Extended Parameterized Burrows–Wheeler Transform

Eric M. Osterkamp[a], Dominik Köppl[b]

[a] *University of Münster, Department of Computer Science, Münster, Germany*
[b] *University of Yamanashi, Faculty of Engineering, Kofu, Japan*

Abstract

The Burrows–Wheeler transform (BWT) lies at the heart of succinct and compressed full-text indexes for pattern matching queries. Notable variants are (a) the extended BWT (eBWT) capable to index multiple circular texts for pattern matching, or (b) the parameterized BWT (pBWT) for parameterized pattern matching. A natural extension is the combination of the virtues of both variants into a new data structure, whose name we coin with *extended parameterized BWT* (epBWT). We show that the epBWT supports pattern matching in context of parameterized pattern matching on multiple circular texts, within the same complexities as known solutions presented for the pBWT [Kim and Cho, IPL'21] for patterns not longer than the shortest indexed text. Additionally, we show how to compute the epBWT within the same complexities as [Iseri et al., ICALP'24], i.e., in compact space and quasi-linear time. As an application, we extend the matching statistics problem to the parameterized pattern matching setting on circular texts.

Keywords: extended Burrows–Wheeler transform, parameterized pattern matching, circular texts, matching statistics

1. Introduction

The seek for compact indexing data structures stems from the need to manage and analyze the ever-increasing amount of available data [2]. A mainstream line of research is devoted to fast yet space-economical variations of the FM-index [3], a full-text index built upon the Burrows–Wheeler transform (BWT) [4]. Given a pattern, the FM-index allows us to query for

⋆This article is an extension of a contribution [1] to the Data Compression Conference 2024.

the number of its occurrences in the indexed text, called *count* query, or the starting positions of these occurrences, called *locate* query. For the former, it suffices to encode the BWT with a data structure supporting rank and select queries such as the wavelet tree [5]. For the latter, we add suffix array samples to map found positions in the BWT into text-range. The few number of tools make the FM-index a predestined data structure for compact text indexing. However, the standard variant only supports exact pattern matching in the classic sense. On the one hand, a common variation in bioinformatics and computational geometry is to search in circular texts [6]. For instance, the genetic data like of the herpes simplex virus (HSV-1) [7] or circular RNA [8, 9] can be modeled as circular strings. With the aim for an index built on circular texts for pattern matching, Mantaci et al. [10] proposed the extended BWT (eBWT), which can additionally index multiple circular strings. On the other hand, structural analysis of RNA data involves a different kind of pattern matching, coined as *parameterized pattern matching* by Baker [11], and introduced to the context of RNA data by Shibuya [12]. BWT-based indexes such as the *parameterized BWTs* (pBWTs) by Ganguly et al. [13] and Kim and Cho [14] emerge as compact indexes for parameterized pattern matching. Recently, Iseri et al. [15] showed how to build the latter pBWT variant.

In the light that bioinformatic problems ask for RNA indexes that can cope with circular as well as parameterized pattern matching, a natural question to ask is whether an extension of the FM-index exists that allows a combination of both types of queries. A straight-forward answer to this question is to use the pBWT to index the concatenation of the text with itself and apply some post-processing after a query. However, this has certain drawbacks: we need twice the amount of space for the second duplicate of the text to index and cannot find conjugates of texts whose iterated concatenation contain the pattern as a prefix. For instance, we consider the pattern $P = \texttt{abcabcab}$ to cyclic match at the beginning of the text $T = \texttt{abc}$ by regarding the text as its infinite concatenation $\texttt{abcabcabc}\cdots$. Such an extension of pattern matching is of interest when querying for repetitive patterns, so that we can support texts that store only short snippets of the repetitive pattern we want to find.

Without the need for text duplication and post-processing, we present the *extended parameterized BWT* (epBWT), which is based on the ideas of Mantaci et al. [10] for the eBWT and the tricks used by Iseri et al. [15] for the pBWT. The epBWT achieves the same complexities as one of the

pBWT versions of Ganguly et al. [13] or of Kim and Cho [14]. As an application, we generalize the *matching statistics* problem proposed by Chang and Lawler [16] to a parameterized version on multiple circular texts, and show that the epBWT index can solve this problem.

1.1. Related Work

We briefly review work on circular texts and parameterized string matching. To avoid confusion, we remind the reader aware of the circular pattern matching problem that this problem asks for reporting all conjugates of the pattern, while we here study the problem to find a pattern in all conjugates of the circular texts subject to index.

Circular Texts. A folklore solution for a pattern matching index on a circular text is to index twice the linearized input by a standard indexing technique such as the suffix tree [17], and use a filter after a query to avoid reporting pattern occurrences completely within the second copy of the input. Let n be the total length of the d circular texts we want to index. A solution that does not need this duplication trick is the eBWT [10]. The eBWT takes as an input a multiset of primitive, cyclic strings and is defined as the last characters of all the rotations of all the cyclic strings in the multiset sorted in a lexicographic-like total order $\leq_\omega$ [10, Definition 4], which does not coincide with the lexicographic order in general. For example, the eBWT of the multiset $\{\mathtt{ab}, \mathtt{ab}, \mathtt{aba}\}$ is $\mathtt{babbaaa}$ since $\mathtt{aab} \leq_\omega \mathtt{aba} \leq_\omega \mathtt{ab} \leq_\omega \mathtt{ab} \leq_\omega \mathtt{baa} \leq_\omega \mathtt{ba} \leq_\omega \mathtt{ba}$.[1] The inverse of the eBWT, known as the *Gessel–Reutenauer transform* [18], has already been conceived more than ten years prior to the discovery of the eBWT. Given a string T, the Gessel–Reutenauer transform maps T to the multiset of strings whose eBWT is T. However, Mantaci et al. [10] did not cover algorithmic details like the construction — yet it is clear that the construction time is upper bounded by the total length of all conjugates [10, after Example 9], which is $O(n^2)$. Later, Hon et al. [19] proposed a circular pattern matching algorithm using eBWT. They use $n \log \sigma(1 + o(1)) + O(n) + O(d \log n)$ bits and can answer a count query in $O(m \lg \sigma)$ for a pattern of length m. Subsequently, with a larger number of authors [20], they gave a construction algorithm for the

[1] The relation $\leq_\omega$ is not to be confused with the here later defined preorder $\preceq_\omega$ since the former assigns a linear order even on periodic strings by measuring the lengths of their roots for breaking ties.

eBWT in $O(n \lg n)$ time. Their idea is to construct a so-called *circular suffix array* CSA such that the i-th position of the eBWT is given by $T[\mathsf{CSA}[i]-1]$, where T is the concatenation of all input texts. For their algorithm to work, they make the assumption that all input strings are non-periodic, belong to different conjugacy classes, and have asymptotically the same lengths. Very recently, Cotumaccio [21] showed how to get rid of these assumptions while retaining the same time and space complexities. Bonomo et al. [22] presented an online algorithm building the eBWT in $O(n \lg n / \lg \lg n)$ time; here, online means that a construction algorithm updates the eBWT on reading a new input string. An interesting aspect regarding the online construction is that the up so far only known online technique [23, 24] for computing the traditional BWT needs the text to be given in reversed order (starting with the last character). Recently, Bannai et al. [25] showed that the eBWT can be constructed in linear time. Subsequently, Boucher et al. [26] practically simplified this construction. One of the currently fastest eBWT construction algorithms in practice is due to Olbrich et al. [27].

Parameterized Matching. The parameterized suffix tree as an extension of suffix trees to the parameterized indexing problem by Baker [11, 28] pioneered a new research area that adapts classic indexes to parameterized pattern matching. We briefly mention some achievements in this field and refer to Mendivelso et al. [29] for a more comprehensive overview of this research area. In timeline, we are aware of parameterized versions of the suffix array [30, 31], the longest previous factor array [32], the position heap [33], the LCP array [34], the linear-sized suffix trie [35], the suffix tray [36], compact suffix trees [37], and DAWGs [38]. Finally, Ganguly et al. [13] proposed the pBWT as the first parameterized index that is compact, i.e., the number of required bits is linear in the length of the input text. Kim and Cho [14] gave a simplification of this technique, requiring however twice the amount of space. The construction of Kim and Cho's variant has been addressed by Hashimoto et al. [39] whose time, like other construction algorithm of preceding data structures, is linearly dependent on both the text length and the alphabet size of the parameterized symbols. Finally, a breakthrough in the time complexity was achieved by Iseri et al. [15], who improved the latter linear dependency to a logarithmic one, and thus gave for the very first time a construction algorithm for a parameterized index with quasilinear time for arbitrarily large alphabets.

1.2. Structure of this Article

We start with Section 2 to cover basic notations and tools. We introduce the pBWT and formalize the type of index that is subject to this article. Subsequently, we present the epBWT index in Section 3 and give a construction algorithm in Section 4. As an application, we solve a generalization of matching statistics to parameterized matching on circular texts in Section 5 and finally give a conclusion in Section 6 with open problems and future work. Our main result can be stated as follows.

Theorem 1.1. *Let $T_1, ..., T_d$ non-empty, parameterized strings over an alphabet Σ of size σ and $n = |T_1 \cdots T_d|$. The epBWT index of $\{T_1, ..., T_d\}$ is a data structure that takes $O(n \lg \sigma)$ bits of space and can be computed in $O(n \lg \sigma)$ bits of space and $O(nt_{\text{query}})$ time, where $t_{\text{query}} = \frac{\lg \sigma \lg n}{\lg \lg n}$. Given a pattern P of length m, it counts all conjugates of $T_1, ..., T_d$ whose infinite concatenation has a prefix that p-matches with P, in $O(mt_{\text{query}})$ time. It can restore the conjugacy classes of $T_1, ..., T_d$ up to p-match equivalence, in $O(nt_{\text{query}})$ time. It can add and remove a text in $O(n't_{\text{query}})$ time, where n' is the maximum length of all texts (including the one to insert). By switching to static data structures, we can improve t_{query} to $\lg \sigma$ and the space to $2n \lg \sigma + 2n + O(n\frac{\lg \lg n}{\lg n})$ bits of space.*

Compared to the conference version [1], we give details on the construction and matching statistics as an application. The presentation of the epBWT index has been simplified and didactically improved with examples and explanations. In particular, we could simplify the description of ω-preorder of the conjugates in the epBWT by introducing a new encoding $\langle .. \rangle_{\text{r}}$ (cf. Lemma 3.1) such that we no longer need to rely on the $[\![..]\!]$-encoding like in the conference paper.

2. Preliminaries

Let $\lg = \log_2$ and $\mathbb{N}_{\geq 1}$ denote, respectively, the logarithm to base two and the set of natural numbers starting at 1. An interval $\{i, i+1, .., j-1, j\}$ of integers is denoted by $[i..j]$, where $[i..j] = \emptyset$ if $j < i$. Appendix A gives an overview of the variables used in this article. We start with basic notations for strings, queries we want to answer on strings, data structures that can answer those queries, and then move to the parameterized matching. In the end of this section, we formalize the problem we tackle in this article.

2.1. Strings

Let Σ denote an *alphabet*, i.e., a non-empty, finite and totally ordered set. An element of Σ is called a *symbol*, a sequence of symbols from Σ a *string over* Σ and a subsequence U of a consecutive range of elements from a string V over Σ a *substring* of V. A substring U of V is *proper* if $U \neq V$. Let Σ^* denote the set of all finite sequences of symbols from Σ, Σ^ω the set of all countably infinite sequences of symbols from Σ, and $\Sigma^\infty = \Sigma^* \cup \Sigma^\omega$. Then an element of Σ^* is called a *finite* string over Σ and an element of Σ^ω an *infinite* string over Σ. Whether we are talking about a finite or an infinite string should be clear from the context at all times. The concatenation of any $U \in \Sigma^*$ and $V \in \Sigma^\infty$ is denoted by $U \cdot V$ or UV. We write U^k if we concatenate $k \in \mathbb{N}$ instances of $U \in \Sigma*$, and we denote by U^ω the infinite string obtained by infinitely iterating U, i.e., $U^\omega = UUU \cdots \in \Sigma^\omega$. We call a string $V \in \Sigma^*$ *primitive* if $V = U^k$ for $U \in \Sigma^*$ and $k \in \mathbb{N}$ already implies $V = U$ and $k = 1$. It is known that for every $V \in \Sigma^*$ there exists a unique primitive $U \in \Sigma^*$ and a unique $k \in \mathbb{N}$ such that $V = U^k$, denoted by $\mathrm{root}(V)$ and $\exp(V)$, respectively.

If $U = VW$ for $V \in \Sigma^*$ and $W \in \Sigma^\infty$, then the substrings V and W are called *prefix* and *suffix* of U, respectively. Assume $W \in \Sigma^*$. Then WV is called a *conjugate* of $U = VW$. For $U \in \Sigma^\infty$, let $|U|$ denote the *length* of U, i.e., $|U| = \infty$ if $U \in \Sigma^\omega$, or the number of symbols in U otherwise. Let ε denote the unique string of length 0 and denote the length of the longest common prefix of $U, V \in \Sigma^\infty$ by $\mathrm{lcp}(U, V)$. Also, let $\Sigma^+ = \Sigma^* - \{\varepsilon\}$.

Fix some $V \in \Sigma^\infty$ and $i, j \in [1..|V|]$. Then $V[i]$ denotes the i-th symbol of V. A substring $V[i] \cdots V[j]$ starting at position i and ending at position j of V is denoted by $V[i..j]$, where we set $V[i..j] = \varepsilon$ if $j < i$. Let $V[..i]$ denote the prefix $V[1..i]$ of V and $V[i..]$ the suffix W of V such that $V = V[..i-1]W$. For notational convenience, if $k \leq 0$, then $V[..k] = \varepsilon$, and if $k \geq |V|+1$, then $V[k..] = \varepsilon$.

We assume $V \in \Sigma^+$ for the remainder of the section. Let $\mathrm{Rot}(V, 0) = V$ and $\mathrm{Rot}(V, k+1) = \mathrm{Rot}(V, k)[2..]\mathrm{Rot}(V, k)[1]$ for $k \in \mathbb{N}$, i.e., $\mathrm{Rot}(V, k)$ denotes the k-th *left rotation* of V. Note that $\mathrm{Rot}(V, k) = \mathrm{Rot}(V, k \mod |V|)$ for every $k \in \mathbb{N}$. We call $k \in [1..|V|]$ satisfying $V[i] = V[i+k]$ for every $i \in [1..|V|-k]$ a *period* of V.

Let $<$ denote the lexicographical order on Σ. For $U, W \in \Sigma^*$, we write $U < W$ if and only if U is a proper prefix of W or $U[\mathrm{lcp}(U, W) + 1] < W[\mathrm{lcp}(U, W)+1]$. For $U, W \in \Sigma^\omega$, we write $U < W$ if and only if there exists $k \in \mathbb{N}$ such that $U[..k-1] = W[..k-1]$ and $U[k] < W[k]$.

2.2. Queries on Strings

Let $f \leq g \in \Sigma$ and $i, j \in [1..|V|]$. The query $\mathrm{rank}_g(V, i)$ returns the number of occurrences of g in $V[..i]$, the query $\mathrm{select}_g(V, i)$ returns the index of the i-th occurrence of g in V if it exists, and 0 otherwise and the query $\mathrm{rnkcnt}_V(i, j, f, g)$ returns the number of entries in $V[i..j]$ having a value in $[f..g]$. Moreover, the operation $\mathrm{insert}_V(i, g)$ inserts the symbol g at position i of V and the operation $\mathrm{delete}_V(i)$ deletes the i-th entry of V, e.g., $\mathrm{insert}_{\mathtt{312}}(2, 4)$ transforms `312` into `3412` and $\mathrm{delete}_{\mathtt{3412}}(3)$ transforms `3412` into `342`. For $i \leq j$, the query $\mathrm{RmQ}_V(i, j)$ returns the index of a minimal value in $V[i..j]$, the query $\mathrm{RMQ}_V(i, j)$ returns the index of a maximal value in $V[i..j]$, and the query $\mathrm{RNV}_V(i, j, g)$ returns the smallest value in $V[i..j]$ larger than g if it exists, and g otherwise.

We borrow notation from Iseri et al. [15, Section 2.1], who introduced *find previous queries* (FPQ) and *find next queries* (FNQ) as syntactic elements for range searches in a string. Both queries are generalizations of predecessor and successor queries by additionally requiring the output to satisfy a predicate.[2] In detail, they are defined as follows. Let $i \in [0..|V|]$. The query $\mathrm{FNQ}_g(V, i)$ returns the smallest index $k \in [\max\{i, 1\}..|V|]$ such that $V[k] = g$ if it exists and $|V| + 1$ otherwise. Similarly, the query $\mathrm{FNQ}_{\geq g}(V, i)$ returns the smallest $k \in [\max\{i, 1\}..|V|]$ such that $V[k] \geq g$ if it exists, and $|V|+1$ otherwise. The query $\mathrm{FPQ}_g(V, i)$ returns the largest $k \in [1..i]$ such that $V[k] = g$ if it exists and 0 otherwise, and the query $\mathrm{FPQ}_{\geq g}(V, i)$ returns the largest $k \in [1..i]$ such that $V[k] \geq g$ if it exists, and 0 otherwise. Last, the query $\mathrm{MI}_V(i, g)$ returns the maximal interval $[\ell..r]$ such that $0 \leq \ell \leq i \leq r \leq |V|$, $V[k] \geq g$ for every $k \in [\ell + 1..r]$. Note that FPQ_g and FNQ_g queries can be answered by a constant number of rank and select queries, and an MI query can be answered by a constant number of $\mathrm{FPQ}_{\geq g}$ and $\mathrm{FNQ}_{\geq g}$ queries. See Figure 1 for examples.

Depending on the queries and operations required, we may represent a string by one of the following data structures. We assume a random access model with word size $\Omega(\lg n)$, where n denotes the length of an input string.

Lemma 2.1 ([41, 42])**.** *A static string of length n over $[0..\sigma]$ with $\sigma \leq n^{O(1)}$ can be stored in a data structure occupying $n \lg \sigma + O(n \frac{\lg \lg n}{\lg n})$ bits of space,*

[2] A cousin of FPQs and FNQs are *next smaller value queries* [40], which can express, together with successor queries, FPQ and FNQs.

i	1	2	3	4	5	6	7	8
$V[i]$	7	14	5	1	11	27	11	7

Figure 1: Example integer string to illustrate queries. Here, $\mathrm{rank}_{11}(V,5) = 1$, $\mathrm{select}_{11}(V,2) = 7$, $\mathrm{select}_{23}(V,1) = 0$, $\mathrm{rnkcnt}_V(2,7,7,14) = 4$, $\mathrm{RmQ}_V(2,5) = 4$, $\mathrm{RMQ}_V(2,5) = 2$, $\mathrm{RNV}_V(1,5,8) = 11$, $\mathrm{RNV}_V(3,5,13) = 13$, $\mathrm{FNQ}_1(V,2) = 4$, $\mathrm{FNQ}_{\geq 23}(V,7) = 9$, $\mathrm{FPQ}_1(V,2) = 0$, $\mathrm{FPQ}_{\geq 1}(V,2) = 2$, $\mathrm{MI}_V(5,6) = [4..8]$, and $\mathrm{MI}_V(2,15) = [2..2]$.

supporting the queries access, rank, select *and* rnkcnt *in* $O(\lg \sigma)$ *time. It can be constructed in* $O(n \lg \sigma)$ *time with additional* $n \lg \sigma + O(1)$ *bits of space.*

Lemma 2.2 ([43])**.** *A static string of length* n *over an alphabet of size* $\sigma \in O(n)$ *can be stored in a data structure occupying* $2n + O(n\frac{\lg \lg n}{\lg n})$ *bits of space, supporting either the query* RmQ *or the query* RMQ *in* $O(1)$ *time. It can be constructed in* $O(n)$ *time with additional* $O(n)$ *bits of space.*

Lemma 2.3 ([44])**.** *A dynamic string of length* n *over* $[0..\sigma]$ *with* $\sigma \leq n^{O(1)}$ *can be stored in a data structure occupying* $O(n \lg \sigma)$ *bits of space, supporting insertion, deletion and the queries* access, rank *and* select *in* $O(\frac{\lg n}{\lg \lg n})$ *time.*

Lemma 2.4 ([15, Lemma 4])**.** *A dynamic string of length* n *over* $[0..\sigma]$ *with* $\sigma \leq n^{O(1)}$ *can be stored in a data structure occupying* $O(n \lg \sigma)$ *bits of space, supporting insertion, deletion and the queries* access, rank, select, RNV, $\mathrm{FPQ}_{\geq}$, $\mathrm{FNQ}_{\geq}$ *and* rnkcnt *in* $O(\frac{\lg \sigma \lg n}{\lg \lg n})$ *time.*

2.3. Parameterized Matching

Let Σ_s and Σ_p denote two disjoint alphabets, $\sigma_\mathrm{s} = |\Sigma_\mathrm{s}|$ and $\sigma_\mathrm{p} = |\Sigma_\mathrm{p}|$. A symbol from Σ_s is called a *static symbol (s-symbol)* and a symbol from Σ_p is called a *parameterized symbol (p-symbol)*. We set $\Sigma = \Sigma_\mathrm{s} \cup \Sigma_\mathrm{p}$, $\sigma = |\Sigma|$, and call a string over Σ a *parameterized string (p-string)*. Finally, we introduce two special symbols that do not convey any information of the input we want to index, but are needed for technical reasons. The first is \$ that denotes the smallest s-symbol. The second is the symbol $\aleph \notin \Sigma_\mathrm{s} \cup \Sigma_\mathrm{p}$, which we stipulate to be larger than any integer.

We assume $\sigma \leq n^{O(1)}$, i.e., each symbol from Σ can be represented by $O(\lg n)$ bits of space. Since we can define order preserving mappings from Σ_s and Σ_p to $[-\sigma_\mathrm{s}.. -1]$ and $[1..\sigma_\mathrm{p}]$, respectively, we can assume that $\mathbb{N} \cap \Sigma_\mathrm{s} = \emptyset$, that any s-symbol is distinguishable from both p-symbols and positive

integers in $O(1)$ time and that the order of two s-symbols can be determined in $O(1)$ time. For any examples throughout, we will use $\Sigma_{\mathrm{s}} = \{\$, \mathtt{a}, \mathtt{b}\}$ and $\Sigma_{\mathrm{p}} = \{\mathtt{A}, \mathtt{B}, \mathtt{C}\}$, where $\$ < \mathtt{a} < \mathtt{b}$.

We say that two p-strings $U, V \in \Sigma^*$ of equal length *parameterized match (p-match)* if and only if there exists a bijection $\psi : \Sigma_{\mathrm{p}} \to \Sigma_{\mathrm{p}}$ such that $U[i] = V[i]$ if $V[i] \in \Sigma_{\mathrm{s}}$, or $U[i] = \psi(V[i])$ otherwise, for each $i \in [1..|U|]$, and write $U \approx V$. For example, $U = \mathtt{BCAaCB} \approx \mathtt{CABaAC} = V$, where $\psi(\mathtt{A}) = \mathtt{C}$, $\psi(\mathtt{B}) = \mathtt{A}$ and $\psi(\mathtt{C}) = \mathtt{B}$.

The goal of this subsection is to study *parameterized matching*, which we define as follows.

