∞} by $\mathsf{UpdateLCP}$

What remains to do is updating the arrays RM and LCP^{∞}. On the one hand, updating RM from RM°_T to RM_T is easy. $\mathsf{RM}^{\circ}_T[a]$ should be incremented by one just if $\mathsf{RM}^{\circ}_T[a] \ge k_T$. Otherwise, $\mathsf{RM}_T[a] = \mathsf{RM}^{\circ}_T[a]$. On the other hand, Lemma 8 implies LCP^{∞}_T is almost identical to LCP^{∞}_S.

Corollary 2. $\mathsf{LCP}^{\infty}_T[i] = \mathsf{LCP}^{\infty}_S[i]$ *if* $i < k_T - 1$, *and* $\mathsf{LCP}^{\infty}_T[i] = \mathsf{LCP}^{\infty}_S[i-1]$ *if* $i > k_T$.

By Corollary 2, we only need to compute $\mathsf{LCP}^{\infty}_T[k_T - 1]$ and $\mathsf{LCP}^{\infty}_T[k_T]$, to which Lemma 8 cannot directly be applied. The following lemma allows us to reduce the calculation of $\mathsf{LCP}^{\infty}_T[k] = lcp^{\infty}(\langle T_{RA_T[k]}\rangle, \langle T_{RA_T[k+1]}\rangle)$ to that of $lcp^{\infty}(\langle T_{RA_T[LF_T^{-1}(k)]}\rangle, \langle T_{RA_T[LF_T^{-1}(k+1)]}\rangle)$, to which Lemma 8 may be applied.

Lemma 11. *Let $1 \le i, j \le n+1$, $p = RA_T[i]$, $q = RA_T[j]$, $\ell = lcp^{\infty}(\langle T_p\rangle, \langle T_q\rangle)$, $i' = LF_T^{-1}(i)$, $j' = LF_T^{-1}(j)$, $p' = RA_T[i']$, $q' = RA_T[j']$, and $\ell' = lcp^{\infty}(\langle T_{p'}\rangle, \langle T_{q'}\rangle)$.*

1. *If $\mathsf{F}_T[i] = \mathsf{F}_T[j] \in \Sigma$, then $\ell = \ell'$.*
2. *If $\mathsf{F}_T[i] \ne \mathsf{F}_T[j]$ and either $\mathsf{F}_T[i] \in \Sigma$ or $\mathsf{F}_T[j] \in \Sigma$, then $\ell = 0$.*
3. *If $\mathsf{F}_T[i], \mathsf{F}_T[j] \in \mathbb{N}_+$, then*

$$\ell = \begin{cases} \ell' + 1 & \text{if } \ell' < \min\{\mathsf{F}_T[i], \mathsf{F}_T[j]\}, \\ \ell' & \text{if } \ell' \ge \mathsf{F}_T[i] = \mathsf{F}_T[j], \\ \min\{\mathsf{F}_T[i], \mathsf{F}_T[j]\} & \text{otherwise.} \end{cases}$$

Proof. Let $T_p = aU$, $T_q = bV$, $T_{p'} = Ua$ and $T_{q'} = Vb$.

1. If $a = b \in \Sigma$, then $\ell = \ell' = lcp^{\infty}(\langle U\rangle, \langle V\rangle)$.

2. In the case $a \ne b$ and $\{a, b\} \cap \Sigma \ne \emptyset$, clearly $\langle T_p\rangle[1] \ne \langle T_q\rangle[1]$. Thus $\ell = 0$.

3. In the case $a, b \in \Pi$, let W be the longest common prefix of $\langle T_{p'}\rangle$ and $\langle T_{q'}\rangle$, u and v be the first occurrence positions of a in $T_{p'}$ and b in $T_{q'}$, respectively, and w be the ℓ'-th occurrence position of ∞ in W.