Problem 1 (P-Count)**.** *Given a text p-string T and a pattern p-string P, count the number of all occurrences of P in T.*

By counting occurrences we mean to count all substrings of T that p-match with P, or more formally, to compute the cardinality of the subset $\mathcal{I} \subset [1..n]$ with $i \in \mathcal{I}$ if and only if $P \approx T[i..i + |P| - 1]$. As an example, we consider the text p-string $T = \mathtt{CABaAC}$ and the pattern p-strings $P_1 = \mathtt{bA}$ and $P_2 = \mathtt{CB}$. It is easy to see that P_1 has no occurrences in T since the s-symbol $\mathtt{b}$ does not occur in T. The pattern p-string P_2, on the other hand, has exactly 3 occurrences since $P_2 \approx T[1..2] \approx T[2..3] \approx T[5..6]$, $P_2 \not\approx T[3..4]$ and $P_2 \not\approx T[4..5]$.

While Baker [11] gives a linear time algorithm to check if two p-strings p-match, they note that it does not generalize conveniently to solving p-Count. Instead, they present an encoding that reduces checking whether two strings p-match to the exact matching of their encodings. In what follows, we introduce two such convenient encodings, the first of which is a variant of Baker's.

Kim and Cho's [14] *prev-encoding* $\langle V \rangle$ of $V \in \Sigma^*$ is a string of length $|V|$ over $\Sigma_{\mathrm{s}} \cup [1..|V| - 1] \cup \{\aleph\}$ such that

$$\langle V \rangle[i] = \begin{cases} V[i] & \text{if } V[i] \in \Sigma_{\mathrm{s}}, \\ \aleph & \text{if } V[i] \in \Sigma_{\mathrm{p}} \wedge \forall (1 \le j < i) : V[i] \neq V[j], \text{ and} \\ \mathrm{FPQ}_{V[i]}(V, i-1) & \text{otherwise,} \end{cases}$$

for every $i \in [1..|V|]$, i.e., the leftmost occurrence of a p-symbol in V is replaced by $\aleph$ and each successive occurrence with the distance to its previous occurrence. Note that $\langle V \rangle[..i] = \langle V[..i] \rangle$ for each $V \in \Sigma^*$ and $i \in [1..|V|]$.

i	$\langle V[i..]\rangle$	$\langle \mathrm{Rot}(V,i)\rangle$	$\langle V[..i]\rangle$	$[\![V[i..]]\!]$	$[\![\mathrm{Rot}(V,i)]\!]$	$[\![V[..i]]\!]$
1	𝔵𝔵𝔵a35	𝔵𝔵a3𝔵1	𝔵	323a21	23a213	1
2	𝔵𝔵a3𝔵	𝔵a𝔵𝔵13	𝔵𝔵	23a23	3a2132	22
3	𝔵a𝔵𝔵	a𝔵𝔵13𝔵	𝔵𝔵𝔵	3a33	a21323	333
4	a𝔵𝔵	𝔵𝔵13𝔵a	𝔵𝔵𝔵a	a22	21323a	333a
5	𝔵𝔵	𝔵1𝔵𝔵a3	𝔵𝔵𝔵a3	22	1323a2	323a2
6	𝔵	𝔵𝔵𝔵a35	𝔵𝔵𝔵a35	1	323a21	323a21

Figure 2: Encodings of substrings and conjugates of $V = \mathtt{CABaAC}$.

For any $U \in \Sigma^*$, let $|U|_{\mathrm{p}} = \mathrm{rank}_{\mathfrak{x}}(\langle U\rangle, |U|)$ denote the number of distinct p-symbols in U. The encoding $[\![V]\!]$ of $V \in \Sigma^*$ proposed by Hashimoto et al. [39] is given by a string of length $|V|$ over $\Sigma_{\mathrm{s}} \cup [1..\sigma_{\mathrm{p}}]$ such that $[\![V]\!][i] = V[i]$ if $V[i] \in \Sigma_{\mathrm{s}}$, or $[\![V]\!][i] = \left|\mathrm{Rot}(V,i)[..\mathrm{select}_{V[i]}(\mathrm{Rot}(V,i),1)]\right|_{\mathrm{p}}$ otherwise, for each $i \in [1..|V|]$. In other words, we view V circularly and replace each occurrence of a p-symbol in V by the number of distinct p-symbols until its next occurrence. See Figure 2 for some examples.

Lemma 2.5 ([11, 39, Propositions 1])**.** *Let $U, V \in \Sigma^*$. Then $\langle U\rangle = \langle V\rangle \Leftrightarrow U \approx V \Leftrightarrow [\![U]\!] = [\![V]\!]$.*

By Lemma 2.5, a naive solution for p-Count is to check either $\langle P\rangle = \langle T[i..i+|P|-1]\rangle$ or $[\![P]\!] = [\![T[i..i+|P|-1]]\!]$ for every $i \in [1..n-|P|+1]$.

In what follows, we count the number of $\mathfrak{x}$'s in the longest common prefix of two p-strings U and V, which helps us to judge, given both are prefixed by symbols, the order between the prev-encoding of the prefixed versions, if the order of the prev-encoding of U and V is already known. We later use that fact for adapting the backward search, the centerpiece of the FM-index, to parameterized matching.

For $V \in \Sigma^+$, let $\pi(V) = [\![V]\!][1]$. For $U, V \in \Sigma^*$, let $\mathrm{lcp}^\infty(U,V) = \mathrm{rank}_{\mathfrak{x}}(\langle U\rangle, \mathrm{lcp}(\langle U\rangle, \langle V\rangle))$. Since $\mathfrak{x}$ does not appear in $\langle V^i\rangle[|V|+1..]$ for any $V \in \Sigma^*$ and integer $i \geq 1$, $\mathrm{lcp}^\infty(U,V) = \mathrm{lcp}^\infty(U^j, V^k)$ for each $U, V \in \Sigma^*$ and integers $j, k \geq 1$. The following results are obtained by examination of the definitions.

Lemma 2.6 ([15, Corollary 7])**.** *Let $U, V \in \Sigma^+$, $e = \mathrm{lcp}^\infty(U[2..], V[2..])$ and $\langle U[2..]\rangle < \langle V[2..]\rangle$. Then $\langle V\rangle < \langle U\rangle$ if and only if $\pi(V) < \pi(U) \wedge (\pi(V) \in \Sigma_{\mathrm{s}} \vee (\pi(U), \pi(V) \notin \Sigma_{\mathrm{s}} \wedge \pi(V) \leq e))$.*

Lemma 2.7 ([15, Table 1])**.** *Let* $U, V \in \Sigma^+$ *and* $e = \mathrm{lcp}^\infty(U[2..], V[2..])$. *Then*

$$\mathrm{lcp}^\infty(U,V) = \begin{cases} 0 & \text{if } \pi(U) \neq \pi(V) \wedge (\pi(U) \in \Sigma_\mathrm{s} \vee \pi(V) \in \Sigma_\mathrm{s}), \\ e & \text{if } \pi(U) = \pi(V) \leq e, \\ \pi(U) & \text{if } \pi(U) \notin \Sigma_\mathrm{s} \wedge \pi(U) \leq e \wedge \pi(U) < \pi(V), \\ \pi(V) & \text{if } \pi(V) \notin \Sigma_\mathrm{s} \wedge \pi(V) \leq e \wedge \pi(V) < \pi(U), \\ e+1 & \text{otherwise, i.e., if } e < \min\{\pi(U), \pi(V)\}. \end{cases}$$

2.4. The Parameterized Burrows–Wheeler Transform

Let $T \in \Sigma^+$ denote a string of length n such that $\mathrm{rank}_\$(T, n) = 1$ and $T[n] = \$$, i.e., \$ is the unique delimiter of the string T. In what follows, we present a variant of Ganguly et al.'s [13] original, space-efficient index solving p-Count for T.

The (parameterized) *suffix array* [30] SA_T is a string of length n over $[1..n]$ such that $\mathrm{SA}_T[i] = j$ if and only if there are $i-1$ suffixes of T whose prev-encodings are smaller than the prev-encoding of the suffix $T[j..]$, i.e.,

$$\begin{aligned} i - 1 &= |\{k \in [1..n] \mid \langle \mathrm{Rot}(T,k) \rangle < \langle \mathrm{Rot}(T, j-1) \rangle\}| \\ &= |\{k \in [1..n] \mid \langle T[k..] \rangle < \langle T[j..] \rangle\}|. \end{aligned}$$

Then the *parameterized Burrows–Wheeler Transform (pBWT)* L_T of T is a string of length n over $\Sigma_\mathrm{s} \cup [1..\sigma_\mathrm{p}]$ such that $\mathrm{L}_T[i] = [\![\mathrm{Rot}(T, \mathrm{SA}_T[i] - 1)]\!][n]$ for each $i \in [1..n]$. The pBWT is further augmented to act as an index. We let F_T denote a string of length n over $\Sigma_\mathrm{s} \cup [1..\sigma_\mathrm{p}]$ such that $\mathrm{F}_T[i] = [\![\mathrm{Rot}(T, \mathrm{SA}_T[i] - 1)]\!][1]$ for each $i \in [1..n]$, and we let LCP^∞_T denote a string of length n over $[0..\sigma_\mathrm{p}]$ such that $\mathrm{LCP}^\infty_T[1] = 0$ and $\mathrm{LCP}^\infty_T[i] = \mathrm{lcp}^\infty(\mathrm{Rot}(T, \mathrm{SA}_T[i] - 1), \mathrm{Rot}(T, \mathrm{SA}_T[i-1] - 1))$ for each $i \in [2..n]$.

The statement regarding construction of the following result is originally for an index of a suffix-based variant of our pBWT, which differs in the definition of F_T and L_T. The transformation into F_T and L_T as defined here is done after construction by computing $[\![T]\!]$ and utilization of the so-called LF-mapping.

Lemma 2.8 ([14, 15])**.** *Let* $T \in \Sigma^+$, $n = |T|$, $T[n] = \$$ *and* $\mathrm{rank}_\$(T, n) = 1$. *Then* F_T, L_T *and* LCP^∞_T *form an index that occupies* $O(n \lg \sigma)$ *bits of space and solves* p-Count *in* $O(m \lg \sigma)$ *time, where* m *is the length of the pattern. The index can be computed in* $O(n \lg \sigma)$ *bits of space and* $O(n \frac{\lg \sigma \lg n}{\lg \lg n})$ *time, and the original text* T *restored up to p-match equivalence in* $O(n \lg \sigma)$ *time.*

i	$\mathrm{SA}_T[i]$	$\mathrm{F}_T[i]$	$\mathrm{L}_T[i]$	$V_T[i]$	i	$\mathrm{SA}_T[i]$	$\mathrm{F}_T[i]$	$\mathrm{L}_T[i]$	$V_T[i]$
1	25	\$	1	0	14	8	1	2	1
2	2	a	1	0	15	10	1	2	2
3	5	a	2	1	16	18	1	b	2
4	3	b	a	0	17	12	1	2	3
5	22	b	2	1	18	20	3	3	1
6	17	b	3	1	19	15	3	2	3
7	6	b	a	1	20	7	2	b	2
8	24	1	1	0	21	9	2	1	2
9	1	1	\$	1	22	11	2	1	2
10	4	2	b	1	23	14	2	3	2
11	21	2	3	1	24	19	3	1	2
12	16	3	3	2	25	13	3	1	3
13	23	1	b	1					

Figure 3: The pBWT index of $T = \texttt{BabBabABBAABBACAbBBCAbBB\$}$. Here, $V_T = \mathrm{LCP}^{\infty}_T$.

We call the index of Lemma 2.8 the *pBWT index of* T. See Figure 3 for an example. The pBWT index of a text $T \in \Sigma^+$ can compute the *suffix range* $\mathrm{SR}_T(P)$ of a pattern $P \in \Sigma^*$, which is the unique maximal interval $[\ell..r] \subseteq [1..n]$ such that $P \approx T[\mathrm{SA}_T[i]..\mathrm{SA}_T[i]+|P|-1]$ if and only if $i \in [\ell..r]$. The length of this interval is consequently the number of occurrences of P in T, i.e., the solution to p-Count.

The computation of the suffix range is also called the *backward search*, which works similarly to its namesake in the BWT. At its core is the so-called *LF-mapping* LF_T, which maps $i \in [1..n]$ to $\mathrm{SA}_T^{-1}[\mathrm{SA}_T[i]-1]$ if $\mathrm{SA}_T[i] > 1$, and otherwise to $\mathrm{SA}_T^{-1}[n] = 1$, where SA_T^{-1} denotes the inverse of the suffix array. It can be computed efficiently by finitely many queries on F_T and L_T, which is due to Lemma 2.6 and basing the definition of F_T and L_T on the $[\![..]\!]$-encoding.

Corollary 2.9 ([14])**.** *Let $T \in \Sigma$ and $i \in [1..|T|]$. Then*

$$\mathrm{LF}_T(i) = \mathrm{select}_{\mathrm{L}_T[i]}(\mathrm{F}_T, \mathrm{rank}_{\mathrm{L}_T[i]}(\mathrm{L}_T, i)).$$

Given a pattern p-string $P \in \Sigma^*$, a symbol $g \in \Sigma$ and the suffix range $\mathrm{SR}_T(P)$, we can use the LF-mapping to compute $\mathrm{SR}_T(gP)$.

Corollary 2.10 ([14])**.** *Let $T, P \in \Sigma^*$ and $g \in \Sigma$. If g is an s-symbol or a p-symbol that already occurred in P, then $\mathrm{LF}_T(i) \in \mathrm{SR}_T(gP)$ if and only*

if $i \in \mathrm{SR}_T(P)$ *and* $\mathrm{L}_T[i] = \pi(gP)$*, and otherwise* $\mathrm{LF}_T(i) \in \mathrm{SR}_T(gP)$ *if and only if* $i \in \mathrm{SR}_T(P)$ *and* $\mathrm{L}_T[i] \geq \pi(gP)$*.*

We give an exemplary, naive backward search for $P = \texttt{bCC}$ with the pBWT index of $T = \texttt{BabBabABBAABBACAbBBCAbBB\$}$, which is presented in Figure 3. Initially, $\mathrm{SR}_T(P[4..]) = \mathrm{SR}_T(\varepsilon) = [1..25]$. Since the p-symbol $P[3] = \texttt{C}$ did not yet occur in $P[4..] = \varepsilon$, we have to compute the LF-mapping for each $i \in [1..25]$ satisfying $\mathrm{L}_T[i] \geq \pi(\texttt{C}) = 1$ to obtain $\mathrm{SR}_T(\texttt{C}) = [8..25]$ by Corollary 2.10. Next, we find that the p-symbol $P[2] = \texttt{C}$ already occurred in $P[3..] = \texttt{C}$. Thus, we have to compute the LF-mapping of each $i \in [8..25]$ satisfying $\mathrm{L}_T[i] = \pi(\texttt{CC}) = 1$ to obtain $\mathrm{SR}_T(\texttt{CC}) = [13..17]$. Last, we find that $P[1] = \texttt{b}$ is an s-symbol. Hence, we have to compute the LF-mapping of each $i \in [13..17]$ such that $\mathrm{L}_T[i] = \pi(\texttt{bCC}) = \texttt{b}$, which yields $\mathrm{SR}_T(P) = [5..6]$. This concludes the backward search. We find that $\texttt{bCC}$ occurs twice, p-matching $T[22..24]$ and $T[17..19]$.

2.5. Circular, Parameterized Matching

Throughout, let $\mathcal{T} = \{T_1, ..., T_d\} \subset \Sigma^+$ with $d \in \mathbb{N}_{\geq 1}$ and $n_k = |T_k|$ for each $k \in [1..d]$. For our running example, let $d = 3$, $T_1 = \texttt{Bab}$, $T_2 = \texttt{ABBA}$ and $T_3 = \texttt{CAbBB}$. We are interested in solving the following problem on $\mathcal{T}$.

Problem 2 (EP-COUNT)**.** *Given a set of p-strings* $\mathcal{T}$ *and a pattern p-string* P*, count each conjugate of the p-strings in* $\mathcal{T}$ *whose infinite iteration has a prefix p-matching* P*. More formally, return*

$$\sum_{x=1}^{d} |\{i \in [1..n_x] \mid \mathrm{Rot}(T_x, i)^\omega[1..|P|] \approx P\}| \,.$$

We call the query described in Problem 2 a *count* query, and call a count query for a pattern no longer than the shortest text a *shortcount* query. As an example, consider the count query for $P = \texttt{CCB}$ in our running example $\mathcal{T}$, which is a shortcount query. We find that only the conjugates $\mathrm{Rot}(T_2, 1)^\omega[..3] = \texttt{BBA}$, $\mathrm{Rot}(T_2, 3)^\omega[..3] = \texttt{AAB}$ and $\mathrm{Rot}(T_3, 3)^\omega[..3] = \texttt{BBC}$ of text p-strings in $\mathcal{T}$ p-match $\texttt{CCB}$, i.e., the count query returns the value 3.

In the remainder of this section, we present a straight-forward solution that supports shortcount queries based on Lemma 2.8. We assume that the special character \$ does not appear in the input, i.e., $\mathrm{rank}_{\$}(T_k, n_k) = 0$ for each $k \in [1..d]$. In the preprocessing, we construct the pBWT index

	T_1	T_1	T_2	T_2	T_3	T_3	
$T[i] =$	B a b B a b A B B	A	A	B	B	A	C	A	b	B	B	C	A	b	B	B	$
$\mathrm{A}_T[i] =$	1 1 1 0 0 0 1 1 1	1	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0
$\mathrm{B}_T[i] =$	0 1 0 1 0 1 0 0 1	0	0	1	0	1	1	1	0	0	1	1	1	0	0	1	0
$i =$	1 2 3 4 5 6 7 8 9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

Figure 4: Bit strings $\mathrm{A}_T[i]$ and $\mathrm{B}_T[i]$ augmenting the pBWT index for answering EP-COUNT, instantiated with our running example whose pBWT index is given in Figure 3.

of $T = T_1^2 \cdots T_d^2\$$. If we compute the suffix range $\mathrm{SR}_T(P)$ of a pattern p-string $P \in \Sigma^*$ no longer than the shortest text, then the length of the suffix range $\mathrm{SR}_T(P)$ is at least the number of occurrences since each conjugate of $T_1, ..., T_d$ appears once as a prefix of distinct suffixes of T. However, an occurrence might be counted twice or cross text boundaries. In our running example, compare the occurrences of $P = \mathtt{a}$ in $T_1, ..., T_3$ totaling 1 with the occurrences in $T = T_1^2 \cdot T_2^2 \cdot T_3^2 \cdot \$$, which total 2.

To remedy this issue, let $\mathrm{A}_T = \mathtt{1}^{n_1}\mathtt{0}^{n_1} \cdots \mathtt{1}^{n_d}\mathtt{0}^{n_d}\mathtt{0}$ and let B_T denote a bit string of length $|T|$ such that $\mathrm{B}_T[i] = 1$ if and only if $\mathrm{A}_T[\mathrm{SA}_T[i]] = 1$. Then the entries of B_T evaluating to 1 are one-to-one to the conjugates of $T_1, ..., T_d$ and consequently $\mathrm{rank}_1(\mathrm{B}_T, r) - \mathrm{rank}_1(\mathrm{B}_T, \ell) + \mathrm{B}_T[\ell]$ is the shortcount of P, where $[\ell..r] = \mathrm{SR}_T(P)$. Thus, the pBWT index of T augmented with A_T and B_T solves EP-COUNT in the case of shortcount queries with the same time complexity as a pBWT index for P-COUNT. See Figure 3 and Figure 4 for the solution to our running example. Note that this index cannot solve count queries in general because a conjugate of a text is considered in EP-COUNT even if it is shorter than P, e.g., the count query for $P = T_1^9$ should return 1, but the solution based on Lemma 2.8 for our running example returns 0.

In what follows, we apply Mantaci and colleagues' [10] techniques to the pBWT to obtain a space-efficient index capable of count queries on multiple texts that does without any artificial symbols or an increase in input size.

3. An Extension of the pBWT

Throughout, let $T = T_1 \cdots T_d$, $n = |T|$ and $\mathrm{LEN}_\mathcal{T} = \mathtt{1}\mathtt{0}^{n_1-1} \cdots \mathtt{1}\mathtt{0}^{n_k-1}\mathtt{1}$. Then $T_k = T[\mathrm{select}_1(\mathrm{LEN}_\mathcal{T}, k)..\mathrm{select}_1(\mathrm{LEN}_\mathcal{T}, k+1) - 1]$ for each $k \in [1..d]$. Let $\mathrm{txt}_\mathcal{T}(i) = \mathrm{rank}_1(\mathrm{LEN}_\mathcal{T}, i)$, $\mathrm{pos}_\mathcal{T}(i) = i - \mathrm{select}_1(\mathrm{LEN}_\mathcal{T}, \mathrm{txt}_\mathcal{T}(i))$ and $C_\mathcal{T}(i) = \mathrm{Rot}(T_{\mathrm{txt}_\mathcal{T}(i)}, \mathrm{pos}_\mathcal{T}(i))$ for each $i \in [1..n]$, i.e., each position i in T corresponds to exactly one of the conjugates $C_\mathcal{T}(i)$ we are considering, which

in turn can be reported by means of this position in the event of a p-matching prefix. In our running example, $C_{\mathcal{T}}(3) = \mathrm{Rot}(T_{\mathrm{txt}_{\mathcal{T}}(3)}, \mathrm{pos}_{\mathcal{T}}(3)) = \mathrm{Rot}(T_1, 3-1) = \mathtt{bBa}$ and $C_{\mathcal{T}}(9) = \mathrm{Rot}(T_{\mathrm{txt}_{\mathcal{T}}(9)}, \mathrm{pos}_{\mathcal{T}}(9)) = \mathrm{Rot}(T_3, 9-8) = \mathtt{AbBBC}$. See Figure 5 for more examples.

3.1. Conjugate Array

We extend the ideas of Mantaci et al. [10] to p-matching, and introduce a new preorder on Σ^* to compare conjugates across texts. For notational convenience, we define the *rotational prev-encoding* $\langle V \rangle_{\mathrm{r}}$ of $V \in \Sigma^*$ by $\langle V \rangle_{\mathrm{r}} = \langle V^2 \rangle[|V|+1..]$.

Lemma 3.1. *Let $V, U \in \Sigma^+$. Then $\langle V \rangle_{\mathrm{r}} = \langle U \rangle_{\mathrm{r}}$ if and only if $V \approx U$.*

Proof. **($\Rightarrow$).** Let $U \not\approx V$. If $|U| \neq |V|$, then $\langle V \rangle_{\mathrm{r}} \neq \langle U \rangle_{\mathrm{r}}$. Hence, assume $|U| = |V|$. Then $\langle U \rangle \neq \langle V \rangle$ by Lemma 2.5. Let $i \in [1..|U|]$ minimal such that $\langle U \rangle[i] \neq \langle V \rangle[i]$. If $\max\{\langle U \rangle[i], \langle V \rangle[i]\} < \infty$, then $\langle U \rangle_{\mathrm{r}}[i] = \langle U \rangle[i] \neq \langle V \rangle[i] = \langle V \rangle_{\mathrm{r}}[i]$, i.e., $\langle V \rangle_{\mathrm{r}} \neq \langle U \rangle_{\mathrm{r}}$. Thus, assume $\max\{\langle U \rangle[i], \langle V \rangle[i]\} = \infty$. Without loss of generality, $\langle U \rangle[i] = \infty$. Then $\langle V \rangle_{\mathrm{r}}[i] = \langle V \rangle[i] < i \leq \langle U \rangle_{\mathrm{r}}[i]$, i.e., $\langle V \rangle_{\mathrm{r}} \neq \langle U \rangle_{\mathrm{r}}$.
($\Leftarrow$). Let $U \approx V$. Then $\langle U \rangle = \langle V \rangle$ by Lemma 2.5. In particular, $|\langle U \rangle_{\mathrm{r}}| = |\langle V \rangle_{\mathrm{r}}|$. Assume $\langle U \rangle_{\mathrm{r}} \neq \langle V \rangle_{\mathrm{r}}$. Let $i \in [1..|U|]$ minimal such that $\langle U \rangle_{\mathrm{r}} \neq \langle V \rangle_{\mathrm{r}}$. Since $\langle W \rangle[k] < \infty$ implies $\langle W \rangle[k] = \langle W \rangle_{\mathrm{r}}[k]$ for each $W \in \Sigma^+$ and $k \in [1..|W|]$, $\langle U \rangle[i] = \langle V \rangle[i] = \infty$, i.e., $\min\{\langle V \rangle_{\mathrm{r}}[i], \langle U \rangle_{\mathrm{r}}[i]\} \geq i$. Without loss of generality, $\langle V \rangle_{\mathrm{r}}[i] < \langle U \rangle_{\mathrm{r}}[i] \leq |V|$. Let $j = i - \langle V \rangle_{\mathrm{r}}[i] + |V|$. Then $\langle V \rangle[j] < \infty$, i.e., $V[j]$ occurs at least twice in V. However, $U[j] \neq U[i]$, which contradicts $U \approx V$ since $V[i] = V[j]$ is mapped to two different p-symbols or an s-symbol and a p-symbol. Hence, $\langle U \rangle_{\mathrm{r}} = \langle V \rangle_{\mathrm{r}}$. □

For any $U, V \in \Sigma^*$, we write $U \preceq_\omega V$ if and only if there exists an integer $i \geq 1$ such that $\langle U^\omega[..i] \rangle < \langle V^\omega[..i] \rangle$ or $\mathrm{root}(\langle V \rangle_{\mathrm{r}}) = \mathrm{root}(\langle U \rangle_{\mathrm{r}})$ holds.

Corollary 3.2. *The relation $\preceq_\omega$ defines a total preorder on Σ^*, i.e., the relation is binary, reflexive, transitive and connected.*

We call the relation $\preceq_\omega$ the *ω-preorder*. We write $U =_\omega V$ if and only if $U \preceq_\omega V \wedge V \preceq_\omega U$, and $U \prec_\omega V$ if and only if $U \preceq_\omega V \wedge U \neq_\omega V$. Note that $=_\omega$ is an equivalence relation. For example, $\mathtt{BBAA} \prec_\omega \mathtt{ABBA}$, $\mathtt{aBBaB} \prec_\omega \mathtt{aBB}$ and $\mathtt{CA} =_\omega \mathtt{BCBC} =_\omega \mathtt{ABABAB}$. The latter example also shows that the ω-preorder is not a total order since $\mathtt{CA} \neq \mathtt{BCBC} \wedge \mathtt{CA} =_\omega \mathtt{BCBC}$ violates antisymmetry. Adapting Hon et al. [20, Lemma 5], we give a convenient but not necessarily

optimal way to compute the ω-preorder of two p-strings making use of the weak periodicity lemma (cf. Crochemore and Rytter's book [45]).

Lemma 3.3 (Weak Periodicity Lemma)**.** *Let j and k be two periods of a string V. If $j + k \leq |V|$, then $\gcd(j, k)$ is also a period of V.*

Lemma 3.4. *Let $V, U \in \Sigma^+$ and $z = \max\{|V|, |U|\}$. Then $V =_\omega U$ if and only if $\langle V^\omega[..3z]\rangle = \langle U^\omega[..3z]\rangle$.*

Proof. Without loss of generality, $|U| = z$. Let $|V| = i$, $|\mathrm{root}(\langle V\rangle_\mathrm{r})| = j$ and $|\mathrm{root}(\langle U\rangle_\mathrm{r})| = k$.
($\Leftarrow$). Assume $\langle V^\omega[..3z]\rangle = \langle U^\omega[..3z]\rangle$. Then, on the one hand,

$$\langle \mathrm{Rot}(V, z)\rangle_\mathrm{r}^\omega[..2z] = \langle V^\omega[..3z]\rangle[z+1..] = \langle U^\omega[..3z]\rangle[z+1..] = \langle U\rangle_\mathrm{r} \cdot \langle U\rangle_\mathrm{r}.$$

Since the rotational prev-encoding is commutative with left rotations, j is a period of $\langle \mathrm{Rot}(V, z)\rangle_\mathrm{r}$. Consequently, both z and j are periods of $\langle U\rangle_\mathrm{r} \cdot \langle U\rangle_\mathrm{r}$. Since $j + z \leq 2z = |\langle U\rangle_\mathrm{r} \cdot \langle U\rangle_\mathrm{r}|$, we can apply Lemma 3.3 and find that $\gcd(j, z)$ is a period of $\langle U\rangle_\mathrm{r} \cdot \langle U\rangle_\mathrm{r}$. As $\gcd(j, z)$ divides $z = |\langle U\rangle_\mathrm{r}|$, $\langle U\rangle_\mathrm{r}$ can be formed by repeating $\langle U\rangle_\mathrm{r}[..\gcd(j, z)]$ an integral number of times, which implies $\gcd(j, z) \geq k$, i.e., $i \geq j \geq k$. On the other hand,

$$\langle V\rangle_\mathrm{r} \cdot \langle V\rangle_\mathrm{r} = \langle V^\omega[..3z]\rangle[i+1..3i] = \langle U^\omega[..3z]\rangle[i+1..3i] = \langle \mathrm{Rot}(U, i)\rangle_\mathrm{r}^\omega[..2i].$$

Since the rotational prev-encoding is commutative with left rotations, k is a period of $\langle \mathrm{Rot}(U, i)\rangle_\mathrm{r}$. As $k \leq i$, k is a period of $\langle V\rangle_\mathrm{r} \cdot \langle V\rangle_\mathrm{r}$ in addition to i. Then $k + i \leq 2i = |\langle V\rangle_\mathrm{r} \cdot \langle V\rangle_\mathrm{r}|$ and Lemma 3.3 imply that $\gcd(k, i)$ is a period of $\langle V\rangle_\mathrm{r} \cdot \langle V\rangle_\mathrm{r}$. As $\gcd(k, i)$ divides $i = |\langle V\rangle_\mathrm{r}|$, $\langle V\rangle_\mathrm{r}[..\gcd(k, i)]$ can be repeated an integral number of times to form $\langle V\rangle_\mathrm{r}$, which implies $\gcd(k, i) \geq j$. Consequently, $k \geq j$, and therefore $|\mathrm{root}(\langle V\rangle_\mathrm{r})| = j = k = |\mathrm{root}(\langle U\rangle_\mathrm{r})|$. In particular, $\mathrm{root}(\langle V\rangle_\mathrm{r}) = \langle V^\omega[..3z]\rangle[3z-j+1..] = \langle U^\omega[..3z]\rangle[3z-k+1..] = \mathrm{root}(\langle U\rangle_\mathrm{r})$, i.e., $V =_\omega U$.
($\Rightarrow$). Let $V =_\omega U$. Assume $\langle V^\omega[..3z]\rangle \neq \langle U^\omega[..3z]\rangle$ with x minimal such that $\langle V^\omega[..3z]\rangle[x] \neq \langle U^\omega[..3z]\rangle[x]$. If $\max\{\langle V^\omega[..3z]\rangle[x], \langle U^\omega[..3z]\rangle[x]\} < \mathbb{x}$, then $\mathrm{Rot}(\mathrm{root}(\langle V\rangle_\mathrm{r}), x-1)[1] = \langle V^\omega[..3z]\rangle[x] \neq \langle U^\omega[..3z]\rangle[x] = \mathrm{Rot}(\mathrm{root}(\langle U\rangle_\mathrm{r}), x-1)[1]$, which contradicts $V =_\omega U$. Hence, and without loss of generality, assume $\langle V^\omega[..3z]\rangle[x] = \mathbb{x}$. Then $\langle U^\omega[..3z]\rangle[x] < \mathbb{x}$, $x \leq |\mathrm{root}(\langle V\rangle_\mathrm{r})|$, and consequently $\mathrm{root}(\langle U\rangle_\mathrm{r})[x] = \langle U^\omega[..3z]\rangle[x] < x \leq \mathrm{root}(\langle V\rangle_\mathrm{r})[x]$, a contradiction to $V =_\omega U$. Thus, $\langle V^\omega[..3z]\rangle = \langle U^\omega[..3z]\rangle$. □

i	$C_{\mathcal{T}}(i)$	$\langle C_{\mathcal{T}}(i)^{\omega}[..15]\rangle$	$\mathrm{CA}_{\mathcal{T}}^{-1}[i]$	$\mathrm{CA}_{\mathcal{T}}[i]$	$\langle C_{\mathcal{T}}(\mathrm{CA}_{\mathcal{T}}[i])^{\omega}[..15]\rangle$
1	Bab	⌾ab3ab3ab3ab3ab	4	2	ab⌾ab3ab3ab3ab3
2	abB	ab⌾ab3ab3ab3ab3	1	3	b⌾ab3ab3ab3ab3a
3	bBa	b⌾ab3ab3ab3ab3a	2	10	b⌾1⌾⌾b4155b4155
4	ABBA	⌾⌾1313131313131	10	1	⌾ab3ab3ab3ab3ab
5	BBAA	⌾1⌾131313131313	6	9	⌾b⌾1⌾5b4155b415
6	BAAB	⌾⌾1313131313131	11	5	⌾1⌾131313131313
7	AABB	⌾1⌾131313131313	7	7	⌾1⌾131313131313
8	CAbBB	⌾⌾b⌾155b4155b41	9	11	⌾1⌾⌾b4155b4155b
9	AbBBC	⌾b⌾1⌾5b4155b415	5	8	⌾⌾b⌾155b4155b41
10	bBBCA	b⌾1⌾⌾b4155b4155	3	4	⌾⌾1313131313131
11	BBCAb	⌾1⌾⌾b4155b4155b	8	6	⌾⌾1313131313131
12	BCAbB	⌾⌾⌾b4155b4155b4	12	12	⌾⌾⌾b4155b4155b4

Figure 5: The conjugate array $\mathrm{CA}_{\mathcal{T}}$ of our running example $\mathcal{T} = \{\mathtt{Bab}, \mathtt{ABBA}, \mathtt{CAbBB}\}$.

Corollary 3.5. *Let $V, U \in \Sigma^+$ and $z = \max\{|V|, |U|\}$. Then $V \prec_{\omega} U$ if and only if $\langle V^{\omega}[..3z]\rangle < \langle U^{\omega}[..3z]\rangle$.*

We generalize the suffix array similarly to Boucher et al. [26]. The *conjugate array* $\mathrm{CA}_{\mathcal{T}}$ of $\mathcal{T}$ is a string of length n over $[1..n]$ such that $\mathrm{CA}_{\mathcal{T}}[i] = j$ if and only if $i-1 = |\{k \in [1..n] \mid C_{\mathcal{T}}(k) \prec_{\omega} C_{\mathcal{T}}(j) \vee (C_{\mathcal{T}}(k) =_{\omega} C_{\mathcal{T}}(j) \wedge k < j)\}|$, i.e., $i-1$ is the number of all conjugates smaller than $C_{\mathcal{T}}(j)$ according to ω-preorder, where we break ties first with respect to text index, and then with respect to text position. By resolving all ties this way, we enforce a linear order on the conjugates considered in $\mathcal{T}$, and conclude that $\mathrm{CA}_{\mathcal{T}}$ is well-defined. Since $\mathrm{CA}_{\mathcal{T}}$ is a permutation, its inverse, which we denote by $\mathrm{CA}_{\mathcal{T}}^{-1}$, is also well-defined. For the conjugate array of our running example, see Figure 5.

The *conjugate range* $\mathrm{CR}_{\mathcal{T}}(P)$ of a pattern $P \in \Sigma^*$ of length m and $\mathcal{T}$ is a maximal interval $[\ell..r] \subseteq [1..n]$ such that $P \approx C_{\mathcal{T}}(\mathrm{CA}_{\mathcal{T}}[i])^{\omega}[..m]$ for every $i \in [\ell..r]$. Note that $\mathrm{CR}_{\mathcal{T}}(\varepsilon) = [1..n]$. The following result is a consequence of Lemma 2.5.

Corollary 3.6. *Let $T_1, ..., T_d \in \Sigma^+$, $n = |T_1 \cdots T_d|$, $P \in \Sigma^*$, $m = |P|$ and $[\ell..r] = \mathrm{CR}_{\mathcal{T}}(P)$. Then $P \approx C_{\mathcal{T}}(\mathrm{CA}_{\mathcal{T}}[i])^{\omega}[..m]$ if and only if $i \in \mathrm{CR}_{\mathcal{T}}(P)$.*

We call the computation of $\mathrm{CR}_{\mathcal{T}}(P)$ for some pattern $P \in \Sigma^*$ the *backward search* for P in $\mathcal{T}$ and the length of $\mathrm{CR}_{\mathcal{T}}(P)$ is the return value to a count query for P in $\mathcal{T}$ by Corollary 3.6, i.e., the solution to EP-COUNT.

i	$T[i]$	$\mathrm{LEN}_{\mathcal{T}}[i]$	$\mathrm{ENC}_{\mathcal{T}}[i]$	$\mathrm{PRV}_{\mathcal{T}}[i]$	$\mathrm{prev}_{\mathcal{T}}(i)$
1	B	1	1	1	3
2	a	0	a	0	1
3	b	0	b	0	2
4	A	1	2	1	5
5	B	0	1	0	4
6	B	0	2	1	7
7	A	0	1	0	6
8	C	1	3	1	12
9	A	0	3	0	8
10	b	0	b	0	9
11	B	0	1	0	10
12	B	0	3	0	11

Figure 6: The permutation $\mathrm{prev}_{\mathcal{T}}$ of our running example and its associated data structures. Here, $\mathrm{LEN}_{\mathcal{T}}[13] = \mathrm{PRV}_{\mathcal{T}}[13] = 1$.

For our running example, we consider the pattern p-strings $P_1 = \mathtt{ACAB}$, $P_2 = \mathtt{CCB}$ and $P_3 = \mathtt{a}$. Then $\mathrm{CR}_{\mathcal{T}}(P_1) = \emptyset$, $\mathrm{CR}_{\mathcal{T}}(P_2) = [6..8]$ and $\mathrm{CR}_{\mathcal{T}}(\mathtt{P_3}) = [1..1]$, and the count queries for P_1, P_2 and P_3 return 0, 3 and 1, respectively. In what follows, we define the necessary tools to allow for an efficient backward search.

3.2. LF-Mapping

Unlike Mantaci et al. [10], we did not assume our texts or their encodings to be primitive, e.g., $[\![T_2]\!] = \mathtt{2121}$ is not primitive. This warrants the following. Let $\mathrm{ENC}_{\mathcal{T}} = [\![T_1]\!] \cdots [\![T_d]\!] = \mathrm{root}([\![T_1]\!])^{\exp([\![T_1]\!])} \cdots \mathrm{root}([\![T_d]\!])^{\exp([\![T_d]\!])}$. We define a bit string $\mathrm{PRV}_{\mathcal{T}}$ of length $n+1$ marking the positions of $\mathrm{ENC}_{\mathcal{T}}$ with a 1 if the position is a starting position of an encoded text root, and set $\mathrm{PRV}_{\mathcal{T}}[n+1] = 1$ as a sentinel. Note that $\mathrm{LEN}_{\mathcal{T}} = \mathrm{PRV}_{\mathcal{T}}$ if and only if $[\![T_k]\!]$ is primitive for every $k \in [1..d]$. For every $i \in [1..n]$, let $\mathrm{prev}_{\mathcal{T}}(i) = \mathrm{select}_1(\mathrm{PRV}_{\mathcal{T}}, \mathrm{rank}_1(\mathrm{PRV}_{\mathcal{T}}, i) + 1) - 1$ if $\mathrm{PRV}_{\mathcal{T}}[i] = 1$, or $\mathrm{prev}_{\mathcal{T}}(i) = i - 1$ otherwise, i.e., the permutation $\mathrm{prev}_{\mathcal{T}}$ of $[1..n]$ moves circularly to the previous position inside a root of an encoded input text. See Figure 6 for an example. Note that $\mathrm{prev}_{\mathcal{T}}$ can be derived from the rotational prev-encoding instead by Lemma 2.5 and Lemma 3.1.

Then the *LF-mapping* $\mathrm{LF}_{\mathcal{T}}$ of $\mathcal{T}$ is a string of length n over $[1..n]$ such that $\mathrm{LF}_{\mathcal{T}}[i] = \mathrm{CA}_{\mathcal{T}}^{-1}[\mathrm{prev}_{\mathcal{T}}(\mathrm{CA}_{\mathcal{T}}[i])]$ for each $i \in [1..n]$. Since the LF-mapping is

a permutation, it has an inverse $\mathrm{LF}_{\mathcal{T}}^{-1}$, which we call the *FL-mapping* of $\mathcal{T}$ and denote by $\mathrm{FL}_{\mathcal{T}}$. The LF-mapping will be at the core of the backward search. However, storing LF-mapping and FL-mapping in their plain form creates the need for two integer arrays of length n and entries of $\lg n$ bits. To represent both the LF- and FL-mapping of $\mathcal{T}$ in compact space, we need one more result on the ω-preorder of two strings, which now follows.

Lemma 3.7. *Let $U, V \in \Sigma^+$, $e = \mathrm{lcp}^{\infty}(\mathrm{Rot}(U,1), \mathrm{Rot}(V,1))$ and assume $\mathrm{Rot}(U,1) \prec_{\omega} \mathrm{Rot}(V,1)$. Then $V \prec_{\omega} U$ if and only if $\pi(V) < \pi(U) \wedge (\pi(V) \in \Sigma_{\mathrm{s}} \vee (\pi(U), \pi(V) \notin \Sigma_{\mathrm{s}} \wedge \pi(V) \leq e))$.*

Proof. Application of Lemma 2.6 and Corollary 3.5. □

Finally, we introduce our representation of the LF- and FL-mapping, which consists of two strings $\mathrm{L}_{\mathcal{T}}$ and $\mathrm{F}_{\mathcal{T}}$, and are defined as follows. First, the *extended parameterized Burrows–Wheeler Transform (epBWT)* $\mathrm{L}_{\mathcal{T}}$ of $\mathcal{T}$ is defined as a string of length n over $[1..\sigma_{\mathrm{p}}] \cup \Sigma_{\mathrm{s}}$ such that $\mathrm{L}_{\mathcal{T}}[i] = \pi(C_{\mathcal{T}}(\mathrm{CA}_{\mathcal{T}}[\mathrm{LF}_{\mathcal{T}}[i]]))$ for each $i \in [1..n]$.

Lemma 3.8 (Non-crossing Property)**.** *Let $\mathcal{T} \subset \Sigma^+$ and $i, j \in [1..n]$ with $i < j$, where n is the accumulated length of all strings of $\mathcal{T}$. If $\mathrm{L}_{\mathcal{T}}[i] = \mathrm{L}_{\mathcal{T}}[j]$, then $\mathrm{LF}_{\mathcal{T}}[i] < \mathrm{LF}_{\mathcal{T}}[j]$.*

Proof. Let $\mathrm{L}_{\mathcal{T}}[i] = \mathrm{L}_{\mathcal{T}}[j]$. Since $C_{\mathcal{T}}(\mathrm{CA}_{\mathcal{T}}[k]) = \mathrm{Rot}(C_{\mathcal{T}}(\mathrm{prev}_{\mathcal{T}}(\mathrm{CA}_{\mathcal{T}}[k])), 1)$ for every $k \in [1..n]$, Lemma 3.7 implies

$$C_{\mathcal{T}}(\mathrm{prev}_{\mathcal{T}}(\mathrm{CA}_{\mathcal{T}}[i])) \preceq_{\omega} C_{\mathcal{T}}(\mathrm{prev}_{\mathcal{T}}(\mathrm{CA}_{\mathcal{T}}[j])) \Leftrightarrow C_{\mathcal{T}}(\mathrm{CA}_{\mathcal{T}}[i]) \preceq_{\omega} C_{\mathcal{T}}(\mathrm{CA}_{\mathcal{T}}[j]).$$

As $i < j$, $C_{\mathcal{T}}(\mathrm{prev}_{\mathcal{T}}(\mathrm{CA}_{\mathcal{T}}[i])) \preceq_{\omega} C_{\mathcal{T}}(\mathrm{prev}_{\mathcal{T}}(\mathrm{CA}_{\mathcal{T}}[j]))$. Then by definition of the conjugate array, $\mathrm{CA}_{\mathcal{T}}^{-1}[\mathrm{prev}_{\mathcal{T}}(\mathrm{CA}_{\mathcal{T}}[i])] < \mathrm{CA}_{\mathcal{T}}^{-1}[\mathrm{prev}_{\mathcal{T}}(\mathrm{CA}_{\mathcal{T}}[j])]$, i.e., $\mathrm{LF}_{\mathcal{T}}[i] < \mathrm{LF}_{\mathcal{T}}[j]$. □

Second, let $\mathrm{F}_{\mathcal{T}}$ denote a string of length n over $[1..\sigma_{\mathrm{p}}] \cup \Sigma_{\mathrm{s}}$ such that $\mathrm{F}_{\mathcal{T}}[\mathrm{LF}_{\mathcal{T}}[i]] = \mathrm{L}_{\mathcal{T}}[i]$ for each $i \in [1..n]$. By the following result, $\mathrm{L}_{\mathcal{T}}$ and $\mathrm{F}_{\mathcal{T}}$ suffice to compute both LF- and FL-mapping of $\mathcal{T}$.

Corollary 3.9. *Let $\mathcal{T} \subset \Sigma^+$ and n be the accumulated length of all strings of $\mathcal{T}$. Then*

$$\mathrm{LF}_{\mathcal{T}}[i] = \mathrm{select}_{\mathrm{L}_{\mathcal{T}}[i]}(\mathrm{F}_{\mathcal{T}}, \mathrm{rank}_{\mathrm{L}_{\mathcal{T}}[i]}(\mathrm{L}_{\mathcal{T}}, i)) \text{ and}$$
$$\mathrm{FL}_{\mathcal{T}}[i] = \mathrm{select}_{\mathrm{F}_{\mathcal{T}}[i]}(\mathrm{L}_{\mathcal{T}}, \mathrm{rank}_{\mathrm{F}_{\mathcal{T}}[i]}(\mathrm{F}_{\mathcal{T}}, i)) \text{ for each } i \in [1..n].$$

We also achieve reversibility to a degree.