Suppose $\ell' < \min\{\mathsf{F}_T[i], \mathsf{F}_T[j]\} = \min\{\mathsf{L}_T[i'], \mathsf{L}_T[j']\}$. That is, $|W|_{\infty} < \min\{|\langle T_{p'}\rangle[:u]|_{\infty}, |\langle T_{q'}\rangle[:v]|_{\infty}\}$. This means $|W| < \min\{u, v\}$ and thus ∞W is the longest common prefix of $\langle T_p\rangle$ and $\langle T_q\rangle$. Thus, we have $\ell = |\infty W|_{\infty} = \ell' + 1$.

Suppose $\ell' \ge \mathsf{F}_T[i] = \mathsf{F}_T[j]$, i.e., $|W|_{\infty} \ge |\langle T_{p'}\rangle[:u]|_{\infty} = |\langle T_{q'}\rangle[:v]|_{\infty}$. Then $|W[:u]|_{\infty} = |W[:v]|_{\infty}$ and $W[u] = W[v] = \infty$ implies $u = v$. Let Z be the longest common prefix of $\langle T_p\rangle$ and $\langle T_q\rangle$. Then, W and Z can be written as $W = X\infty Y$ and $Z = \infty X u Y$, where $|X| = u - 1$. Therefore, $\ell = \ell'$.

Algorithm 4: Updating $\mathsf{LCP}^\infty[i]$

Function $\mathsf{UpdateLCP}(\mathsf{L}, \mathsf{F}, \mathsf{LCP}^\infty, i)$
  $j = i + 1$; $x = 0$;
  if $\mathsf{F}[i] = \mathsf{F}[j]$ or $\mathsf{F}[i], \mathsf{F}[j] \in \mathbb{N}_+$ **then**
    $i' = \mathsf{select}_{\mathsf{F}[i]}(\mathsf{L}, \mathsf{rank}_{\mathsf{F}[i]}(\mathsf{F}, i))$; // $i' = LF_T^{-1}(i)$
    $j' = \mathsf{select}_{\mathsf{F}[j]}(\mathsf{L}, \mathsf{rank}_{\mathsf{F}[j]}(\mathsf{F}, j))$; // $j' = LF_T^{-1}(i+1)$
    for $y = |\Pi|$ **downto** 0 **do**
      if $\mathsf{rank}_y(\mathsf{LCP}^\infty, i'-1) \neq \mathsf{rank}_y(\mathsf{LCP}^\infty, j'-1)$ **then** $x = y$;
    // $x = lcp^\infty(\langle S_{RA_S[i']}\rangle, \langle S_{RA_S[j']}\rangle)$
    if $\mathsf{F}[i], \mathsf{F}[j] \in \mathbb{N}_+$ **then**
      if $x < \min\{\mathsf{F}[i], \mathsf{F}[j]\}$ **then** $x = x + 1$;
      else if $\mathsf{F}[i] \neq \mathsf{F}[j]$ **then** $x = \min\{\mathsf{F}[i], \mathsf{F}[j]\}$;
  return x;

Otherwise, $\mathsf{F}_T[i] \neq \mathsf{F}_T[j]$ and $\ell' \geq \min\{\mathsf{F}_T[i], \mathsf{F}_T[j]\}$. Assume $\mathsf{F}_T[i] < \mathsf{F}_T[j]$ (the case $\mathsf{F}_T[j] < \mathsf{F}_T[i]$ is symmetric). Then $u \leq |W|$. Moreover, we have $u < v$, since otherwise, $\langle T_{q'}\rangle[\colon v]$ had to be a prefix of $\langle T_{p'}\rangle[\colon u]$, which is impossible by $\mathsf{F}_T[i] < \mathsf{F}_T[j]$. Let $\langle T_p[\colon |W|+1]\rangle = \infty X u Y$, where $|X| = u - 1$. Then we have $\langle T_q[\colon |W|+1]\rangle = \infty X \infty Y'$ for some $Y' \in (\Sigma \cup \mathbb{N}_\infty)^*$. Thus $\ell = |\infty X|_\infty = \mathsf{F}_T[i]$. $\square$