Corollary 3.10. *Let $\mathcal{T} = \{T_1, ..., T_d\} \subset \Sigma^+$. Given $\mathrm{F}_\mathcal{T}$ and $\mathrm{L}_\mathcal{T}$, we can restore the conjugacy classes of $\mathcal{T}$ up to $=_\omega$-equivalence. In particular, we can restore the conjugacy classes of $\mathcal{T}$ up to p-match equivalence if $[\![T]\!]_1, ..., [\![T]\!]_d$ are primitive.*

Proof. We obtain the text-roots of the $[\![..]\!]$-encoded input texts from the cycles of the LF-mapping and either $\mathrm{F}_\mathcal{T}$ or $\mathrm{L}_\mathcal{T}$. Since the LF-mapping can be computed from $\mathrm{F}_\mathcal{T}$ and $\mathrm{L}_\mathcal{T}$ by Corollary 3.9, the claims follow from Lemma 2.5 and Lemma 3.1. □

The LF-mapping of our running example can be found in Figure 7. To restore the input texts, we examine its cycles. We start with $\mathrm{F}_\mathcal{T}[1] = \mathtt{a}$ and compute $\mathrm{LF}_\mathcal{T}[1] = 4$, which yields $\mathrm{F}_\mathcal{T}[4] \cdot \mathrm{F}_\mathcal{T}[1] = \mathtt{1a}$. Next, we proceed to compute $\mathrm{LF}_\mathcal{T}[4] = 2$ resulting in $\mathrm{F}_\mathcal{T}[2] \cdot \mathrm{F}_\mathcal{T}[4] \cdot \mathrm{F}_\mathcal{T}[1] = \mathtt{b1a}$. Since $\mathrm{LF}_\mathcal{T}[2] = 1$, the cycle ends; we have found a root of a $[\![..]\!]$-encoded conjugate of an input text string, namely $[\![\mathrm{Rot}(T_1, 2)]\!] = \mathtt{b1a}$. If we do this for all the other cycles of the LF-mapping, we obtain $\mathrm{F}_\mathcal{T}[8] \cdot \mathrm{F}_\mathcal{T}[12] \cdot \mathrm{F}_\mathcal{T}[9] \cdot \mathrm{F}_\mathcal{T}[5] \cdot \mathrm{F}_\mathcal{T}[3] = \mathtt{1333b} = [\![\mathrm{Rot}(T_3, 3)]\!]$ and $\mathrm{F}_\mathcal{T}[10] \cdot \mathrm{F}_\mathcal{T}[6] = \mathrm{F}_\mathcal{T}[11] \cdot \mathrm{F}_\mathcal{T}[7] = 21 = \mathrm{root}([\![T_2]\!])$.

To enable restoration up to p-match and conjugacy class without the assumption that the $[\![..]\!]$-encoded input texts are primitive, we can proceed as follows. For each $k \in [1..d]$, our goal is to ignore all but one cycle in the LF-mapping corresponding to T_k and store $\exp([\![T_k]\!])$ in an easily accessible way. Let $\mathrm{D}_\mathcal{T}$ denote a string of length n over $[1..n]$ such that $\mathrm{D}_\mathcal{T}[i] = \exp([\![T_{\mathrm{txt}_\mathcal{T}(\mathrm{CA}_\mathcal{T}[i])}]\!])$ if $\mathrm{select}_1(\mathrm{LEN}_\mathcal{T}, \mathrm{rank}_1(\mathrm{LEN}_\mathcal{T}, \mathrm{CA}_\mathcal{T}[i])) = \mathrm{select}_1(\mathrm{PRV}_\mathcal{T}, \mathrm{rank}_1(\mathrm{PRV}_\mathcal{T}, \mathrm{CA}_\mathcal{T}[i]))$, and $\mathrm{D}_\mathcal{T}[i] = 0$ otherwise, i.e., non-zero entries of $\mathrm{D}_\mathcal{T}$ correspond to entries in the first iteration of a root of a $[\![..]\!]$-encoded text and yield the exponent of the corresponding $[\![..]\!]$-encoded text. Restoration of up to p-match and conjugacy class is then straight-forward.

Note that we have to construct $\mathrm{D}_\mathcal{T}$ during the construction of $\mathrm{F}_\mathcal{T}$ and $\mathrm{L}_\mathcal{T}$ since we are unable to discern the cycles of any two texts that have ω-equal conjugates. Moreover, since $\sum_{i=1}^{n} \mathrm{D}_\mathcal{T}[i] = n$, we can encode $\mathrm{D}_\mathcal{T}$ as a bit string $\mathrm{D}'_\mathcal{T}$ of length $2n$, where entries of $\mathrm{D}_\mathcal{T}$ are encoded as unary zeros that are separated by '1's with an additional '1' as a sentinel. Then $\mathrm{D}_\mathcal{T}[i] = \mathrm{rank}_0(\mathrm{D}'_\mathcal{T}, \mathrm{select}_1(\mathrm{D}'_\mathcal{T}, i))$. Thus, $\mathrm{D}_\mathcal{T}$ can be stored in $O(n)$ bits of space if its binary encoding $\mathrm{D}'_\mathcal{T}$ is represented by the data structure of Lemma 2.1 or Lemma 2.3.

In our running example, $[\![T_2]\!]$ is not primitive. Consequently, $\mathrm{D}_{\mathcal{T}} = \texttt{111112011201}$, which we encode as $\mathrm{D}'_{\mathcal{T}} = \texttt{010101010100110101001101}$. The LF-mapping yields four cycles as before. However, for the cycle (7 11) we get $\mathrm{D}_{\mathcal{T}}[7] = \mathrm{D}_{\mathcal{T}}[11] = 0$ indicating that the cycle does not correspond to the first iteration of a root within a $[\![..]\!]$-encoded indexed text. Thus, we discard the cycle. The cycle (6 10), on the other hand, corresponds to the first iteration of a root within a $[\![..]\!]$-encoded indexed text, and $\mathrm{D}_{\mathcal{T}}[6] = \mathrm{D}_{\mathcal{T}}[10] = 2$ yields the number of iterations to restore the text T_2 up to conjugacy class and p-match equivalence.

3.3. Backward Search

The backward search for a pattern $P \in \Sigma^*$ in $\mathcal{T}$ consists of updating the conjugate range of $P[i+1..]$ to that of $P[i..]$ for each $i \in [1..\,|P|]$, where $\mathrm{CR}_{\mathcal{T}}(P[|P|+1..]) = \mathrm{CR}_{\mathcal{T}}(\varepsilon) = [1..n]$. In what follows, we present two different methods to compute these conjugate range updates, where the second one is motivated by our construction algorithm of the epBWT presented in Section 4. The first one has better complexities while the second one allows for an efficient construction and the possibilities to add and remove indexed texts. For the first method, we adapt the findings of Kim and Cho [14].

Lemma 3.11. *Let $\mathcal{T} \subset \Sigma^+$ and $P \in \Sigma^*$. Then Algorithm 1 correctly computes $[\ell'..r'] = \mathrm{CR}_{\mathcal{T}}(P[i..])$ for each $i \in [1..|P|]$.*

Proof. Let $m = |P|$, $i \in [1..m]$, $h = \pi(P[i..])$, $[\ell..r] = \mathrm{CR}_{\mathcal{T}}(P[i+1..])$ and $c = |P[i+1..]|_{\mathrm{p}}$. Algorithm 1 takes $[\ell..r]$, h, c, $\mathrm{F}_{\mathcal{T}}$, $\mathrm{L}_{\mathcal{T}}$ and $\mathrm{LF}_{\mathcal{T}}$ as input.

By definition, $h \leq c+1$. For the computation of r' and the length $\delta = r' - \ell' + 1$ of $[\ell'..r']$, we consider two cases depending on h, with the first being treated from Line 2 through Line 6 and the second from Line 7 through Line 11. If $\delta > 0$, then $\ell' = r' - \delta + 1$, which is computed in Line 12.
Case 1. Assume $h \leq c$, i.e., $h \in \Sigma_{\mathrm{s}}$ or $h \notin \Sigma_{\mathrm{s}}$ and there exists some $j \in [i+1..m]$ such that $P[i] = P[j]$. Then $C_{\mathcal{T}}(\mathrm{CA}_{\mathcal{T}}[\mathrm{LF}_{\mathcal{T}}[j]])^{\omega}[..m-i+1] \approx P[i..]$ if and only if $C_{\mathcal{T}}(\mathrm{CA}_{\mathcal{T}}[j])^{\omega}[..m-i] \approx P[i+1]$ and $\mathrm{L}_{\mathcal{T}}[j] = h$, for each $j \in [1..n]$. Thus, $\mathrm{LF}_{\mathcal{T}}[j] \in [\ell'..r']$ if and only if $j \in [\ell..r]$ and $\mathrm{L}_{\mathcal{T}}[j] = h$. Then $\delta = \mathrm{rnkcnt}_{\mathrm{L}_{\mathcal{T}}}(\ell, r, h, h)$, which is computed in Line 3. If $\delta \leq 0$, then we return an empty interval in Line 5. If $\delta > 0$, then the non-crossing property of Lemma 3.8 already implies $r' = \mathrm{LF}_{\mathcal{T}}[\mathrm{select}_h(\mathrm{L}_{\mathcal{T}}, \mathrm{rank}_h(\mathrm{L}_{\mathcal{T}}, r))]$, which is computed in Line 6.
Case 2. Assume $h = c+1$, i.e., $h \notin \Sigma_{\mathrm{s}}$ and $P[i] \neq P[j]$ for each $j \in [i+1..m]$. Then $C_{\mathcal{T}}(\mathrm{CA}_{\mathcal{T}}[\mathrm{LF}_{\mathcal{T}}[j]])^{\omega}[..m-i+1] \approx P[i..]$ if and only if $C_{\mathcal{T}}(\mathrm{CA}_{\mathcal{T}}[j])^{\omega}[..m-$

Algorithm 1: Updating $[\ell..r] = \mathrm{CR}_{\mathcal{T}}(P[i+1..])$ to $[\ell'..r'] = \mathrm{CR}_{\mathcal{T}}(P[i..])$. Here, $h = \pi(P[i..])$ and $c = |P[i+1..]|_{\mathrm{p}}$.

```
1  Function crangeupdrmq([ℓ..r], h, c, F_T, L_T, LF_T):
2      if h ≤ c then                                   // Case 1 in Lemma 3.11
3          δ ← rnkcnt_{L_T}(ℓ, r, h, h);
4          if δ ≤ 0 then
5              return [r + 1..r];                      // empty interval
6          r' ← LF_T[select_h(L_T, rank_h(L_T, r))];
7      else                                            // Case 2 in Lemma 3.11
8          δ ← rnkcnt_{L_T}(ℓ, r, h, σ_p);
9          if δ ≤ 0 then
10             return [r + 1..r];                      // empty interval
11         r' ← LF_T[RMQ_{LF_T}(ℓ, r)];
12     ℓ' ← r' − δ + 1;
13     return [ℓ'..r'];
```

$i] \approx P[i+1..]$ and $\mathrm{L}_{\mathcal{T}}[j] > c$, for each $j \in [1..n]$. Thus, $\mathrm{LF}_{\mathcal{T}}[j] \in [\ell'..r']$ if and only if $j \in [\ell..r]$ and $\mathrm{L}_{\mathcal{T}}[j] > c$. Consequently, $\delta = \mathrm{rnkcnt}_{\mathrm{L}_{\mathcal{T}}}(\ell, r, h, \sigma_{\mathrm{p}})$, which is computed in Line 8. If $\delta \leq 0$, then we return an empty interval in Line 10. By Lemma 3.7, $C_{\mathcal{T}}(\mathrm{CA}_{\mathcal{T}}[\mathrm{LF}_{\mathcal{T}}[j]]) \prec_{\omega} C_{\mathcal{T}}(\mathrm{CA}_{\mathcal{T}}[\mathrm{LF}_{\mathcal{T}}[k]])$ for each $j, k \in [\ell..r]$ with $\mathrm{L}_{\mathcal{T}}[j] \leq c$ and $\mathrm{L}_{\mathcal{T}}[k] > c$. Hence, $r' = \max\{\mathrm{LF}_{\mathcal{T}}[j] \mid j \in [\ell..r]\}$, which we compute in Line 11. □

Finally, we preprocess the pattern such that we can access the encoding of its suffixes in constant time.

Lemma 3.12 ([13, Section 5.2])**.** *Given a p-string P of length m, we can process P in $O(m \lg \sigma)$ time such that we can subsequently compute $\pi(P[i..])$ and $|P[i+1..]|_{\mathrm{p}}$ in $O(1)$ time, for every $i \in [1..m]$.*

Theorem 3.13. *Let $\mathcal{T} \subset \Sigma^+$ and n be the accumulated length of all strings of $\mathcal{T}$. There exists an index solving* EP-COUNT *in $O(m \lg \sigma)$ time, where m is the length of the queried pattern p-string. The index takes $2n \lg \sigma + 2n + O(n \frac{\lg \lg n}{\lg n})$ bits of space.*

Proof. The index consists of representations of $\mathrm{F}_{\mathcal{T}}$, $\mathrm{L}_{\mathcal{T}}$ and $\mathrm{LF}_{\mathcal{T}}$. We represent $\mathrm{F}_{\mathcal{T}}$ and $\mathrm{L}_{\mathcal{T}}$ by the data structure of Lemma 2.1 and $\mathrm{LF}_{\mathcal{T}}$ by the

data structure of Lemma 2.2, which leads to the claimed space complexity. Let $P \in \Sigma^*$ with $m = |P|$. We preprocess P with Lemma 3.12. Then the claim regarding count queries follows from Lemma 2.1, Lemma 2.2, and Lemma 3.11. □

For the second method, we took inspiration from Hashimoto et al. [39] and Iseri et al. [15]. Let $\mathrm{LCP}^{\infty}_{\mathcal{T}}$ denote a string of length n over $[0..\sigma_{\mathrm{p}}]$ such that $\mathrm{LCP}^{\infty}_{\mathcal{T}}[1] = 0$ and $\mathrm{LCP}^{\infty}_{\mathcal{T}}[i] = \mathrm{lcp}^{\infty}(C_{\mathcal{T}}(\mathrm{CA}_{\mathcal{T}}[i]), C_{\mathcal{T}}(\mathrm{CA}_{\mathcal{T}}[i-1]))$ for $i \in [2..n]$.

Lemma 3.14. *Let $\mathcal{T} \subset \Sigma^+$ and n be the accumulated length of all strings of $\mathcal{T}$. Then for each $i, j \in [1..n]$ with $i < j$,*

$$\mathrm{lcp}^{\infty}(C_{\mathcal{T}}(\mathrm{CA}_{\mathcal{T}}[i]), C_{\mathcal{T}}(\mathrm{CA}_{\mathcal{T}}[j])) = \mathrm{RNV}_{\mathrm{LCP}^{\infty}_{\mathcal{T}}}(i+1, j, -1).$$

Proof. Let $U, V, W \in \Sigma^+$ and $z = \max\{|U|, |V|, |W|\}$. If $\langle U^{\omega}[..3z]\rangle \leq \langle V^{\omega}[..3z]\rangle \leq \langle W^{\omega}[..3z]\rangle$, then we conclude that $\mathrm{lcp}(\langle U^{\omega}[..3z]\rangle, \langle W^{\omega}[..3z]\rangle) = \min\{\mathrm{lcp}(\langle U^{\omega}[..3z]\rangle, \langle V^{\omega}[..3z]\rangle), \mathrm{lcp}(\langle V^{\omega}[..3z]\rangle, \langle W^{\omega}[..3z]\rangle)\}$. By Lemma 3.4, $U \preceq_{\omega} V \preceq_{\omega} W$ implies $\mathrm{lcp}^{\infty}(U, W) = \min\{\mathrm{lcp}^{\infty}(U, V), \mathrm{lcp}^{\infty}(V, W)\}$.

Let $i, j \in [1..n]$ and $i < j$. Then $\mathrm{lcp}^{\infty}(C_{\mathcal{T}}(\mathrm{CA}_{\mathcal{T}}[i]), C_{\mathcal{T}}(\mathrm{CA}_{\mathcal{T}}[j])) = \min\{\mathrm{LCP}^{\infty}_{\mathcal{T}}[k] \mid k \in [i+1..j]\} = \mathrm{RNV}_{\mathrm{LCP}^{\infty}_{\mathcal{T}}}(i+1, j, -1)$. □

Lemma 3.15. *Let $\mathcal{T} \subset \Sigma^+$ and $P \in \Sigma^*$. Then Algorithm 2 correctly computes $[\ell'..r'] = \mathrm{CR}_{\mathcal{T}}(P[i..])$ for each $i \in [1..|P|]$.*

Proof. Let $m = |P|$, $i \in [1..m]$, $h = \pi(P[i..])$, $[\ell..r] = \mathrm{CR}_{\mathcal{T}}(P[i+1..])$ and $c = |P[i+1..]|_{\mathrm{p}}$. Algorithm 2 takes $[\ell..r]$, h, c, $\mathrm{F}_{\mathcal{T}}$, $\mathrm{L}_{\mathcal{T}}$ and $\mathrm{LCP}^{\infty}_{\mathcal{T}}$ as input.

By definition, $h \leq c + 1$. The case where $h \leq c$ is treated from Line 2 through Line 6, and the proof for correctness is the same as that of Case 1 in the proof of Lemma 3.11. Thus, assume $h = c + 1$, which we treat from Line 7 through Line 16. Then $C_{\mathcal{T}}(\mathrm{CA}_{\mathcal{T}}[\mathrm{LF}_{\mathcal{T}}[j]])^{\omega}[..m-i+1] \approx P[i..]$ if and only if $C_{\mathcal{T}}(\mathrm{CA}_{\mathcal{T}}[j])^{\omega}[..m-i] \approx P[i+1..]$ and $\mathrm{L}_{\mathcal{T}}[j] > c$, for each $j \in [1..n]$. Hence, $\mathrm{LF}_{\mathcal{T}}[j] \in [\ell'..r']$ if and only if $j \in [\ell..r]$ and $\mathrm{L}_{\mathcal{T}}[j] > c$. Consequently, $\delta = |\mathrm{CR}_{\mathcal{T}}(P[i..])| = \mathrm{rnkcnt}_{\mathrm{L}_{\mathcal{T}}}(\ell, r, h, \sigma_{\mathrm{p}})$, which is computed in Line 8. If $\delta \leq 0$, then we return an empty interval in Line 10.

If $\delta > 0$, then we compute the highest index x of $\mathrm{L}_{\mathcal{T}}$ such that $x \in [\ell..r]$ and $\mathrm{L}_{\mathcal{T}}[x]$ takes the lowest value v in $\mathrm{L}_{\mathcal{T}}[\ell..r]$ higher than c in Line 11 and Line 12. We then proceed to compute $y = \mathrm{LF}_{\mathcal{T}}[x] - \ell'$, i.e., we count the entries $j \in [\ell..r]$ with $\mathrm{L}_{\mathcal{T}}[j] \geq h$ and $j \neq x$ such that $C_{\mathcal{T}}(\mathrm{LF}_{\mathcal{T}}[j]) \preceq_{\omega}$

Algorithm 2: Updating $[\ell..r] = \mathrm{CR}_{\mathcal{T}}(P[i+1..])$ to $[\ell'..r'] = \mathrm{CR}_{\mathcal{T}}(P[i..])$. Here, $h = \pi(P[i..])$ and $c = |P[i+1..]|_{\mathrm{p}}$.

```
1  Function crangeupdlcp([ℓ..r], h, c, F_T, L_T, LCP^∞_T):
2      if h ≤ c then
3          δ ← rnkcnt_{L_T}(ℓ, r, h, h);
4          if δ ≤ 0 then
5              return [r + 1..r];                          // empty interval
6          r' ← LF_T[select_h(L_T, rank_h(L_T, r))];
7      else                                                // Case in Lemma 3.15
8          δ ← rnkcnt_{L_T}(ℓ, r, h, σ_p);
9          if δ ≤ 0 then
10             return [r + 1..r];                          // empty interval
11         v ← RNV_{L_T}(ℓ, r, c);
12         x ← select_v(L_T, rank_v(L_T, r));
13         [ℓ''..r''] ← MI_{LCP^∞_T}(x, v);
14         y ← rnkcnt_{L_T}(max{ℓ, ℓ''}, x − 1, v, v);
15         y ← y + rnkcnt_{L_T}(ℓ, ℓ'' − 1, v, σ_p);
16         r' ← LF_T[x] + δ − y − 1;
17     ℓ' ← r' − δ + 1;
18     return [ℓ'..r'];
```

$C_{\mathcal{T}}(\mathrm{LF}_{\mathcal{T}}[x])$. To that end, let $[\ell''..r''] = \mathrm{MI}_{\mathrm{LCP}^{\infty}_{\mathcal{T}}}(x, v)$ as computed in Line 13. Then $\mathrm{lcp}^{\infty}(C_{\mathcal{T}}(\mathrm{CA}_{\mathcal{T}}[x]), C_{\mathcal{T}}(\mathrm{CA}_{\mathcal{T}}[j])) \geq v$ if and only if $j \in [\ell''..r'']$.

By Lemma 3.7 and choice of x, $C_{\mathcal{T}}(\mathrm{LF}_{\mathcal{T}}[x]) \prec_{\omega} C_{\mathcal{T}}(\mathrm{LF}_{\mathcal{T}}[j])$ for each $j \in [x+1..r]$ with $\mathrm{L}_{\mathcal{T}}[j] \geq h$, i.e., we do not need to consider the entries of $[x+1..r]$ in our computation of y. For each $j \in [\max\{\ell, \ell''\}..x-1]$ with $\mathrm{L}_{\mathcal{T}}[j] \geq h$, $C_{\mathcal{T}}(\mathrm{LF}_{\mathcal{T}}[j]) \preceq_{\omega} C_{\mathcal{T}}(\mathrm{LF}_{\mathcal{T}}[x])$ if and only if $\mathrm{L}_{\mathcal{T}}[j] = v$ by Lemma 3.7 and choice of $[\ell''..r'']$. Again by Lemma 3.7 and choice of $[\ell''..r'']$, $C_{\mathcal{T}}(\mathrm{LF}_{\mathcal{T}}[j]) \preceq_{\omega} C_{\mathcal{T}}(\mathrm{LF}_{\mathcal{T}}[x])$ for each $j \in [\ell..\ell''-1]$ with $\mathrm{L}_{\mathcal{T}}[j] \geq h$.

Then the non-crossing property implies $y = \mathrm{rnkcnt}_{\mathrm{L}_{\mathcal{T}}}(\max\{\ell, \ell''\}, x - 1, v, v) + \mathrm{rnkcnt}_{\mathrm{L}_{\mathcal{T}}}(\ell, \ell'' - 1, v, \sigma_{\mathrm{p}})$, which is computed in Line 14 and Line 15. The computation of r' and ℓ' in Line 16 and Line 17, respectively, is like before. □

Theorem 3.16. *Let $\mathcal{T} \subset \Sigma^+$ and n be the accumulated length of all strings*

i	$\mathrm{CA}_{\mathcal{T}}[i]$	$C_{\mathcal{T}}(\mathrm{CA}_{\mathcal{T}}[i])$	$\mathrm{F}_{\mathcal{T}}[i]$	$\mathrm{L}_{\mathcal{T}}[i]$	$\mathrm{LCP}^{\infty}_{\mathcal{T}}[i]$	$\mathrm{LF}_{\mathcal{T}}[i]$	$\mathrm{FL}_{\mathcal{T}}[i]$
1	2	abB	a	1	0	4	2
2	3	bBa	b	a	0	1	4
3	10	bBBCA	b	3	1	5	8
4	1	Bab	1	b	0	2	1
5	9	AbBBC	3	3	1	9	3
6	5	BBAA	1	2	1	10	10
7	7	AABB	1	2	2	11	11
8	11	BBCAb	1	b	2	3	12
9	8	CAbBB	3	3	1	12	5
10	4	ABBA	2	1	2	6	6
11	6	BAAB	2	1	2	7	7
12	12	BCAbB	3	1	2	8	9

Figure 7: The epBWT index of our running example $\mathcal{T} = \{\mathtt{Bab}, \mathtt{ABBA}, \mathtt{CAbBB}\}$.

of $\mathcal{T}$. There exists an index solving ep-Count *in $O(m\frac{\lg\sigma\lg n}{\lg\lg n})$ time, where m is the length of the queried pattern p-string. The index takes $O(n\lg\sigma)$ bits of space.*

Proof. The index consists of representations of $\mathrm{F}_{\mathcal{T}}$, $\mathrm{L}_{\mathcal{T}}$ and $\mathrm{LCP}^{\infty}_{\mathcal{T}}$. We represent $\mathrm{L}_{\mathcal{T}}$ and $\mathrm{LCP}^{\infty}_{\mathcal{T}}$ by the data structure of Lemma 2.4, and $\mathrm{F}_{\mathcal{T}}$ by the data structure of Lemma 2.3, which leads to the claimed space complexity. Let $P \in \Sigma^*$ with $m = |P|$. We preprocess P with Lemma 3.12. Then the claim regarding count queries follows from Lemma 2.3, Lemma 2.4, and Lemma 3.15. □

We call the index described in Theorem 3.16 the *epBWT index of $\mathcal{T}$*. See Figure 7 for the epBWT index of our running example. Note that the epBWT index of $\{V\}$ is simply the pBWT index of V if $V \in \Sigma^+$, $\mathrm{rank}_{\$}(V, |V|) = 1$ and $V[|V|] = \$$.

3.4. Showcase

We give an exemplary backward search for $P = \mathtt{bCCA}$ with the epBWT index of our running example $\mathcal{T} = \{\mathtt{Bab}, \mathtt{ABBA}, \mathtt{CAbBB}\}$, which is presented in Figure 7.

Since ε is a prefix of every conjugate considered in $\mathcal{T}$, we initially have $[\ell..r] = \mathrm{CR}_{\mathcal{T}}(P[5..]) = \mathrm{CR}_{\mathcal{T}}(\varepsilon) = [1..12]$. For the computation of $\mathrm{CR}_{\mathcal{T}}(\mathtt{A})$, we

observe that the p-symbol $\mathtt{A}$ did not yet appear in $P[5..]$. Computationally, we can infer the appearance of $\mathtt{A}$ in $P[5..]$ from $h = \pi(P[4..]) = \pi(\mathtt{A}) = 1$, $c = |P[5..]|_{\mathrm{p}} = |\varepsilon|_{\mathrm{p}} = 0$ and $h = 1 = 0 + 1 = c + 1$. Algorithm 2 consequently enters Line 7. Next, the proof of Lemma 3.15 tells us that the conjugates corresponding to entries $i \in [\ell..r] = [1..12]$ with $\mathrm{L}_{\mathcal{T}}[i] \geq h = 1$ are the left rotations of conjugates whose prefix of length 1 p-matches $\mathtt{A}$. Thus, we compute $\delta = \mathrm{rnkcnt}_{\mathrm{L}_{\mathcal{T}}}(1, 12, 1, 3) = 9$ in Line 8 of Algorithm 2 to find the length of $\mathrm{CR}_{\mathcal{T}}(\mathtt{A})$. In particular, we find that $\mathrm{CR}_{\mathcal{T}}(\mathtt{A})$ is not empty. It remains to find either the left or right boundary of $\mathrm{CR}_{\mathcal{T}}(\mathtt{A})$ and compute the other from the length. To that end, we compute $v = \mathrm{RNV}_{\mathrm{L}_{\mathcal{T}}}(1, 12, 0) = 1$, $x = \mathrm{select}_1(\mathrm{L}_{\mathcal{T}}, \mathrm{rank}_1(\mathrm{L}_{\mathcal{T}}, 12)) = 12$ and $[\ell''..r''] = \mathrm{MI}_{\mathrm{LCP}^{\infty}_{\mathcal{T}}}(12, 1) = [4..12]$, which are the lowest value in $\mathrm{L}_{\mathcal{T}}[1..12]$ larger than c, the highest index at which the value v occurs and the interval whose entries correspond to prev-encoded conjugates that have at least v occurrences of $\mathbb{x}$ in the longest common prefix with the prev-encoded conjugate corresponding to x, respectively. With v, x and $[\ell''..r'']$ we can leverage Lemma 3.7 as shown in the proof of Lemma 3.15 to compute the number y of entries $j \in [1..12] - \{12\}$ with $\mathrm{L}_{\mathcal{T}}[j] \geq 1$ such that $C_{\mathcal{T}}(\mathrm{LF}_{\mathcal{T}}[j]) \preceq_{\omega} C_{\mathcal{T}}(\mathrm{LF}_{\mathcal{T}}[12])$, i.e., $\mathrm{LF}_{\mathcal{T}}[x]$ is the $(y+1)$-th smallest element of $\mathrm{CR}_{\mathcal{T}}(\mathtt{A})$ by choice of x. From Line 14 through Line 15 of Algorithm 2 we obtain $y = \mathrm{rnkcnt}_{\mathrm{L}_{\mathcal{T}}}(\max\{1, 4\}, 12 - 1, 1, 1) + \mathrm{rnkcnt}_{\mathrm{L}_{\mathcal{T}}}(1, 4 - 1, 1, 3) = 2 + 2 = 4$. Then $r' = \mathrm{LF}_{\mathcal{T}}[12] + 9 - 4 - 1 = 8 + 4 = 12$ and $\mathrm{CR}_{\mathcal{T}}(\mathtt{A}) = [\ell'..r'] = [12 - 9 + 1..12] = [4..12]$, which one can confirm by looking at the range of integer values of $\mathrm{F}_{\mathcal{T}}$ in Figure 7.

Next, we compute $\mathrm{CR}_{\mathcal{T}}(P[3..]) = \mathrm{CR}_{\mathcal{T}}(\mathtt{CA})$. We set $[\ell..r] = \mathrm{CR}_{\mathcal{T}}(\mathtt{A}) = [4..12]$, and compute $h = \pi(\mathtt{CA}) = 2$ and $c = |\mathtt{A}|_{\mathrm{p}} = 1$. Consequently, $\mathtt{C}$ did not appear in $P[4..] = \mathtt{A}$ and we have to proceed as in the previous update of the conjugate range. We get $\delta = \mathrm{rnkcnt}_{\mathrm{L}_{\mathcal{T}}}(4, 12, 2, 3) = 4$, $v = \mathrm{RNV}_{\mathrm{L}_{\mathcal{T}}}(4, 12, 1) = 2$, $x = \mathrm{select}_2(\mathrm{L}_{\mathcal{T}}, \mathrm{rank}_2(\mathrm{L}_{\mathcal{T}}, 12)) = 7$, $[\ell''..r''] = \mathrm{MI}_{\mathrm{LCP}^{\infty}_{\mathcal{T}}}(7, 2) = [6..8]$ and $y = \mathrm{rnkcnt}_{\mathrm{L}_{\mathcal{T}}}(\max\{4, 6\}, 7 - 1, 2, 2) + \mathrm{rnkcnt}_{\mathrm{L}_{\mathcal{T}}}(4, 6 - 1, 2, 3) = 1 + 1 = 2$. Then $r' = \mathrm{LF}_{\mathcal{T}}[7] + 4 - 2 - 1 = 11 + 1 = 12$ and $\mathrm{CR}_{\mathcal{T}}(\mathtt{CA}) = [\ell'..r'] = [12 - 4 + 1..12] = [9..12]$, which is again readily confirmed by, e.g., Figure 5 and $\langle \mathtt{CA} \rangle = \mathbb{xx}$ with Lemma 2.5.

Next, we compute $\mathrm{CR}_{\mathcal{T}}(P[2..]) = \mathrm{CR}_{\mathcal{T}}(\mathtt{CCA})$. We set $[\ell..r] = \mathrm{CR}_{\mathcal{T}}(\mathtt{CA}) = [9..12]$, and compute $h = \pi(\mathtt{CCA}) = 1$ and $c = |\mathtt{CA}|_{\mathrm{p}} = 2$. This time, we have $h \leq c$, i.e., $\mathtt{C}$ already appeared in $\mathtt{CA}$ and we enter Line 2 of Algorithm 2. Then the proof of Lemma 3.15 tells us that the conjugates corresponding to entries $i \in [\ell..r] = [9..12]$ with $\mathrm{L}_{\mathcal{T}}[i] = h = 1$ are the left rotations of conjugates whose prefix of length 3 p-matches $\mathtt{CCA}$. Then

$\delta = \mathrm{rnkcnt}_{\mathrm{L}_\mathcal{T}}(9, 12, 1, 1) = 3$ is the length of $\mathrm{CR}_\mathcal{T}(\mathtt{CCA})$. Moreover, by the non-crossing property coined in Lemma 3.8, the LF-mapping of the largest element x in $\mathrm{CR}_\mathcal{T}(\mathtt{CA})$ such that $\mathrm{L}_\mathcal{T}[x] = h$ is already the right boundary of $\mathrm{CR}_\mathcal{T}(\mathtt{CCA})$, i.e., $r' = \mathrm{LF}_\mathcal{T}[\mathrm{select}_1(\mathrm{L}_\mathcal{T}, \mathrm{rank}_1(\mathrm{L}_\mathcal{T}, 12))] = \mathrm{LF}_\mathcal{T}[12] = 8$. Then $\mathrm{CR}_\mathcal{T}(\mathtt{CCA}) = [8 - 3 + 1..8] = [6..8]$.

Last, we compute $\mathrm{CR}_\mathcal{T}(P[1..]) = \mathrm{CR}_\mathcal{T}(\mathtt{bCCA})$. We set $[\ell..r] = \mathrm{CR}_\mathcal{T}(\mathtt{CCA}) = [6..8]$, and compute $h = \pi(\mathtt{bCCA}) = b$ and $c = |\mathtt{CCA}|_\mathrm{p} = 2$. Here, $h < c$ since we stipulated that s-symbols are smaller than any integer. We proceed similarly to the previous case leveraging the non-crossing property of entries of $\mathrm{L}_\mathcal{T}[\ell..r]$ evaluating to $\mathtt{b}$. As $\delta = \mathrm{rnkcnt}_{\mathrm{L}_\mathcal{T}}(6, 8, b, b) = 1$ indicates that $\mathrm{CR}_\mathcal{T}(\mathtt{bCCA})$ is non-empty, we may compute the right boundary $r' = \mathrm{LF}_\mathcal{T}[\mathrm{select}_b(\mathrm{L}_\mathcal{T}, \mathrm{rank}_b(\mathrm{L}_\mathcal{T}, 8))] = \mathrm{LF}_\mathcal{T}[8] = 3$ of $\mathrm{CR}_\mathcal{T}(\mathtt{bCCA})$. Then $\mathrm{CR}_\mathcal{T}(\mathtt{bCCA}) = [3 - 1 + 1..3] = [3..3]$. Thus, the count query for P returns 1, namely $P = \mathtt{bCCA} \approx \mathtt{bBBC} = \mathrm{Rot}(T_3, 2)^\omega[..4]$.

4. Constructing the epBWT in Compact Space

Our final goal is an iterative construction of the epBWT index. To this end, let $\mathcal{R} \subset \Sigma^+$ be a set of p-strings and $S \in \Sigma^+$ a p-string we want to add to $\mathcal{R}$, where we assume that S is at least as long as the longest p-string in $\mathcal{R}$. First, we give techniques to extend the epBWT index of $\mathcal{R}$ to that of $\mathcal{S} = \mathcal{R} \cup \{S\}$. We subsequently generalize this extension such that we can construct the epBWT index of $\mathcal{T}$ by iteratively extending the epBWT index of $\{T_1, ..., T_{k-1}\}$ to that of $\{T_1, ..., T_k\}$ for each $k \in [1..d]$ in ascending order, where $\mathrm{F}_\emptyset = \mathrm{L}_\emptyset = \mathrm{LCP}^\infty_\emptyset = \varepsilon$. In this section, we assume $|T_k| = n_k \leq n_{k+1} = |T_{ki+1}|$ for each $k \in [1..d-1]$ to obtain the claimed time bounds.

4.1. Extending an epBWT Index

Idea. Consider two decks of cards, each with a different back color: red (for $\mathcal{R}$) and slate (for $\{S\}$). The first deck has $|\mathrm{CA}_\mathcal{R}|$ cards with *red*-colored backs and whose card ranks are $C_\mathcal{R}(1)$ through $C_\mathcal{R}(|\mathrm{CA}_\mathcal{R}|)$. The second deck has $\left|\mathrm{CA}_{\{S\}}\right|$ cards with *slate-colored* backs and whose card ranks are $C_{\{S\}}(1)$ through $C_{\{S\}}(\left|\mathrm{CA}_{\{S\}}\right|)$. So the red cards and the slate cards correspond one-to-one to the conjugates considered in $\mathcal{R}$ and $\{S\}$, respectively. We stack the cards of each deck such that the stack of red cards and the stack of slate cards represent the conjugate arrays of $\mathcal{R}$ and $\{S\}$, respectively. We then label the cards with the corresponding values of F, L and LCP^∞, e.g.,

the topmost card of the stack of red cards is labeled with $\mathrm{F}_{\mathcal{R}}[1]$, $\mathrm{L}_{\mathcal{R}}[1]$ and $\mathrm{LCP}^{\infty}_{\mathcal{R}}[1]$. Then the stack of red cards and the stack of slate cards represent the epBWT index of $\mathcal{R}$ and $\{S\}$, respectively.

Now, in order to obtain a representation of the conjugate array of $\mathcal{S}$, we have to correctly place the previously labeled slate cards into the stack of red cards representing the epBWT index of $\mathcal{R}$. To obtain a representation of the epBWT index of $\mathcal{S}$, we also need to update the LCP^{∞} label of any slate card that has a red card directly on *top* of it, and we need to update the LCP^{∞} labels of red cards if the card is directly *underneath* a slate card.

Approach. Per assumption, we already have the epBWT index of $\mathcal{R}$. Thus, our tasks are fourfold.

(i) For each conjugate S' of S, compute the number of conjugates considered in $\mathcal{R}$ that are not larger according to ω-preorder.

(ii) For each conjugate S' of S, compute the lcp^{∞}-value with the largest conjugate considered in $\mathcal{R}$ that is not larger than S' according to ω-preorder.

(iii) For each conjugate S' of S, compute the lcp^{∞}-value with the smallest conjugate considered in $\mathcal{R}$ that is larger than S' according to ω-preorder.

(iv) Compute the epBWT index of $\{S\}$.

Then (i) and (iv) allow us to infer the conjugate array of $\mathcal{S}$. In addition, (iv) yields the entries of $\mathrm{F}_{\mathcal{S}}$ and $\mathrm{L}_{\mathcal{S}}$ corresponding to conjugates of S as well as $\mathrm{LCP}^{\infty}_{\mathcal{S}}[i]$ if both $\mathrm{CA}_{\mathcal{S}}[i]$ and $\mathrm{CA}_{\mathcal{S}}[i-1]$ correspond to conjugates of S, where $i \in [2..\,|\mathrm{CA}_{\mathcal{S}}|]$. Moreover, (ii) yields $\mathrm{LCP}^{\infty}_{\mathcal{S}}[i]$ if $\mathrm{CA}_{\mathcal{S}}[i]$ and $\mathrm{CA}_{\mathcal{S}}[i-1]$ correspond to a conjugate considered in $\{S\}$ and $\mathcal{R}$, respectively, for $i \in [2..\,|\mathrm{CA}_{\mathcal{S}}|]$. Finally, (iii) yields $\mathrm{LCP}^{\infty}_{\mathcal{S}}[i]$ if $\mathrm{CA}_{\mathcal{S}}[i]$ and $\mathrm{CA}_{\mathcal{S}}[i-1]$ correspond to a conjugate considered in $\mathcal{R}$ and $\{S\}$, respectively, for $i \in [2..\,|\mathrm{CA}_{\mathcal{S}}|]$. The remaining entries of $\mathrm{F}_{\mathcal{S}}$, $\mathrm{L}_{\mathcal{S}}$ and $\mathrm{LCP}^{\infty}_{\mathcal{S}}$ are inferred from the epBWT index of $\mathcal{R}$. Below, we will define the data structures used to store the helper values presented in (i), (ii) and (iii) and rigorously prove that they suffice to compute the epBWT index of $\mathcal{S}$ if those of $\mathcal{R}$ and $\{S\}$ are given. Subsequently, in Section 4.2, we will delve into the computation of both the epBWT index of $\{S\}$ and the helper values.

Algorithm 3: Extending $\mathrm{F}_{\mathcal{R}}$ and $\mathrm{L}_{\mathcal{R}}$ to $\mathrm{F}_{\mathcal{S}}$ and $\mathrm{L}_{\mathcal{S}}$, respectively. Here, $\mathcal{R}, \subset \Sigma^+$, $S \in \Sigma^+$, $\mathcal{S} = \mathcal{R} \cup \{S\}$ and $\lambda = |S|$.

```
1 Function extend1(P_S, F_R, L_R, F_{S}, L_{S}):
2     for i ← 1 to λ do
3         pos ← select_1(P_S, i);
4         insert_{F_R}(pos, F_{S}[i]);
5         insert_{L_R}(pos, L_{S}[i]);
6     return F_R, L_R;
```

Helper Values. Let $\rho = |\mathrm{L}_{\mathcal{R}}|$ and $\lambda = |S| = \max\{|V| \mid V \in \mathcal{S}\}$ be, respectively, the length of the current epBWT and the longest string we want to index, which is S. Let $\mathrm{CNT}_{\mathcal{S}}$ denote a string of length λ over $[1..\rho]$ such that

$$\mathrm{CNT}_{\mathcal{S}}[i] = \left|\{j \in [1..\rho] \mid C_{\mathcal{R}}(j) \preceq_{\omega} C_{\{S\}}(\mathrm{CA}_{\{S\}}[i])\}\right| \text{ for each } i \in [1..\lambda],$$

i.e., $\mathrm{CNT}_{\mathcal{S}}$ stores the helper values described in (i). Note that $\mathrm{CNT}_{\mathcal{S}}$ needs $O((\rho+\lambda)\lg(\rho+\lambda))$ bits of space. In view of a construction in compact space, we want to represent $\mathrm{CNT}_{\mathcal{S}}$ in $O((\rho+\lambda)\lg\sigma)$ bits of space. To that end, let $\mathrm{P}_{\mathcal{S}}$ denote the bit string of length $\rho+\lambda$ satisfying $\mathrm{rank}_1(\mathrm{P}_{\mathcal{S}}, \rho+\lambda) = \lambda$ and $\mathrm{CNT}_{\mathcal{S}}[i] = \mathrm{rank}_0(\mathrm{P}_{\mathcal{S}}, \mathrm{select}_1(\mathrm{P}_{\mathcal{S}}, i))$ for each $i \in [1..\lambda]$. Then $\mathrm{P}_{\mathcal{S}}$ takes $O(\rho+\lambda)$ bits of space and allows us access to $\mathrm{CNT}_{\mathcal{S}}[i]$ in $O(\frac{\lg(\rho+\lambda)}{\lg\lg(\rho+\lambda)})$ time for each $i \in [1..\lambda+\rho]$ if represented by the data structure of Lemma 2.3. For our running example, we have $\mathrm{P}_{\{T_1\}} = \texttt{111}$, $\mathrm{P}_{\{T_1,T_2\}} = \texttt{0001111}$ and $\mathrm{P}_{\mathcal{T}} = \texttt{001010011001}$, where S is defined as T_i for $\mathcal{R} = \{T_1, \ldots, T_{i-1}\}$. We can construct $\mathrm{P}_{\mathcal{S}}$ by zeroing $\mathrm{P}_{\mathcal{R}}$ and calling $\mathrm{insert}_{\mathrm{P}_{\mathcal{R}}}(\mathrm{select}_0(\mathrm{P}_{\mathcal{R}}, \mathrm{CNT}_{\mathcal{S}}[i])+1, 1)$ for each $i \in [1..\lambda]$. Thus, we only need to compute each entry of $\mathrm{CNT}_{\mathcal{S}}$ once and do not need to store the entry explicitly. The following result is immediate.

Lemma 4.1. *Let $\mathcal{R} \subset \Sigma^+$, $S \in \Sigma^+$, $\mathcal{S} = \mathcal{R} \cup \{S\}$ and $\lambda = |S|$. Given $\mathrm{P}_{\mathcal{S}}$, $\mathrm{F}_{\mathcal{R}}$, $\mathrm{L}_{\mathcal{R}}$, $\mathrm{F}_{\{S\}}$ and $\mathrm{L}_{\{S\}}$, Algorithm 3 correctly computes $\mathrm{F}_{\mathcal{S}}$ and $\mathrm{L}_{\mathcal{S}}$.*

Consequently, the extension of both $\mathrm{F}_{\mathcal{R}}$ and $\mathrm{L}_{\mathcal{R}}$ boils down to the computation of $\mathrm{P}_{\mathcal{S}}$ and the epBWT index of $\{S\}$. We tackle the extension of $\mathrm{LCP}^{\infty}_{\mathcal{R}}$ to $\mathrm{LCP}^{\infty}_{\mathcal{S}}$ similarly. Let $\mathrm{TOP}_{\mathcal{S}}$ and $\mathrm{UDN}_{\mathcal{S}}$ denote strings of length λ

i	$\mathrm{F}_{\mathcal{S}}$	$\mathrm{L}_{\mathcal{S}}$	$\mathrm{LCP}^{\infty}_{\mathcal{S}}$	P_{S}
1	a	1	0	0
2	b	a	0	0
3	b	2	1	1
4	b	3	2	0
5	1	b	0	0
6	2	1	1	1
7	3	3	2	0
8	1	2	1	1
9	1	b	1	1
10	1	2	2	0
11	1	2	2	0
12	1	b	2	0
13	3	3	1	0
14	2	1	2	1
15	2	1	2	0
16	2	1	2	0
17	3	1	2	0

(a) The epBWT index of $\mathcal{S} = \mathcal{T} \cup \{S\}$.

i	$\mathrm{F}_{\{S\}}[i]$	$\mathrm{L}_{\{S\}}[i]$	$\mathrm{LCP}^{\infty}_{\{S\}}[i]$
1	b	2	0
2	2	1	0
3	1	2	1
4	1	b	1
5	2	1	1

i	$\mathrm{CNT}_{\mathcal{S}}[i]$	$\mathrm{TOP}_{\mathcal{S}}[i]$	$\mathrm{UDN}_{\mathcal{S}}[i]$
1	2	1	2
2	4	1	2
3	5	1	1
4	5	1	2
5	9	2	2

(b) Auxiliary strings for the extension.