One can compute $\ell' = lcp^\infty(\langle T_{p'}\rangle, \langle T_{q'}\rangle)$ in Lemma 11 for $1 \leq p' < q' \leq n$ using Lemmas 8 and 1 as

$$\begin{aligned} lcp^\infty(\langle T_{p'}\rangle, \langle T_{q'}\rangle) = lcp^\infty(\langle S_{p'}\rangle, \langle S_{q'}\rangle) &= \min\{\, \mathsf{LCP}_S^\infty[h] \mid i' \leq h < j' \,\} \\ &= \min(\{0\} \cup \{\, y \mid \mathsf{rank}_y(\mathsf{LCP}_S^\infty, i'-1) \neq \mathsf{rank}_y(\mathsf{LCP}_S^\infty, j'-1) \,\})\,. \end{aligned}$$

Finally, when $q' = n + 1$, we have $\mathsf{F}_T[j] = \$ \neq \mathsf{F}_T[i]$, and thus $lcp^\infty(\langle T_p\rangle, \langle T_q\rangle) = 0$. Algorithm 4 computes $\mathsf{LCP}_T^\infty[i]$ using F_T, L_T, and LCP_S^∞.

Lemma 12. *Algorithm 4 computes* $\mathsf{LCP}_T^\infty[i]$ *in* $O(|\Pi|\frac{\log n}{\log\log n})$ *amortized time.*

By Lemmas 6, 10, and 12, we have the following theorem.

Theorem 1. *Given* $c \in \Sigma \cup \Pi$, $n = |S|$, $\mathsf{L} = \mathsf{L}_S$, $\mathsf{F} = \mathsf{F}_S$, $\mathsf{Right} = \mathsf{Right}_S$, $\mathsf{Left} = \mathsf{Left}_S$, $\mathsf{RM} = \mathsf{RM}_S$, $\mathsf{C} = \mathsf{C}_S$, *and* $\mathsf{LCP}^\infty = \mathsf{LCP}_S^\infty$ *for some* $S \in (\Sigma \cup \Pi)^*$, *Algorithm 1 computes* $|T|$, L_T, F_T, Right_T, Left_T, RM_T, C_T, *and* LCP_T^∞ *for* $T = cS$ *in* $O(|\Pi|\frac{\log n}{\log\log n})$ *amortized time per input character.*

Corollary 3. *For a p-string T of length n, $pBWT_T$ can be computed in an online manner by reading T from right to left in $O(n|\Pi|\frac{\log n}{\log\log n})$ time.*

References

[1] Brenda S. Baker. A theory of parameterized pattern matching: algorithms and applications. In *Proceedings of the twenty-fifth annual ACM symposium on Theory of Computing (STOC 1993)*, pages 71–80, 1993.

[2] Brenda S. Baker. Parameterized pattern matching: Algorithms and applications. *Journal of Computer and System Sciences*, 52(1):28–42, 1996.

[3] Michael Burrows and David Wheeler. A block-sorting lossless data compression algorithm. Technical Report 124, Digital Equipment Corporation, 1994.

[4] Diptarama, Takashi Katsura, Yuhei Otomo, Kazuyuki Narisawa, and Ayumi Shinohara. Position heaps for parameterized strings. In *Proceedings of the 28th Annual Symposium on Combinatorial Pattern Matching (CPM 2017)*, pages 8:1–8:13, 2017.

[5] Kimmo Fredriksson and Maxim Mozgovoy. Efficient parameterized string matching. *Information Processing Letters*, 100(3):91 – 96, 2006.

[6] Noriki Fujisato, Yuto Nakashima, Shunsuke Inenaga, Hideo Bannai, and Masayuki Takeda. Right-to-left online construction of parameterized position heaps. In *Proceedings of the Prague Stringology Conference 2018 (PSC 2018)*, pages 91–102, 2018.