Figure 8: The extension of the epBWT index of $\mathcal{R} = \mathcal{T}$ by $S = \mathtt{BBCCb}$.

over $[-1..\sigma_{\mathrm{p}}]$ such that

$$\mathrm{TOP}_{\mathcal{S}}[i] = \begin{cases} \mathrm{lcp}^{\infty}(C_{\mathcal{R}}(\mathrm{CA}_{\mathcal{R}}[j]), C_{\{S\}}(\mathrm{CA}_{\{S\}}[i])) & \text{if } j \geq 1, \\ -1 & \text{otherwise, and} \end{cases}$$

$$\mathrm{UDN}_{\mathcal{S}}[i] = \begin{cases} \mathrm{lcp}^{\infty}(C_{\mathcal{R}}(\mathrm{CA}_{\mathcal{R}}[j+1]), C_{\{S\}}(\mathrm{CA}_{\{S\}}[i])) & \text{if } j+1 \leq \rho, \\ -1 & \text{otherwise,} \end{cases}$$

for each $i \in [1..\lambda]$ with $j = \mathrm{CNT}_{\mathcal{S}}[i]$, i.e., $\mathrm{TOP}_{\mathcal{S}}$ and $\mathrm{UDN}_{\mathcal{S}}$ store the helper values presented in (ii) and (iii), respectively. For an example, see Figure 8.

Lemma 4.2. *Let $\mathcal{R} \subset \Sigma^+$, $S \in \Sigma^+$, and $\mathcal{S} = \mathcal{R} \cup \{S\}$. Then Algorithm 4 correctly computes the epBWT index of $\mathcal{S}$.*

Proof. Let $\lambda = |S|$ and $\rho = |\mathrm{L}_{\mathcal{R}}|$. The input of Algorithm 4 consists of $\mathrm{P}_{\mathcal{S}}$, $\mathrm{TOP}_{\mathcal{S}}$, $\mathrm{UDN}_{\mathcal{S}}$ and the epBWT indexes of $\mathcal{R}$ and $\{S\}$. The statement

Algorithm 4: Extending the epBWT index of $\mathcal{R}$ to that of $\mathcal{S}$. Here, $\mathcal{R} \subset \Sigma^+$, $S \in \Sigma^+$, $\mathcal{S} = \mathcal{R} \cup \{S\}$ and $\lambda = |S|$.

Function `extend2`($\mathrm{P}_\mathcal{S}$, $\mathrm{TOP}_\mathcal{S}$, $\mathrm{UDN}_\mathcal{S}$, $\mathrm{F}_\mathcal{R}$, $\mathrm{L}_\mathcal{R}$, $\mathrm{LCP}^\infty_\mathcal{R}$, $\mathrm{F}_{\{S\}}$, $\mathrm{L}_{\{S\}}$, $\mathrm{LCP}^\infty_{\{S\}}$)**:**
  $\mathrm{F}_\mathcal{R}, \mathrm{L}_\mathcal{R} \leftarrow$ `extend1`$(\mathrm{P}_\mathcal{S}, \mathrm{F}_\mathcal{R}, \mathrm{L}_\mathcal{R}, \mathrm{F}_{\{S\}}, \mathrm{L}_{\{S\}})$; `// Alg. 3`
  for $i \leftarrow 1$ **to** λ **do**
    $pos \leftarrow \mathrm{select}_1(\mathrm{P}_\mathcal{S}, i)$;
    if $\mathrm{TOP}_\mathcal{S}[i] \geq 0$ *and* $\mathrm{P}_\mathcal{S}[pos - 1] = 0$ **then**
      $\mathrm{insert}_{\mathrm{LCP}^\infty_\mathcal{R}}(pos, \mathrm{TOP}_\mathcal{S}[i])$;
    else $\mathrm{insert}_{\mathrm{LCP}^\infty_\mathcal{R}}(pos, \mathrm{LCP}^\infty_{\{S\}}[i])$;
    if $\mathrm{UDN}_\mathcal{S}[i] \geq 0$ *and* $\mathrm{P}_\mathcal{S}[pos + 1] = 0$ **then**
      $\mathrm{LCP}^\infty_\mathcal{R}[pos + 1] \leftarrow \mathrm{UDN}_\mathcal{S}[i]$;
  return $\mathrm{F}_\mathcal{R}, \mathrm{L}_\mathcal{R}, \mathrm{LCP}^\infty_\mathcal{R}$;

regarding $\mathrm{F}_\mathcal{S}$ and $\mathrm{L}_\mathcal{S}$ follows from Lemma 4.1. By definition, initially

$$\mathrm{LCP}^\infty_\mathcal{R}[..\mathrm{rank}_0(\mathrm{P}_\mathcal{S}, \mathrm{select}_1(\mathrm{P}_\mathcal{S}, 1)) - 1] = \mathrm{LCP}^\infty_\mathcal{S}[..\mathrm{select}_1(\mathrm{P}_\mathcal{S}, 1) - 1] \text{ and}$$
$$\mathrm{LCP}^\infty_\mathcal{R}[\mathrm{rank}_0(\mathrm{P}_\mathcal{S}, \mathrm{select}_1(\mathrm{P}_\mathcal{S}, \lambda)) + 2..] = \mathrm{LCP}^\infty_\mathcal{S}[\mathrm{select}_1(\mathrm{P}_\mathcal{S}, \lambda) + 2..].$$

Moreover, for each $i \in [1..\lambda - 1]$, initially

$$\mathrm{LCP}^\infty_\mathcal{R}[\mathrm{rank}_0(\mathrm{P}_\mathcal{S}, \mathrm{select}_1(\mathrm{P}_\mathcal{S}, i)) + 2..\mathrm{rank}_0(\mathrm{P}_\mathcal{S}, \mathrm{select}_1(\mathrm{P}_\mathcal{S}, i + 1)) - 1]$$
$$= \mathrm{LCP}^\infty_\mathcal{S}[\mathrm{select}_1(\mathrm{P}_\mathcal{S}, i) + 2..\mathrm{select}_1(\mathrm{P}_\mathcal{S}, i + 1) - 1].$$

Now, assume $\mathrm{LCP}^\infty_\mathcal{R}$ was updated such that

$$\mathrm{LCP}^\infty_\mathcal{R}[..\mathrm{select}_1(\mathrm{P}_\mathcal{S}, i) - 1] = \mathrm{LCP}^\infty_\mathcal{S}[..\mathrm{select}_1(\mathrm{P}_\mathcal{S}, i) - 1],$$
$$\mathrm{LCP}^\infty_\mathcal{R}[\mathrm{rank}_0(\mathrm{P}_\mathcal{S}, \mathrm{select}_1(\mathrm{P}_\mathcal{S}, \lambda)) + 1 + i..] = \mathrm{LCP}^\infty_\mathcal{S}[\mathrm{select}_1(\mathrm{P}_\mathcal{S}, \lambda) + 2..],$$

and $\mathrm{LCP}^\infty_\mathcal{R}[\mathrm{rank}_0(\mathrm{P}_\mathcal{S}, \mathrm{select}_1(\mathrm{P}_\mathcal{S}, j)) + 1 + i..\mathrm{rank}_0(\mathrm{P}_\mathcal{S}, \mathrm{select}_1(\mathrm{P}_\mathcal{S}, j + 1)) - 2 + i] = \mathrm{LCP}^\infty_\mathcal{S}[\mathrm{select}_1(\mathrm{P}_\mathcal{S}, i) + 2..\mathrm{select}_1(\mathrm{P}_\mathcal{S}, i + 1) - 1]$, for some fixed $i \in [1..\lambda]$ and each $j \in [i..\lambda - 1]$.

- If $\mathrm{TOP}_\mathcal{S}[i] \geq 0$ and $\mathrm{P}_\mathcal{S}[\mathrm{select}_1(\mathrm{P}_\mathcal{S}, i) - 1] = 0$, then $\mathrm{CA}^{-1}_\mathcal{S}[\mathrm{select}_1(\mathrm{P}_\mathcal{S}, i) - 1] \in [1..\rho]$ and thus $\mathrm{LCP}^\infty_\mathcal{S}[\mathrm{select}_1(\mathrm{P}_\mathcal{S}, i)] = \mathrm{TOP}_\mathcal{S}[i]$.
- If $\mathrm{TOP}_\mathcal{S}[i] \geq 0$ and $\mathrm{P}_\mathcal{S}[\mathrm{select}_1(\mathrm{P}_\mathcal{S}, i) - 1] = 1$, then $\mathrm{CA}^{-1}_\mathcal{S}[\mathrm{select}_1(\mathrm{P}_\mathcal{S}, i) - 1] \in [\rho + 1..\rho + \lambda]$, i.e., $\mathrm{LCP}^\infty_\mathcal{S}[\mathrm{select}_1(\mathrm{P}_\mathcal{S}, i)] = \mathrm{LCP}^\infty_{\{S\}}[i]$.

- If $\mathrm{TOP}_{\mathcal{S}}[i] < 0$ and $i = 1$, then $\mathrm{LCP}^{\infty}_{\mathcal{S}}[\mathrm{select}_1(\mathrm{P}_{\mathcal{S}}, i)] = 0 = \mathrm{LCP}^{\infty}_{\{S\}}[i]$.
- If $\mathrm{TOP}_{\mathcal{S}}[i] < 0$ and $i > 1$, then $\mathrm{LCP}^{\infty}_{\mathcal{S}}[\mathrm{select}_1(\mathrm{P}_{\mathcal{S}}, i)] = \mathrm{LCP}^{\infty}_{\{S\}}[i]$.

These cases are treated from Line 5 through Line 7 and we updated $\mathrm{LCP}^{\infty}_{\mathcal{R}}$ such that $\mathrm{LCP}^{\infty}_{\mathcal{R}}[..\mathrm{select}_1(\mathrm{P}_{\mathcal{S}}, i)] = \mathrm{LCP}^{\infty}_{\mathcal{S}}[..\mathrm{select}_1(\mathrm{P}_{\mathcal{S}}, i)]$.

If $\mathrm{select}_1(\mathrm{P}_{\mathcal{S}}, i) < \rho + \lambda$ and $\mathrm{P}_{\mathcal{S}}[\mathrm{select}_1(\mathrm{P}_{\mathcal{S}}, i) + 1] = 1$, or $\mathrm{UDN}_{\mathcal{S}}[i] < 0$, then by assumption we already have

$$\begin{aligned}\mathrm{LCP}^{\infty}_{\mathcal{R}}[..\mathrm{select}_1(\mathrm{P}_{\mathcal{S}}, i+1) - 1] &= \mathrm{LCP}^{\infty}_{\mathcal{R}}[..\mathrm{select}_1(\mathrm{P}_{\mathcal{S}}, i)] \\ &= \mathrm{LCP}^{\infty}_{\mathcal{S}}[..\mathrm{select}_1(\mathrm{P}_{\mathcal{S}}, i)] \\ &= \mathrm{LCP}^{\infty}_{\mathcal{S}}[..\mathrm{select}_1(\mathrm{P}_{\mathcal{S}}, i+1) - 1]\end{aligned}$$

for $i+1 \leq \lambda$ and $\mathrm{LCP}^{\infty}_{\mathcal{R}} = \mathrm{LCP}^{\infty}_{\mathcal{S}}$ for $i = \lambda$. Otherwise, $\mathrm{CA}^{-1}_{\mathcal{S}}[\mathrm{select}_1(\mathrm{P}_{\mathcal{S}}, i) + 1] \in [1..\rho]$ and consequently $\mathrm{LCP}^{\infty}_{\mathcal{S}}[\mathrm{select}_1(\mathrm{P}_{\mathcal{S}}, i)] = \mathrm{UDN}_{\mathcal{S}}[i]$. By assumption, $\mathrm{LCP}^{\infty}_{\mathcal{R}}[..\mathrm{select}_1(\mathrm{P}_{\mathcal{S}}, i+1)-1] = \mathrm{LCP}^{\infty}_{\mathcal{S}}[..\mathrm{select}_1(\mathrm{P}_{\mathcal{S}}, i+1)-1]$ for $i+1 \leq \lambda$ and $\mathrm{LCP}^{\infty}_{\mathcal{R}} = \mathrm{LCP}^{\infty}_{\mathcal{S}}$ for $i = \lambda$. We treat this part from Line 8 through Line 9. We readily confirm that after the loop our assumption holds for $i+1$ if $i + 1 \leq \lambda$. Since i was chosen arbitrarily, the statement follows. □

4.2. Building the Remaining Parts

We conclude that the extension of the epBWT of $\mathcal{R}$ to that of $\mathcal{S}$ can be reduced to computing $\mathrm{TOP}_{\mathcal{S}}$, $\mathrm{UDN}_{\mathcal{S}}$, $\mathrm{P}_{\mathcal{S}}$ and the epBWT index of $\{S\}$. For their computation, we assume $\mathcal{R}, \mathcal{S} \subset (\Sigma - \{\$\})^+$ in this subsection. This is without loss of generality because we can always add a new s-symbol to Σ taking over the role of \$, which is stipulated to be the smallest s-symbol.

Let $z = \max\{|V| \mid V \in \mathcal{S}\}$, $S' = S^{\omega}[..4z]\$$ and denote by Y_S a bit string of length $4z+1$ such that $\mathrm{Y}_S[i] = 1$ if and only if $\mathrm{CA}_{\{S'\}}[i] \in [2..\lambda+1]$ for each $i \in [1..4z+1]$. Here, Y_S induces a one-to-one mapping of the conjugates of S to the conjugates of S' such that $\mathrm{Rot}(S, i)^{\omega}[..3z] = S'[i+1..3z+i+1]$ for each $i \in [1..\lambda]$, which allows us to draw conclusions about the ω-preorder and the $[\![..]\!]$-encoding of the conjugates of S from the ω-preorder and the $[\![..]\!]$-encoding of the conjugates of S' by application of Lemma 3.4 and Corollary 3.5.

The convenience of working with S' instead of S will become clear shortly. For what follows, recall that we assumed S to be at least as long as the longest string in $\mathcal{R}$, i.e., $\lambda = z$ and $S' = S^{\omega}[..4\lambda]\$ = S^4\$$.

Lemma 4.3. *Algorithm 5 correctly computes the epBWT index of* $\{S\}$ *for* $S \in (\Sigma - \{\$\})^+$.

Algorithm 5: Computing the epBWT of the singleton set $\{S\}$.
Here, $S \in (\Sigma - \{\$\})^+$, $S' = S^4\$$ and $\lambda = |S|$.

Function `singletonepBWT`(Y_S, $\mathrm{F}_{\{S'\}}$, $\mathrm{L}_{\{S'\}}$, $\mathrm{LCP}^{\infty}_{\{S'\}}$)**:**
  $\mathrm{F}_{\{S\}}, \mathrm{L}_{\{S\}} \leftarrow$ zeroed string of length λ over $\Sigma_{\mathrm{s}} \cup [1..\sigma_{\mathrm{p}}]$;
  $\mathrm{LCP}^{\infty}_{\{S\}} \leftarrow$ zeroed string of length λ over $[0..\sigma_{\mathrm{p}}]$;
  $prev \leftarrow 0$;
  for $i \leftarrow 1$ **to** λ **do**
    $curr \leftarrow \mathrm{select}_1(\mathrm{Y}_S, i)$;
    $\mathrm{F}_{\{S\}}[i] \leftarrow \mathrm{F}_{\{S'\}}[curr]$;
    $\mathrm{L}_{\{S\}}[i] \leftarrow \mathrm{L}_{\{S'\}}[curr]$;
    $\mathrm{LCP}^{\infty}_{\{S\}}[i] \leftarrow \mathrm{RNV}_{\mathrm{LCP}^{\infty}_{\{S'\}}}(prev + 1, curr, -1)$;
    $prev \leftarrow curr$;
  return $\mathrm{F}_{\{S\}}, \mathrm{L}_{\{S\}}, \mathrm{LCP}^{\infty}_{\{S\}}$;

Proof. Let $S' = S^4\$$, and $\lambda = |S|$. Algorithm 5 takes as its input Y_S and the epBWT index of $\{S'\}$. We have

$$\langle C_{\{S\}}(\mathrm{CA}_{\{S\}}[i])^3 \rangle = \langle C_{\{S'\}}(\mathrm{CA}_{\{S'\}}[\mathrm{select}_1(\mathrm{Y}_S, i)])[..3\lambda] \rangle$$

for each $i \in [1..\lambda]$ by construction. Thus, $\mathrm{select}_1(\mathrm{Y}_S, i) \in \mathrm{CR}_{\{S'\}}(C_{\{S\}}(i)^3)$ for each $i \in [1..\lambda]$, which in turn implies

$$\begin{aligned} [\![C_{\{S'\}}(\mathrm{CA}_{\{S'\}}[\mathrm{select}_1(\mathrm{Y}_S, i)])[..\lambda]]\!] &= [\![C_{\{S'\}}(\mathrm{CA}_{\{S'\}}[\mathrm{select}_1(\mathrm{Y}_S, i)])]\!][..\lambda] \\ &= [\![C_{\{S\}}(\mathrm{CA}_{\{S\}}[i])]\!] \end{aligned}$$

by choice of S' and the backward search. Also, by choice of S',

$$[\![C_{\{S'\}}(\mathrm{CA}_{\{S'\}}[\mathrm{select}_1(\mathrm{Y}_S, i)])]\!][\lambda] = [\![C_{\{S'\}}(\mathrm{CA}_{\{S'\}}[\mathrm{select}_1(\mathrm{Y}_S, i)])]\!][4\lambda + 1]$$

for $i \in [1..\lambda]$. Then the statement follows from Lemma 3.4 and Lemma 3.14. □

By choice of S', the epBWT index of $\{S'\}$ is the pBWT index of S', and we can leverage Lemma 2.8 for its construction. Moreover, during construction of the pBWT index of S' the string Y_S can be built similarly to the bit string indicating sample positions of the suffix array in the proof of Lemma 2.8. The epBWT index of $\{S'\}$ and Y_S now come in handy if we consider the set of strings $\mathcal{S}' = \mathcal{R} \cup \{S'\}$.

Lemma 4.4. *Let* $\mathcal{R} \subset (\Sigma - \{\$\})^+$ *and* $S \in (\Sigma - \{\$\})^+$. *Define* $\mathcal{S} = \mathcal{R} \cup \{S\}$, $S' = S^4\$$, $\mathcal{S}' = \mathcal{R} \cup \{S'\}$, $\lambda = |S| = \max\{|V| \mid V \in \mathcal{S}\}$ *and* $\rho = |\mathrm{L}_{\mathcal{R}}|$. *Then* $\mathrm{CNT}_{\mathcal{S}}[i] = \mathrm{CNT}_{\mathcal{S}'}[j] + |\mathrm{CR}_{\mathcal{R}}(S^3)|$,

$$\mathrm{TOP}_{\mathcal{S}}[i] = \begin{cases} \mathrm{TOP}_{\mathcal{S}'}[j] & \text{if } |\mathrm{CR}_{\mathcal{R}}(S^3)| = 0, \\ \mathrm{UDN}_{\mathcal{S}'}[j] & \text{otherwise, and} \end{cases}$$

$$\mathrm{UDN}_{\mathcal{S}}[i] = \begin{cases} \mathrm{UDN}_{\mathcal{S}'}[j] & \text{if } |\mathrm{CR}_{\mathcal{R}}(S^3)| = 0, \\ \mathrm{LCP}^{\infty}_{\mathcal{R}}[\mathrm{CNT}_{\mathcal{S}}[i] + 1] & \text{if } \mathrm{CNT}_{\mathcal{S}}[i] < \rho, \\ -1 & \text{otherwise,} \end{cases}$$

for each $i \in [1..\lambda]$ *with* $j = \mathrm{select}_1(\mathrm{Y}_S, i)$.

Proof. By choice of S', $C_{\mathcal{R}}(x) \neq_{\omega} C_{\{S'\}}(y)$ for each $x \in [1..\rho]$ and $y \in [1..4\lambda + 1]$. In particular, $C_{\{S'\}}(y) \prec_{\omega} C_{\mathcal{R}}(x)$ for each $x \in [1..\rho]$ and $y \in [1..4\lambda + 1]$ with $\langle C_{\{S'\}}(y)[..3\lambda]\rangle = \langle C_{\mathcal{R}}(x)^{\omega}[..3\lambda]\rangle$. Since

$$\langle C_{\{S\}}(\mathrm{CA}_{\{S\}}[i])^3\rangle = \langle C_{\{S'\}}(\mathrm{CA}_{\{S'\}}[\mathrm{select}_1(\mathrm{Y}_S, i)])[..3\lambda]\rangle \text{ for each } i \in [1..\lambda]$$

by construction, the statement regarding $\mathrm{CNT}_{\mathcal{S}}[i]$ then follows from the definition of a conjugate range, Lemma 3.4 and Corollary 3.5.

Fix some $i \in [1..\lambda]$ with $j = \mathrm{select}_1(\mathrm{Y}_S, i)$. Then the statement regarding $\mathrm{TOP}_{\mathcal{S}}[i]$ and $\mathrm{UDN}_{\mathcal{S}}[i]$ is immediate from their definition if $|\mathrm{CR}_{\mathcal{R}}(S^3)| = 0$. Assume $|\mathrm{CR}_{\mathcal{R}}(S^3)| \neq 0$. Then

$$\mathrm{UDN}_{\mathcal{S}'}[j] = \mathrm{lcp}^{\infty}(C_{\mathcal{R}}(\mathrm{CA}_{\mathcal{R}}[\mathrm{CNT}_{\mathcal{S}'}[i] + k]), C_{\{S'\}}(\mathrm{CA}_{\{S'\}}[i]))$$

for each $k \in [1..\,|\mathrm{CR}_{\mathcal{R}}(S^3)|]$ by the definition of a conjugate range and the proof of Lemma 3.14. In particular, $\mathrm{UDN}_{\mathcal{S}'}[j] = \mathrm{TOP}_{\mathcal{S}}[i]$. For the statement regarding $\mathrm{UDN}_{\mathcal{S}}[i]$, we simply apply its definition to the fact that $\langle C_{\{S\}}(\mathrm{CA}_{\{S\}}[i])^3\rangle = \langle C_{\mathcal{R}}(\mathrm{CA}_{\mathcal{R}}[\mathrm{CNT}_{\mathcal{S}}[i]])^{\omega}[..3\lambda]\rangle$. □

We give an example in Figure 9. It remains to show how to compute $\mathrm{TOP}_{\mathcal{S}'}$, $\mathrm{UDN}_{\mathcal{S}'}$ and the individual values of $\mathrm{CNT}_{\mathcal{S}'}$ for our extension. By choice of S', $\mathrm{CNT}_{\mathcal{S}'}[1] = \mathrm{CNT}_{\mathcal{S}'}[\mathrm{CA}^{-1}_{\{S'\}}[4\lambda + 1]] = 0$, $\mathrm{TOP}_{\mathcal{S}'}[1] = -1$, $\mathrm{UDN}_{\mathcal{S}'}[1] = 0$ and $[\![S']\!]$ is primitive, with the latter implying that $\mathrm{LF}_{\{S'\}}$ decomposes into a single cycle. In what follows, we will show how to compute $\mathrm{CNT}_{\mathcal{S}'}[j]$, $\mathrm{TOP}_{\mathcal{S}'}[j]$ and $\mathrm{UDN}_{\mathcal{S}'}[j]$ from $\mathrm{CNT}_{\mathcal{S}'}[i]$, $\mathrm{TOP}_{\mathcal{S}'}[i]$ and $\mathrm{UDN}_{\mathcal{S}'}[i]$ for each $i \in [1..4\lambda + 1]$ with $j = \mathrm{LF}_{\{S'\}}[i]$. Then application of this result 4λ

i	$\mathrm{CA}_{\{S'\}}[i]$	$\mathrm{F}_{\{S'\}}[i]$	$\mathrm{L}_{\{S'\}}[i]$	$\mathrm{LCP}^{\infty}_{\{S'\}}[i]$	$\mathrm{Y}_{S}[i]$	$\mathrm{CNT}_{\mathcal{S}'}[i]$	$\mathrm{TOP}_{\mathcal{S}'}[i]$	$\mathrm{UDN}_{\mathcal{S}'}[i]$
1	21	\$	b	0	0	0	-1	0
2	20	b	2	0	0	1	0	0
3	15	b	2	0	0	2	1	2
4	10	b	2	2	0	2	1	2
5	5	b	2	2	1	2	1	2
6	19	2	1	0	0	4	1	1
7	14	2	1	1	0	4	1	2
8	9	2	1	2	0	4	1	2
9	4	2	1	2	1	4	1	2
10	18	1	2	1	0	5	1	1
11	13	1	2	1	0	5	1	1
12	8	1	2	2	0	5	1	1
13	3	1	2	2	1	5	1	1
14	16	1	b	1	0	5	1	2
15	11	1	b	2	0	5	1	2
16	6	1	b	2	1	5	1	2
17	1	1	\$	2	0	5	1	2
18	17	2	1	1	0	9	2	2
19	12	2	1	2	0	9	2	2
20	7	2	1	2	0	9	2	2
21	2	2	1	2	1	9	2	2

Figure 9: Auxiliary data structures for extending the epBWT index of our running example $\mathcal{T} = \mathcal{R}$ by $S = \texttt{BBCCb}$. Here, $\mathcal{S} = \mathcal{R} \cup \{S\}$, $S' = S^4\$$, $\mathcal{S}' = \mathcal{R} \cup \{S'\}$, and $\left|\mathrm{CR}_{\mathcal{R}}(S^3)\right| = 0$. By Lemma 4.3 and Lemma 4.4, $\mathrm{F}_{\{S\}}$, $\mathrm{L}_{\{S\}}$, $\mathrm{CNT}_{\mathcal{S}}$, $\mathrm{TOP}_{\mathcal{S}}$ and $\mathrm{UDN}_{\mathcal{S}}$ is the subsequence of entries i in $\mathrm{F}_{\{S'\}}$, $\mathrm{L}_{\{S'\}}$, $\mathrm{CNT}_{\mathcal{S}'}$, $\mathrm{TOP}_{\mathcal{S}'}$ and $\mathrm{UDN}_{\mathcal{S}'}$, respectively, such that $\mathrm{Y}_S[i] = 1$. Moreover, we use RNV queries on $\mathrm{LCP}^{\infty}_{\{S'\}}$ as presented in Algorithm 5 to obtain $\mathrm{LCP}^{\infty}_{\{S\}} = \texttt{00111}$. The extracted data structures are presented in Figure 8.

times starting from $i = 1$ will yield all values of $\mathrm{CNT}_{\mathcal{S}'}$, $\mathrm{TOP}_{\mathcal{S}'}$ and $\mathrm{UDN}_{\mathcal{S}'}$ since $\mathrm{LF}_{\{S'\}}$ decomposes into a single cycle.

The following result is an adaptation of the techniques presented by Iseri et al. [15, Section 3.1].

Lemma 4.5. *Let* $\mathcal{R} \subset (\Sigma - \{\$\})^+$, $S \in (\Sigma - \{\$\})^+$, $S' = S^4\$$, $\mathcal{S}' = \mathcal{R} \cup \{S'\}$ *and* $\lambda = |S| \geq \max\{|V| \mid V \in \mathcal{R}\}$. *Algorithm 6 correctly computes* $\mathrm{CNT}_{\mathcal{S}'}[\mathrm{LF}_{\{S'\}}[i]]$ *for each given* $i \in [1..4\lambda + 1]$.

Proof. Let $\rho = |\mathrm{L}_{\mathcal{R}}|$. The input of Algorithm 6 for some $i \in [1..4\lambda + 1]$ consists of $\pi(C_{\{S'\}}(\mathrm{CA}_{\{S'\}}[\mathrm{LF}_{\{S'\}}[i]]))$, $\mathrm{CNT}_{\mathcal{S}'}[i]$, $\mathrm{TOP}_{\mathcal{S}'}[i]$, $\mathrm{UDN}_{\mathcal{S}'}[i]$ and the epBWT index of $\mathcal{R}$.

Fix some $i \in [1..4\lambda + 1]$ and let $j = \mathrm{LF}_{\{S'\}}[i]$ and $h = \mathrm{F}_{\{S'\}}[j] = \pi(C_{\{S'\}}(\mathrm{CA}_{\{S'\}}[j]))$. We assume $\mathcal{R} \neq \emptyset$ throughout the proof since the statement is immediate if $\mathcal{R} = \emptyset$.

Case 1. Assume $h \leq 0$, i.e., $h \in \Sigma_{\mathrm{s}}$. Then Lemma 3.7 implies $\mathrm{CNT}_{\mathcal{S}'}[j] = \mathrm{rnkcnt}_{\mathrm{L}_{\mathcal{R}}}(1, \rho, \$, h) - \mathrm{rnkcnt}_{\mathrm{L}_{\mathcal{R}}}(i + 1, \rho, h, h)$. The case is treated from Line 3 through Line 4.

Case 2. Assume $h \geq 1$ and $\mathrm{rnkcnt}_{\mathrm{L}_{\mathcal{R}}}(1, \rho, 1, \sigma_{\mathrm{p}}) = 0$, i.e., $C_{\{S'\}}(\mathrm{CA}_{\{S'\}}[j])[1]$ is a p-symbol and $C_{\mathcal{R}}(\alpha) \in \Sigma_{\mathrm{s}}^+$ for each $\alpha \in [1..\rho]$. Then $\mathrm{CNT}_{\mathcal{S}'}[j] = \rho$ since $\mathfrak{w}$ is larger than any s-symbol. The case is treated in Line 5.

Case 3. Assume $h \geq 1$ and $\mathrm{rnkcnt}_{\mathrm{L}_{\mathcal{R}}}(1, \rho, 1, \sigma_{\mathrm{p}}) \geq 1$, i.e., $C_{\{S'\}}(\mathrm{CA}_{\{S'\}}[j])[1]$ is a p-symbol and $C_{\mathcal{R}}(\alpha) \notin \Sigma_{\mathrm{s}}^+$ for some $\alpha \in [1..\rho]$. The case is treated from Line 6. We will follow the proof of Lemma 12 by Iseri et al. [15] in less detail and adapt it where necessary.

For notational convenience, let $V_\alpha = C_{\mathcal{R}}(\mathrm{CA}_{\mathcal{R}}[\mathrm{LF}_{\mathcal{R}}[\alpha]])$ for each $\alpha \in [1..\rho]$ and $U = C_{\{S'\}}(\mathrm{CA}_{\{S'\}}[j])$. We define

$$\gamma = \max\{\mathrm{lcp}(\langle V_\alpha^\omega[..4\lambda + 1]\rangle, \langle U\rangle) \mid \alpha \in [1..\rho]\}, \text{ and}$$

$$t_e = \begin{cases} \mathrm{select}_{\mathfrak{w}}(\langle \mathrm{Rot}(U, 1)\rangle, e) & \text{if } e \in [1..\min\{h, |U|_{\mathrm{p}}\}], \text{ or} \\ 4\lambda + 1 & \text{if } e \in [\min\{h, |U|_{\mathrm{p}}\} + 1..\sigma_{\mathrm{p}} + 1]. \end{cases}$$

In other words, the left rotation of V_α, $\mathrm{Rot}(V_\alpha, 1)$, is the α-smallest conjugate in $\mathcal{R}$, U and $\mathrm{Rot}(U, 1)$ are the j-smallest and the i-smallest conjugate in $\{S'\}$, respectively, and γ is the maximum length of the prefixes of the prev-encodings of U with all V_α's. We will not compute γ directly, but $e \in [1..\min\{h, |U|_{\mathrm{p}}\}]$ such that $\gamma \in [t_e..t_{e+1}]$, and then infer $\mathrm{CNT}_{\mathcal{S}'}[j]$. The following is immediate by definitions.

Algorithm 6: Computing $\mathrm{CNT}_{\mathcal{S}'}[j]$ from $\mathrm{CNT}_{\mathcal{S}'}[i]$, $\mathrm{TOP}_{\mathcal{S}'}[i]$ and $\mathrm{UDN}_{\mathcal{S}'}[i]$ for $i \in [1..4\lambda+1]$ with $j = \mathrm{LF}_{\{S'\}}[i]$. Here, $\mathcal{R}, \mathcal{S} \subset (\Sigma - \{\$\})^+$, $\mathcal{S} = \mathcal{R} \cup \{S\}$, $S' = S^4\$$, $\mathcal{S}' = \mathcal{R} \cup \{S'\}$, $\rho = |\mathrm{L}_{\mathcal{R}}|$, $\lambda = |S| = \max\{|V| \mid V \in \mathcal{S}\}$ and $h = \pi(C_{\{S'\}}(\mathrm{CA}_{\{S'\}}[j])) = \mathrm{F}_{\{S'\}}[j]$.

Function `nextcnt`(h, $\mathrm{CNT}_{\mathcal{S}'}[i]$, $\mathrm{TOP}_{\mathcal{S}'}[i]$, $\mathrm{UDN}_{\mathcal{S}'}[i]$, $\mathrm{F}_{\mathcal{R}}$, $\mathrm{L}_{\mathcal{R}}$, $\mathrm{LCP}^{\infty}_{\mathcal{R}}$)**:**

- **if** $\rho = 0$ **then return** 0 ; // $\mathcal{R} = \emptyset$
- **if** $h < 1$ **then** // Case 1 of Lemma 4.5
  - **return** $\mathrm{rnkcnt}_{\mathrm{L}_{\mathcal{R}}}(1, \rho, \$, h) - \mathrm{rnkcnt}_{\mathrm{L}_{\mathcal{R}}}(i+1, \rho, h, h)$;
- **if** $\mathrm{rnkcnt}_{\mathrm{L}_{\mathcal{R}}}(1, \rho, 1, \sigma_{\mathrm{p}}) = 0$ **then return** ρ ; // Case 2
- $k \leftarrow \mathrm{CNT}_{\mathcal{S}'}[i]$;
- **for** $e \leftarrow \min\{h, \max\{\mathrm{TOP}_{\mathcal{S}'}[i], \mathrm{UDN}_{\mathcal{S}'}[i]\}\}$ **to** 0 **do** // Case 3
  - $[\ell..r] \leftarrow \mathrm{MI}_{\mathrm{LCP}^{\infty}_{\mathcal{R}}}(\max\{1, k\}, e)$;
  - **if** $e = h$ **then** // Case 3.1
    - **if** $x \leftarrow \mathrm{FPQ}_e(\mathrm{L}_{\mathcal{R}}, k) \in [\ell..r]$ **then return** $\mathrm{LF}_{\mathcal{R}}[x]$;
    - **if** $x \leftarrow \mathrm{FNQ}_e(\mathrm{L}_{\mathcal{R}}, k) \in [\ell..r]$ **then return** $\mathrm{LF}_{\mathcal{R}}[x] - 1$;
    - **if** $x \leftarrow \mathrm{FNQ}_{\geq e+1}(\mathrm{L}_{\mathcal{R}}, \ell) \in [\ell..r]$ **then** // check for (H1)
      - **return** $\min \mathrm{MI}_{\mathrm{LCP}^{\infty}_{\mathcal{R}}}(\mathrm{LF}_{\mathcal{R}}[x], e+1) - 1$;
  - **else** // Case 3.2
    - **if** $x \leftarrow \mathrm{FPQ}_{\geq e+1}(\mathrm{L}_{\mathcal{R}}, k) \in [\ell..r]$ **then** // Case 3.2.1
      - $[\ell'..r'] \leftarrow \mathrm{MI}_{\mathrm{LCP}^{\infty}_{\mathcal{R}}}(x, e+1)$;
      - **if** $x' \leftarrow \mathrm{FPQ}_{\geq e+2}(\mathrm{L}_{\mathcal{R}}, r') \in [\ell'..r']$ **then**
        - **return** $\max \mathrm{MI}_{\mathrm{LCP}^{\infty}_{\mathcal{R}}}(\mathrm{LF}_{\mathcal{R}}[x'], e+2)$;
      - **else return** $\mathrm{LF}_{\mathcal{R}}[x]$;
    - **if** $x \leftarrow \mathrm{FNQ}_{\geq e+1}(\mathrm{L}_{\mathcal{R}}, k) \in [\ell..r]$ **then** // Case 3.2.2
      - $[\ell'..r'] \leftarrow \mathrm{MI}_{\mathrm{LCP}^{\infty}_{\mathcal{R}}}(x, e+1)$;
      - **if** $x' \leftarrow \mathrm{FNQ}_{e+1}(\mathrm{L}_{\mathcal{R}}, \ell') \in [\ell'..r']$ **then**
        - **return** $\mathrm{LF}_{\mathcal{R}}[x'] - 1$;
      - **else return** $\min \mathrm{MI}_{\mathrm{LCP}^{\infty}_{\mathcal{R}}}(\mathrm{LF}_{\mathcal{R}}[x], e+2) - 1$;
    - **if** $x \leftarrow \mathrm{FPQ}_e(\mathrm{L}_{\mathcal{R}}, r) \in [\ell..r]$ **then** // check for (H1)
      - **return** $\mathrm{LF}_{\mathcal{R}}[x]$;

Fact. *Let* $e \in [1..\min\{h, |U|_{\mathrm{p}}\}]$ *and* $\alpha, \beta \in [1..\rho]$. *If* $\mathrm{lcp}(\langle U\rangle, \langle V_\alpha^\omega[..4\lambda{+}1]\rangle) = t_e$, *then either*

(H1) $\mathrm{lcp}^\infty(\mathrm{Rot}(U,1), \mathrm{Rot}(V_\alpha,1)) \geq e \wedge (h > \pi(V_\alpha) = e \vee \pi(V_\alpha) > h = e)$, *or*

(H2) $\mathrm{lcp}(\langle \mathrm{Rot}(U,1)\rangle, \langle \mathrm{Rot}(V_\alpha,1)^\omega[..4\lambda+1]\rangle) = t_e - 1 \wedge \min\{h, \pi(V_\alpha)\} \geq e$.

Moreover, if (H1) *holds for* V_α, $\mathrm{lcp}(\langle U\rangle, \langle V_\beta^\omega[..4\lambda+1]\rangle) = t_e$ *and* (H2) *holds for* V_β, *then* $\neg(V_\alpha \prec_\omega V_\beta \prec_\omega U)$ *and* $\neg(U \prec_\omega V_\beta \prec_\omega V_\alpha)$.

Fix some $e \in [1..\min\{h, \max\{\mathrm{TOP}_{\mathcal{S}'}[i], \mathrm{UDN}_{\mathcal{S}'}[i]\}\}]$ and let $[\ell..r] \subseteq [1..\rho]$ such that $\mathrm{lcp}^\infty(\mathrm{Rot}(U,1), \mathrm{Rot}(V_\alpha,1)) \geq e$ if and only if $\alpha \in [\ell..r]$, i.e., $[\ell..r]$ is the non-empty interval computed in Line 8. We will check if either $\gamma \in [t_e + 1..t_{e+1}]$ or if there exists some $\alpha \in [1..\rho]$ such that $\mathrm{lcp}(\langle U\rangle, \langle V_\alpha^\omega[..4\lambda + 1]\rangle) = t_e$ and V_α satisfies (H1), and show to compute $\mathrm{CNT}_{\mathcal{S}'}[j]$ if either is the case. Note that if neither is the case, then for each $\alpha \in [1..\rho]$ either $\mathrm{lcp}^\infty(\mathrm{Rot}(U,1), \mathrm{Rot}(V_\alpha,1)) \leq e-1$ or $\pi(V_\alpha) \leq e-1$ by the fact.

Case 3.1. Assume $e = h$. This case is treated from Line 9 through Line 13. For each $\alpha \in [1..\rho]$, $\mathrm{lcp}(\langle U\rangle, \langle V_\alpha^\omega[..4\lambda+1]\rangle) > t_e$ if and only if $\mathrm{L}_{\mathcal{R}}[\alpha] = h$ and $\alpha \in [\ell..r]$. By Lemma 3.7, $\mathrm{Rot}(U,1) \prec_\omega \mathrm{Rot}(V_\alpha,1)$ if and only if $U \prec_\omega V_\alpha$ for each $\alpha \in [\ell..r]$ with $\mathrm{L}_{\mathcal{R}}[\alpha] = h$. Thus, it suffices to compute either the largest $\alpha \in [\ell..\mathrm{CNT}_{\mathcal{S}'}[i]]$ or smallest $\alpha \in [\mathrm{CNT}_{\mathcal{S}'}[i] + 1..r]$ satisfying $\mathrm{L}_{\mathcal{R}}[\alpha] = h$, resulting in $\mathrm{CNT}_{\mathcal{S}'}[j] = \mathrm{LF}_{\mathcal{R}}[\alpha]$ or $\mathrm{CNT}_{\mathcal{S}'}[j] = \mathrm{LF}_{\mathcal{R}}[\alpha] - 1$, respectively, if they exist. The computation is done in Line 10 and Line 11, respectively, and if neither returns a value, we conclude that $\gamma \in [1..t_e]$.

From Line 12 through Line 13 we check if there exists some $\alpha \in [1..\rho]$ such that $\mathrm{lcp}(\langle U\rangle, \langle V_\alpha^\omega[..4\lambda+1]\rangle) = t_e$ and V_α satisfies (H1). But this condition is equivalent to $\alpha \in [\ell..r]$ and $\mathrm{L}_{\mathcal{R}}[\alpha] > h$. If some $\alpha \in [\ell..r]$ with $\mathrm{L}_{\mathcal{R}}[\alpha] > h$ exists, then $\mathrm{CNT}_{\mathcal{S}'}[j] = \min \mathrm{MI}_{\mathrm{LCP}^\infty_{\mathcal{R}}}(\mathrm{LF}_{\mathcal{R}}[\alpha], e+1) - 1$ by the fact.

Case 3.2. Assume that either $\mathrm{lcp}^\infty(\mathrm{Rot}(U,1), \mathrm{Rot}(V_\alpha,1)) \leq e$ or that $\pi(V_\alpha) \leq e$ for each $\alpha \in [1..\rho]$. Note that this assumption is fulfilled by definition for $e = \max\{\mathrm{TOP}_{\mathcal{S}'}[i], \mathrm{UDN}_{\mathcal{S}'}[i]\} < h$. Also, $\gamma \in [1..t_{e+1}]$.

We first check if $\gamma \in [t_e+1..t_{e+1}]$. For each $\alpha \in [1..\rho]$, $\mathrm{lcp}^\infty(U, V_\alpha) \geq e+1$ if and only if $\alpha \in [\ell..r]$ and $\mathrm{L}_{\mathcal{R}}[\alpha] \geq e+1$ by Lemma 3.7. Consequently, $\gamma \in [t_e + 1..t_{e+1}]$ if some $\alpha \in [\ell..r]$ with $\mathrm{L}_{\mathcal{R}}[\alpha] \geq e+1$ exists.

Case 3.2.1. Assume we have an $x \in [\ell..\mathrm{CNT}_{\mathcal{S}'}[i]]$ such that $\mathrm{L}_{\mathcal{R}}[x] \geq e+1$ and $\alpha \leq x$ for any $\alpha \in [\ell..\mathrm{CNT}_{\mathcal{S}'}[i]]$ with $\mathrm{L}_{\mathcal{R}}[\alpha] \geq e+1$. We treat this case from Line 15 through Line 19. Let $t' = \mathrm{select}_{\aleph}(\langle \mathrm{Rot}(V_x,1)\rangle, e+1)$ and let $[\ell'..r'] \subseteq [1..\rho]$ such that $\mathrm{lcp}(\langle \mathrm{Rot}(V_\alpha,1)\rangle, \langle \mathrm{Rot}(V_\beta,1)\rangle) \geq t'$ if and only if

$\alpha, \beta \in [\ell'..r']$, i.e., $[\ell'..r']$ is the interval computed in Line 16. Then $V_x \prec_\omega U$ and $r' \geq \mathrm{CNT}_{\mathcal{S}'}[i]$. By assumption and Lemma 3.7, $\mathrm{CNT}_{\mathcal{S}'}[j] \in [\ell'..x]$. Then $\mathrm{lcp}(\langle C_{\mathcal{R}}(\mathrm{CA}_{\mathcal{R}}[\mathrm{CNT}_{\mathcal{S}'}[j]])^\omega[..4\lambda+1]\rangle, \langle U\rangle) \leq t'$, which in turn implies that $\langle C_{\mathcal{R}}(\mathrm{CA}_{\mathcal{R}}[\mathrm{CNT}_{\mathcal{S}'}[j]])^\omega[..4\lambda+1]\rangle$ is prefixed by $W = \langle V_x^\omega[..t']\rangle$. If there exists some $\alpha \in [\ell'..r']$ such that $\mathrm{L}_{\mathcal{R}}[\alpha] \geq e+2$, then $\langle V_\alpha\rangle$ is prefixed by $W \cdot \mathbb{x}$ and consequently $\mathrm{CNT}_{\mathcal{S}'}[j] = \max \mathrm{MI}_{\mathrm{LCP}^\infty_{\mathcal{R}}}(\mathrm{LF}_{\mathcal{R}}[\alpha], e+2)$, with the condition being checked in Line 17 and computation of $\mathrm{CNT}_{\mathcal{S}'}[j]$ taking place in Line 18. Otherwise $\langle C_{\mathcal{R}}(\mathrm{CA}_{\mathcal{R}}[\mathrm{CNT}_{\mathcal{S}'}[j]])^\omega[..4\lambda+1]\rangle$ is prefixed by $W \cdot t'$ and consequently $\mathrm{CNT}_{\mathcal{S}'}[j] = \mathrm{LF}_{\mathcal{R}}[x]$, which is computed in Line 19.