[7] Noriki Fujisato, Yuto Nakashima, Shunsuke Inenaga, Hideo Bannai, and Masayuki Takeda. Direct linear time construction of parameterized suffix and LCP arrays for constant alphabets. In *Proceedings of the 26th International Symposium on String Processing and Information Retrieval (SPIRE 2019)*, pages 382–391, 2019.

[8] Arnab Ganguly, Rahul Shah, and Sharma V. Thankachan. pBWT: Achieving succinct data structures for parameterized pattern matching and related problems. In *Proceedings of the 28th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2017)*, pages 397–407, 2017.

[9] Arnab Ganguly, Rahul Shah, and Sharma V. Thankachan. Fully functional parameterized suffix trees in compact space. In *Proceedings of the 49th International Colloquium on Automata, Languages, and Programming (ICALP 2022)*, pages 65:1–65:18, 2022.

[10] Tomohiro I, Satoshi Deguchi, Hideo Bannai, Shunsuke Inenaga, and Masayuki Takeda. Lightweight parameterized suffix array construction. In *Proceedings of the 20th International Workshop on Combinatorial Algorithms (IWOCA 2009)*, pages 312–323, 2009.

[11] Sung-Hwan Kim and Hwan-Gue Cho. Simpler FM-index for parameterized string matching. *Information Processing Letters*, 165:106026, 2021.

[12] Juan Mendivelso, Sharma V. Thankachan, and Yoan J. Pinzón. A brief history of parameterized matching problems. *Discrete Applied Mathematics*, 274:103–115, 2020.

[13] Katsuhito Nakashima, Noriki Fujisato, Diptarama Hendrian, Yuto Nakashima, Ryo Yoshinaka, Shunsuke Inenaga, Hideo Bannai, Ayumi Shinohara, and Masayuki Takeda. DAWGs for parameterized matching: Online construction and related indexing structures. In *Proceedings of the 31st Annual Symposium on Combinatorial Pattern Matching (CPM 2020)*, pages 26:1–26:14, 2020.

[14] Katsuhito Nakashima, Diptarama Hendrian, Ryo Yoshinaka, and Ayumi Shinohara. An Extension of Linear-size Suffix Tries for Parameterized Strings. In *SOFSEM 2020 Student Research Forum*, pages 97–108, 2020.

[15] Gonzalo Navarro and Yakov Nekrich. Optimal dynamic sequence representations. *SIAM Journal on Computing*, 43(5):1781–1806, 2014.

[16] Alberto Policriti and Nicola Prezza. Fast online Lempel–Ziv factorization in compressed space. In *Proceedings of the the 22nd International Symposium on String Processing and Information Retrieval (SPIRE 2015)*, pages 13–20, 2015.

[17] Tetsuo Shibuya. Generalization of a suffix tree for RNA structural pattern matching. *Algorithmica*, 39(1):1–19, 2004.

[18] Sharma V. Thankachan. Compact text indexing for advanced pattern matching problems: Parameterized, order-isomorphic, 2D, etc. (invited talk). In *Proceedings of the 33rd Annual Symposium on Combinatorial Pattern Matching (CPM 2022)*, pages 3:1–3:3, 2022.

A Proofs

Proposition 1. *For any p-strings S and T, $S \approx T$ if and only if $[\![S]\!] = [\![T]\!]$.*

Proof. For simplicity, assume that S and T contain no static character. Suppose $S \approx T$. Since $S[i] = S[j]$ iff $T[i] = T[j]$ for any indices i, j, we have

$$[\![S]\!][i] = |\Pi \upharpoonright S_{n-i}[1 : \mathit{Left}_{S_{n-i}}(S[i])]| = |\Pi \upharpoonright T_{n-i}[1 : \mathit{Left}_{T_{n-i}}(T[i])]| = [\![T]\!][i]\,.$$

for all i.

Suppose $S \not\approx T$. Let i be the leftmost position such that $\langle S\rangle[i] \neq \langle T\rangle[i]$. We may assume without loss of generality that $\langle S\rangle[i] < \langle T\rangle[i]$. Let $j = i - \langle S\rangle[i]$. Then,

$$[\![S]\!][j] = |\Pi \upharpoonright S[j+1 : i]| = |\Pi \upharpoonright S[j : i-1]| = |\Pi \upharpoonright T[j : i-1]|\,,$$

since $S[j : i-1] \approx T[j : i-1]$. The fact $S[j] \notin \Pi \upharpoonright S[j+1 : i-1]$ implies $T[j] \notin \Pi \upharpoonright T[j+1 : i-1]$. Moreover, $\langle S\rangle[i] < \langle T\rangle[i]$ implies $T[i] \notin \Pi \upharpoonright T[j : i-1]$. Hence,

$$[\![T]\!][j] \geq |(\Pi \upharpoonright T[j : i-1]) \cup \{T[i]\}| > |\Pi \upharpoonright T[j : i-1]| = [\![S]\!][j]\,.$$

Lemma 3 is a corollary to the following lemma.

Lemma 13. *For any i and j such that $1 \leq i < j \leq n$, $RA_S^{-1}[i] < RA_S^{-1}[j]$ iff $RA_T^{-1}[i] < RA_T^{-1}[j]$.*

Proof. Let $S_i = U\$V$ and $S_j = XY\$Z$, where $|U\$| = |X| = i < j \leq n$. We have $T_i = U\$cV$ and $T_j = XY\$cZ$. Since \$ does not occur in X, $\langle U\$\rangle \neq \langle X\rangle$. Thus,

$$RA_S^{-1}[i] < RA_S^{-1}[j] \iff \langle U\$\rangle < \langle X\rangle \iff RA_T^{-1}[i] < RA_T^{-1}[j]\,.$$

Lemma 4. *If $c \in \Sigma$, then for any $i \in \{1, \dots, n\}$, $\mathsf{F}^\circ_T[i] = \mathsf{F}_S[i]$ and $\mathsf{L}^\circ_T[i] = \mathsf{L}_S[i]$ except for $\mathsf{L}^\circ_T[k_S] = c$.*

Proof. If $c \in \Sigma$, then $[\![S]\!] = [\![T]\!][2 :]$ by definition. So, for any $i \in \{1, \dots, n\}$,

$$\mathsf{F}^\circ_T[i] = [\![T_{RA_S[i]}]\!][1] = [\![S_{RA_S[i]}]\!][1] = \mathsf{F}_S[i]$$

and

$$\mathsf{L}^\circ_T[i] = [\![T_{RA_S[i]}]\!][n+1] = \begin{cases} [\![S_{RA_S[i]}]\!][n] = \mathsf{L}_S[i] & \text{if } RA_S[i] \neq n, \\ c & \text{if } RA_S[i] = n. \end{cases}$$

∎

Lemma 5. *Suppose $c \in \Pi$.*

$$[\![T]\!][1] = \begin{cases} |\Pi \upharpoonright S| + 1 & \textit{if } \mathit{Left}_S(c) = 0, \\ |\Pi \upharpoonright S[1 : \mathit{Left}_S(c)]| & \textit{otherwise.} \end{cases}$$

For $1 \leq p \leq n$, if $S[p] \in \Sigma$ or $p \neq \mathit{Right}_S(S[p])$, then $[\![T]\!][p+1] = [\![S]\!][p]$. If $S[p] = a \in \Pi$ and $p = \mathit{Right}_S(a)$, then

$$[\![T]\!][p+1] = \begin{cases} |\Pi \upharpoonright S[p+1 : n]| + 1 & \textit{if } a = c, \\ [\![S]\!][p] + 1 & \textit{if } \mathit{Left}_S(c) = 0 \textit{ or} \\ & \quad \mathit{Left}_S(a) < \mathit{Left}_S(c) \leq \mathit{Right}_S(c) < \mathit{Right}_S(a), \\ [\![S]\!][p] & \textit{otherwise.} \end{cases}$$

Proof. The claim on value of $[\![T]\!][1]$ is clear by definition. For $p \geq 1$, if $T[p+1] = S[p] \in \Sigma$, then $[\![T]\!][p+1] = [\![S]\!][p]$.

Let us consider the case $S[p] = a \in \Pi$. If $p \neq \mathit{Right}_S(a)$, then a occurs somewhere after p in S. Let $q > p$ be the first occurrence position of a after p in S. By definition, $[\![S]\!][p] = |\Pi \upharpoonright S[p+1:q]| = |\Pi \upharpoonright T[p+2:q+1]| = [\![T]\!][p+1]$.

Suppose $p = \mathit{Right}_S(a)$ for some $a \in \Pi$. If $a = c$, since $T[1] = c$ and $c \notin \Pi \upharpoonright S[p+1:n]$, we have $[\![T]\!][p+1] = |\Pi \upharpoonright S[p+1:n] \cup \{c\}| = |\Pi \upharpoonright S[p+1:n]| + 1$. If $\mathit{Right}_S(c) = 0$ or $\mathit{Left}_S(a) < \mathit{Left}_S(c) \leq \mathit{Right}_S(c) < \mathit{Right}_S(a) = p$, $[\![S]\!][p]$ counts the number of distinct p-characters in $S[p{+}1:]S[:\mathit{Left}_S(a)]$, where c does not occur. On the other hand, $[\![T]\!][p+1]$ counts the ones in $S[p+1:]cS[:\mathit{Left}_S(a)]$. That is, $[\![T]\!][p+1] = [\![S]\!][p] + 1$. Otherwise, if $\mathit{Left}_S(c) < \mathit{Left}_S(a)$ or $\mathit{Right}_S(a) < \mathit{Right}_S(c)$, we already have c in $S[p+1:]S[:\mathit{Left}_S(a)]$. Thus $[\![T]\!][p+1] = [\![S]\!][p]$. □

Lemma 7. *If $c \in \Sigma$, $k_T = |T|_{\Sigma_{<c}} + |\{\, i \mid \mathsf{L}^{\circ}_T[i] = c,\ 1 \leq i \leq k_S \,\}|$.*

Proof. By definition, $k_T = |T|_{\Sigma_{<c}} + |\{\, j \mid \mathsf{F}_T[j] = c,\ 1 \leq j \leq k_T \,\}|$. By Corollary 1 and the bijectivity of LF_T, the second term equals

$$|\{\, i \mid \mathsf{L}_T[i] = c,\ 1 \leq i \leq LF_T^{-1}(k_T) \,\}|$$

and further more equals

$$|\{\, i \mid \mathsf{L}^{\circ}_T[i] = c,\ 1 \leq i \leq k_S \,\}|$$

because L_T and L°_T are different only in that L_T has an extra element $\$ < c$, and the position $LF_T^{-1}(k_T)$ in L_T corresponds to the position k_S in L°_T. □

Lemma 8. *For $1 \leq p < q \leq n$, $lcp^{\infty}(\langle T_p \rangle, \langle T_q \rangle) = lcp^{\infty}(\langle S_p \rangle, \langle S_q \rangle)$.*

Proof. Let $S_p = U\$V$ and $S_q = XY\$Z$, where $|U\$| = |X| = p < q = |XY\$|$. Then, $T_p = U\$cV$ and $T_q = XY\$cZ$. Since \$ does not appear in X,

$$lcp^{\infty}(\langle S_p \rangle, \langle S_q \rangle) = lcp^{\infty}(U\$, X) = lcp^{\infty}(\langle T_p \rangle, \langle T_q \rangle)\,.$$