Case 3.2.2. Assume we have an $x \in [\mathrm{CNT}_{\mathcal{S}'}[i]+1..r]$ such that $\mathrm{L}_{\mathcal{R}}[x] \geq e+1$ and $\alpha \geq x$ for any $\alpha \in [\ell..r]$ with $\mathrm{L}_{\mathcal{R}}[\alpha] \geq e+1$. We treat this case from Line 20 through Line 24. Let $t' = \mathrm{select}_{\mathbb{x}}(\langle \mathrm{Rot}(V_x, 1)\rangle, e+1)$ and let $[\ell'..r'] \subseteq [1..\rho]$ such that $\mathrm{lcp}(\langle \mathrm{Rot}(V_\alpha, 1)\rangle, \langle \mathrm{Rot}(V_\beta, 1)\rangle) \geq t'$ if and only if $\alpha, \beta \in [\ell'..r']$, i.e., $[\ell'..r']$ is the interval computed in Line 21. Then $U \prec_\omega V_x$ and $\ell' \leq \mathrm{CNT}_{\mathcal{S}'}[j]+1$. By assumption and Lemma 3.7, $\mathrm{CNT}_{\mathcal{S}'}[j]+1 \in [x..r']$. Then $\mathrm{lcp}(\langle C_{\mathcal{R}}(\mathrm{CA}_{\mathcal{R}}[\mathrm{CNT}_{\mathcal{S}'}[j]+1])^\omega[..4\lambda+1]\rangle, \langle U\rangle) \leq t'$, which in turn implies that $\langle C_{\mathcal{R}}(\mathrm{CA}_{\mathcal{R}}[\mathrm{CNT}_{\mathcal{S}'}[j]+1])^\omega[..4\lambda+1]\rangle$ is prefixed by $W = \langle V_x^\omega[..t']\rangle$. If $\alpha \in [\ell'..r']$ such that $\mathrm{L}_{\mathcal{R}}[\alpha] = e+1$, then $\langle V_\alpha\rangle$ is prefixed by $W \cdot t'$. Then $\mathrm{CNT}_{\mathcal{S}'}[j]+1 = \mathrm{LF}_{\mathcal{R}}[x']$ if $x' \in [\ell'..r']$ minimal such that $\mathrm{L}_{\mathcal{R}}[x'] = e+1$ by Lemma 3.7, with the condition checked in Line 22 and the value of $\mathrm{CNT}_{\mathcal{S}'}[j]$ computed in Line 23. If on the other hand $\mathrm{L}_{\mathcal{R}}[\alpha] \neq e+1$ for each $\alpha \in [\ell'..r']$, then $\langle C_{\mathcal{R}}(\mathrm{CA}_{\mathcal{R}}[\mathrm{CNT}_{\mathcal{S}'}[j]+1])^\omega[..4\lambda+1]\rangle$ is prefixed by $W \cdot \mathbb{x}$ and consequently $\mathrm{CNT}_{\mathcal{S}'}[j]+1 = \min \mathrm{MI}_{\mathrm{LCP}^\infty_{\mathcal{R}}}(\mathrm{LF}_{\mathcal{R}}[x], e+2)$, computed in Line 24. This concludes Case 3.2.2.

If $\mathrm{L}_{\mathcal{R}}[\alpha] \leq e$ for each $\alpha \in [\ell..r]$, then $\gamma \in [1..t_e]$, which is the case if we reach Line 25. It remains to check if there exists some $\alpha \in [1..\rho]$ such that $\mathrm{lcp}(\langle U\rangle, \langle V_\alpha^\omega[..4\lambda+1]\rangle) = t_e$ and V_α satisfies (H1). But this condition is equivalent to $\alpha \in [\ell..r]$ and $\mathrm{L}_{\mathcal{R}}[\alpha] = e$. Then $\mathrm{CNT}_{\mathcal{S}'}[j] = \mathrm{LF}_{\mathcal{R}}[\alpha]$ for the largest $\alpha \in [\ell..r]$ with $\mathrm{L}_{\mathcal{R}}[\alpha] = e$ by Lemma 3.7 if such an α exists. This concludes Case 3.

Note that a value is guaranteed to be returned before the for-loop from Line 7 terminates since $\mathcal{R} \neq \emptyset$, with correctness of the returned value following from Case 3. □

For the computation of the entries of $\mathrm{TOP}_{\mathcal{S}'}$ and $\mathrm{UDN}_{\mathcal{S}'}$, we leverage Lemma 2.7 to arrive at the following.

Corollary 4.6. *Let $U, V \in \Sigma^+$ and $e = \mathrm{lcp}^\infty(\mathrm{Rot}(U, 1), \mathrm{Rot}(V, 1))$. Then, given e, $\pi(U)$ and $\pi(V)$, Algorithm 7 correctly computes $\mathrm{lcp}^\infty(U, V)$.*

Algorithm 7: Computing $\mathrm{lcp}^{\infty}(U,V)$ for $U,V \in \Sigma^{+}$ with $e = \mathrm{lcp}^{\infty}(\mathrm{Rot}(U,1),\mathrm{Rot}(V,1))$ and $\mathrm{Rot}(U,1) \prec_{\omega} \mathrm{Rot}(V,1)$.

```
1 Function rotlcp(e, π(U), π(V)):
2     if π(U) = π(V) ≤ e then return e ;
3     if min{π(U), π(V)} ≤ 0 then return 0 ;
4     if π(U) ≤ min{e, π(V)} then return π(U) ;
5     if π(V) ≤ min{e, π(U)} then return π(V) ;
6     return e + 1;
```

Lemma 4.7. *Let $\mathcal{R} \subset (\Sigma - \{\$\})^{+}$, $S \in (\Sigma - \{\$\})^{+}$, $S' = S^4\$$, $\mathcal{S}' = \mathcal{R} \cup \{S'\}$ and $\lambda = |S| \geq \max\{|V| \mid V \in \mathcal{R}\}$. Algorithm 8 correctly computes $\mathrm{TOP}_{\mathcal{S}'}[\mathrm{LF}_{\{S'\}}[i]]$ and $\mathrm{UDN}_{\mathcal{S}'}[\mathrm{LF}_{\{S'\}}[i]]$ for each $i \in [1..4\lambda+1]$.*

Proof. The input of Algorithm 8 consists of $\pi(C_{\{S'\}}(\mathrm{LF}_{\{S'\}}[i]))$, $\mathrm{CNT}_{\mathcal{S}'}[i]$, $\mathrm{CNT}_{\mathcal{S}'}[\mathrm{LF}_{\{S'\}}[i]]$, $\mathrm{TOP}_{\mathcal{S}'}[i]$, $\mathrm{UDN}_{\mathcal{S}'}[i]$ for some $i \in [1..4\lambda+1]$ and the epBWT index of $\mathcal{R}$. We show the statement for $\mathrm{TOP}_{\mathcal{S}'}[\mathrm{LF}_{\{S'\}}[i]]$ since the proof for $\mathrm{UDN}_{\mathcal{S}'}[\mathrm{LF}_{\{S'\}}[i]]$ is similar.

Fix some arbitrary $i \in [1..4\lambda+1]$. We catch the border case where $\mathrm{CNT}_{\mathcal{S}}[\mathrm{LF}_{\{S'\}}[i]] = 0$ in Line 2. Thus, assume $\mathrm{CNT}_{\mathcal{S}}[\mathrm{LF}_{\{S'\}}[i]] > 0$. Let $U = C_{\mathcal{R}}(\mathrm{CA}_{\mathcal{R}}[\mathrm{CNT}_{\mathcal{S}}[\mathrm{LF}_{\{S'\}}[i]]])$ and $V = C_{\{S'\}}(\mathrm{CA}_{\{S'\}}[\mathrm{LF}_{\{S'\}}[i]])$. Then we utilize Lemma 3.14 and the FL-mapping of the epBWT index of $\mathcal{R}$ to obtain $\mathrm{lcp}^{\infty}(\mathrm{Rot}(U,1),\mathrm{Rot}(V,1))$ in Line 4 through Line 9, where the different cases are due to the ω-preorder of $\mathrm{Rot}(U,1)$ and $\mathrm{Rot}(V,1)$. Then the first statement follows from Corollary 4.6 with the computation of $\mathrm{TOP}_{\mathcal{S}'}[\mathrm{LF}_{\{S'\}}[i]] = \mathrm{lcp}^{\infty}(U,V)$ in Line 10. □

We give a summarizing result.

Lemma 4.8. *Let $\mathcal{R} \subset (\Sigma - \{\$\})^{+}$, $S \in (\Sigma - \{\$\})^{+}$, $S' = S^4\$$, $\mathcal{S} = \mathcal{R} \cup \{S\}$ and $\lambda = |S| = \max\{|V| \mid V \in \mathcal{S}\}$. Algorithm 9 correctly computes $\mathrm{P}_{\mathcal{S}}$ and the epBWT index of $\mathcal{S}$.*

Proof. Let $\rho = |\mathrm{L}_{\mathcal{R}}|$. The input of Algorithm 9 consists of $\mathrm{P}_{\mathcal{R}}$, $|\mathrm{CR}_{\mathcal{R}}(S^3)|$, Y_S, and the epBWT indexes of $\mathcal{R}$ and $\{S'\}$.

We correctly compute the epBWT index of $\{S\}$ in Line 2 by Lemma 4.3. If $\mathcal{R} = \emptyset$, we simply construct $\mathrm{P}_{\mathcal{S}} = \mathtt{1}^{\lambda}$ and return the epBWT for the singleton set $\{S\}$ in Line 3 through Line 5.

Algorithm 8: Computing $\mathrm{TOP}_{\mathcal{S}'}[j]$ and $\mathrm{UDN}_{\mathcal{S}'}[j]$ from $\mathrm{CNT}_{\mathcal{S}'}[i]$, $\mathrm{CNT}_{\mathcal{S}'}[j]$, $\mathrm{TOP}_{\mathcal{S}'}[i]$ and $\mathrm{UDN}_{\mathcal{S}'}[i]$ for $i \in [1..4\lambda+1]$ with $j = \mathrm{LF}_{\{S'\}}[i]$. Here, $\mathcal{R}, \mathcal{S} \subset (\Sigma - \{\$\})^+$, $\mathcal{S} = \mathcal{R} \cup \{S\}$, $S' = S^4\$$, $\mathcal{S}' = \mathcal{R} \cup \{S'\}$, $\lambda = |S|$, $\rho = |\mathrm{L}_{\mathcal{R}}|$ and $h = \pi(C_{\{S'\}}(j)) = \mathrm{F}_{\{S'\}}[j]$.

Function `nexttopbot`(h, $\mathrm{CNT}_{\mathcal{S}'}[i]$, $\mathrm{CNT}_{\mathcal{S}'}[j]$, $\mathrm{TOP}_{\mathcal{S}'}[i]$, $\mathrm{UDN}_{\mathcal{S}'}[i]$, $\mathrm{F}_{\mathcal{R}}$, $\mathrm{L}_{\mathcal{R}}$, $\mathrm{LCP}^{\infty}_{\mathcal{R}}$)**:**

- **if** $\mathrm{CNT}_{\mathcal{S}'}[j] = 0$ **then** $top \leftarrow -1$; // compute $\mathrm{TOP}_{\mathcal{S}'}[j]$
- **else**
  - $k \leftarrow \mathrm{FL}_{\mathcal{R}}[\mathrm{CNT}_{\mathcal{S}'}[j]]$;
  - **if** $k = \mathrm{CNT}_{\mathcal{S}'}[i] + 1$ **then** $e \leftarrow \mathrm{UDN}_{\mathcal{S}'}[i]$;
  - **else if** $k > \mathrm{CNT}_{\mathcal{S}'}[i] + 1$ **then**
    - $e \leftarrow \min\{\mathrm{UDN}_{\mathcal{S}'}[i], \mathrm{RNV}_{\mathrm{LCP}^{\infty}_{\mathcal{R}}}(\mathrm{CNT}_{\mathcal{S}'}[i] + 2, k, -1)\}$;
  - **else if** $k = \mathrm{CNT}_{\mathcal{S}'}[i]$ **then** $e \leftarrow \mathrm{TOP}_{\mathcal{S}'}[i]$;
  - **else** $e \leftarrow \min\{\mathrm{TOP}_{\mathcal{S}'}[i], \mathrm{RNV}_{\mathrm{LCP}^{\infty}_{\mathcal{R}}}(k+1, \mathrm{CNT}_{\mathcal{S}'}[i], -1)\}$;
  - $top \leftarrow$ `rotlcp`$(e, h, \mathrm{F}_{\mathcal{R}}[\mathrm{CNT}_{\mathcal{S}'}[j]])$; // Alg. 7
- **if** $\mathrm{CNT}_{\mathcal{S}'}[j] = \rho$ **then** $bot \leftarrow -1$; // compute $\mathrm{UDN}_{\mathcal{S}'}[j]$
- **else**
  - $k \leftarrow \mathrm{FL}_{\mathcal{R}}[\mathrm{CNT}_{\mathcal{S}'}[j] + 1]$;
  - **if** $k = \mathrm{CNT}_{\mathcal{S}'}[i] + 1$ **then** $e \leftarrow \mathrm{UDN}_{\mathcal{S}'}[i]$;
  - **else if** $k > \mathrm{CNT}_{\mathcal{S}'}[i] + 1$ **then**
    - $e \leftarrow \min\{\mathrm{UDN}_{\mathcal{S}'}[i], \mathrm{RNV}_{\mathrm{LCP}^{\infty}_{\mathcal{R}}}(\mathrm{CNT}_{\mathcal{S}'}[i] + 2, k, -1)\}$;
  - **else if** $k = \mathrm{CNT}_{\mathcal{S}'}[i]$ **then** $e \leftarrow \mathrm{TOP}_{\mathcal{S}'}[i]$;
  - **else** $e \leftarrow \min\{\mathrm{TOP}_{\mathcal{S}'}[i], \mathrm{RNV}_{\mathrm{LCP}^{\infty}_{\mathcal{R}}}(k+1, \mathrm{CNT}_{\mathcal{S}'}[i], -1)\}$;
  - $bot \leftarrow$ `rotlcp`$(e, h, \mathrm{F}_{\mathcal{R}}[\mathrm{CNT}_{\mathcal{S}'}[j] + 1])$; // Alg. 7
- **return** top, bot;

Now, assume $\mathcal{R} \neq \emptyset$. Then the condition of Line 3 does not evaluate to true. We initialize $\mathrm{TOP}_{\mathcal{S}}$ and $\mathrm{UDN}_{\mathcal{S}}$ and set the initial values of $cnt = \mathrm{CNT}_{\mathcal{S}'}[1] = 0$, $bot = \mathrm{UDN}_{\mathcal{S}'}[1] = 0$, $top = \mathrm{TOP}_{\mathcal{S}'}[1] = -1$ and the current index in Line 6 through Line 10. Correctness of the initial values follows by choice of S'. We proceed to compute $\mathrm{P}_{\mathcal{S}}$, $\mathrm{TOP}_{\mathcal{S}}$ and $\mathrm{UDN}_{\mathcal{S}}$ in Line 11 through Line 24. Correctness follows from Lemma 4.4, Lemma 4.5 and Lemma 4.7. Last, we correctly compute the epBWT index of $\mathcal{S}$ by Lemma 4.2 in Line 25. □

The input of Algorithm 9 is computed as follows. We construct the pBWT index of S' with Lemma 2.8, which is the epBWT index of $\{S'\}$ by choice of S'. During construction of the pBWT index, Y_S is built similarly to the bit string indicating sample positions of the suffix array in the proof of Lemma 2.8 without impacting space or time complexity of the construction algorithm. For the computation of $|\mathrm{CR}_{\mathcal{R}}(S^3)|$, we use a count query for S^3 in $\mathcal{R}$, which is supported by the epBWT index of $\mathcal{R}$ as shown in Theorem 3.16. Since $\mathrm{P}_{\mathcal{R}}$ and the epBWT index of $\mathcal{R}$ are known a priori, we covered all arguments of the function `extend3` of Algorithm 9.

Remark 4.9. For the extension of the epBWT from $\mathcal{R}$ to $\mathcal{S}$ we assumed that $\lambda = |S| = \max\{|V| \mid V \in \mathcal{S}\} = z$ for simplicity and because the results are sufficient for our construction algorithm. However, the extension in case of $\lambda \leq z - 1$ is done similarly with $S' = S^{\omega}[..4z] \cdot \$$, i.e., the epBWT index is extendable in general.

4.3. Constructing the epBWT Index of $\mathcal{T}$

We finally analyze the time and space complexities of the epBWT construction.

Theorem 4.10. *Let $T_1, ..., T_d \in (\Sigma - \{\$\})^+$, $n = |T_1 \cdots T_d|$ and $n_k = |T_k|$ for each $k \in [1..d]$ with $d \in \mathbb{N}_{\geq 1}$. If $n_i \leq n_j$ for each $i, j \in [1..d]$ with $i \leq j$, then the epBWT index of $\mathcal{T} = \{T_1, ..., T_d\}$ can be computed in $O(n \lg \sigma)$ bits of space and $O(n \frac{\lg \sigma \lg n}{\lg \lg n})$ time.*

Proof. Let $n_i \leq n_j$ for each $i, j \in [1..d]$ with $i \leq j$. For notational convenience, let $\mathcal{T}_k = \{T_1, ..., T_k\}$, $T'_k = T_k^4\$$, $\mathcal{T}'_k = \{T_1, ..., T_{k-1}, T'_k\}$ for each $k \in [1..d]$ and $t_{\mathrm{query}} = \frac{\lg \sigma \lg n}{\lg \lg n}$. Then for each $k \in [1..d]$, the pBWT index of T'_k is the epBWT index of $\{T'_k\}$.

Fix some $k \in [1..d]$. We leverage Lemma 2.8 to construct the epBWT index of $\{T'_k\}$ and Y_{T_k} in $O(n_k \lg \sigma)$ bits of space and $O(n_k t_{\mathrm{query}})$ time. Here,

Algorithm 9: Extending the epBWT index of $\mathcal{R}$ to that of $\mathcal{S}$. Here, $\mathcal{R} \subset (\Sigma - \{\$\})^+$, $S \in (\Sigma - \{\$\})^+$, $\mathcal{S} = \mathcal{R} \cup \{S\}$, $S' = S^4\$$, $\mathcal{S}' = \mathcal{R} \cup \{S'\}$, $\rho = |\mathrm{L}_\mathcal{R}|$ and $\lambda = |S| = \max\{|V| \mid V \in \mathcal{S}\}$.

Function `extend3`($\mathrm{F}_\mathcal{R}$, $\mathrm{L}_\mathcal{R}$, $\mathrm{LCP}^\infty_\mathcal{R}$, $\mathrm{P}_\mathcal{R}$, $|\mathrm{CR}_\mathcal{R}(S^3)|$, Y_S, $\mathrm{F}_{\{S'\}}$, $\mathrm{L}_{\{S'\}}$, $\mathrm{LCP}^\infty_{\{S'\}}$)**:**

- $\mathrm{F}_{\{S\}}, \mathrm{L}_{\{S\}}, \mathrm{LCP}^\infty_{\{S\}} \leftarrow$ `singletonepBWT`$(\mathrm{Y}_S, \mathrm{F}_{\{S'\}}, \mathrm{L}_{\{S'\}}, \mathrm{LCP}^\infty_{\{S'\}})$; // Alg. 5
- **if** $\mathrm{P}_\mathcal{R} = \varepsilon$ **then** // $\mathcal{R} = \emptyset$
  - $\mathrm{P}_\mathcal{S} \leftarrow$ non-zeroed bit string of length λ;
  - **return** $\mathrm{F}_{\{S\}}, \mathrm{L}_{\{S\}}, \mathrm{LCP}^\infty_{\{S\}}, \mathrm{P}_\mathcal{S}$;
- **for** $i \leftarrow 1$ **to** $\mathrm{rank}_1(\mathrm{P}_\mathcal{R}, |\mathrm{P}_\mathcal{R}|)$ **do** $\mathrm{P}_\mathcal{R}[\mathrm{select}_1(\mathrm{P}_\mathcal{R}, 1)] \leftarrow 0$;
- $\mathrm{TOP}_\mathcal{S}, \mathrm{UDN}_\mathcal{S} \leftarrow$ zeroed string of length λ over $[-1..\sigma_\mathrm{p}]$;
- $index \leftarrow 1$;
- $cnt, bot \leftarrow 0$; // initialize $\mathrm{CNT}_{\mathcal{S}'}[1] = \mathrm{UDN}_{\mathcal{S}'}[1] = 0$
- $top \leftarrow -1$; // initialize $\mathrm{TOP}_{\mathcal{S}'}[1] = -1$
- **for** $j \leftarrow 1$ **to** 4λ **do** // $\mathrm{CNT}_{\mathcal{S}'}, \mathrm{TOP}_{\mathcal{S}'}, \mathrm{UDN}_{\mathcal{S}'}$ with Alg. 6 and Alg. 8
  - $index \leftarrow \mathrm{LF}_{\{S'\}}[index]$;
  - $nxtcnt \leftarrow$ `nextcnt`$(\mathrm{F}_{\{S'\}}[index], cnt, top, bot, \mathrm{L}_\mathcal{R}, \mathrm{LCP}^\infty_\mathcal{R})$;
  - $top, bot \leftarrow$ `nexttopbot`$(\mathrm{F}_{\{S'\}}[index]), cnt, nxtcnt, top, bot, \mathrm{F}_\mathcal{R}, \mathrm{L}_\mathcal{R}, \mathrm{LCP}^\infty_\mathcal{R})$;
  - **if** $\mathrm{Y}_S[index] = 1$ **then** // $\mathrm{CNT}_\mathcal{S}, \mathrm{TOP}_\mathcal{S}, \mathrm{UDN}_\mathcal{S}$ from Lemma 4.4
    - $\mathrm{insert}_{\mathrm{P}_\mathcal{R}}(\mathrm{select}_0(\mathrm{P}_\mathcal{R}, nxtcnt + |\mathrm{CR}_\mathcal{R}(S^3)|) + 1, 1)$; // $\mathrm{P}_\mathcal{S}$ from extending zeroed $\mathrm{P}_\mathcal{R}$ with $\mathrm{CNT}_\mathcal{S}$ values
    - **if** $|\mathrm{CR}_\mathcal{R}(S^3)| = 0$ **then**
      - $\mathrm{TOP}_\mathcal{S}[\mathrm{rank}_1(\mathrm{Y}_S, index)] \leftarrow top$;
      - $\mathrm{UDN}_\mathcal{S}[\mathrm{rank}_1(\mathrm{Y}_S, index)] \leftarrow bot$;
    - **else**
      - $\mathrm{TOP}_\mathcal{S}[\mathrm{rank}_1(\mathrm{Y}_S, index)] \leftarrow bot$;
      - **if** $nxtcnt = \rho$ **then** $\mathrm{UDN}_\mathcal{S}[\mathrm{rank}_1(\mathrm{Y}_S, index)] \leftarrow -1$;
      - **else** $\mathrm{UDN}_\mathcal{S}[\mathrm{rank}_1(\mathrm{Y}_S, index)] \leftarrow \mathrm{LCP}^\infty_\mathcal{R}[nxtcnt + 1]$;
  - $cnt \leftarrow nxtcnt$;
- $\mathrm{F}_\mathcal{S}, \mathrm{L}_\mathcal{S}, \mathrm{LCP}^\infty_\mathcal{S} \leftarrow$ `extend2`$(\mathrm{P}_\mathcal{S}, \mathrm{TOP}_\mathcal{S}, \mathrm{UDN}_\mathcal{S}, \mathrm{F}_\mathcal{R}, \mathrm{L}_\mathcal{R}, \mathrm{LCP}^\infty_\mathcal{R}, \mathrm{F}_{\{S\}}, \mathrm{L}_{\{S\}}, \mathrm{LCP}^\infty_{\{S\}})$; // Alg. 4
- **return** $\mathrm{F}_\mathcal{S}, \mathrm{L}_\mathcal{S}, \mathrm{LCP}^\infty_\mathcal{S}, \mathrm{P}_\mathcal{R}$;

Y_{T_k} is built similarly to the bit string indicating sample positions of the suffix array in the proof of Lemma 2.8, $\mathrm{L}_{\{T'_k\}}$ and $\mathrm{LCP}^{\infty}_{\{T'_k\}}$ are represented by the data structure of Lemma 2.4 and $\mathrm{F}_{\{T'_k\}}$ and Y_{T_k} are represented by that of Lemma 2.3. By Theorem 3.16, we compute $\mathrm{CR}_{\mathcal{T}_{k-1}}(T_k^3)$ and its length in $O(n_k t_{\mathrm{query}})$ time.

By Lemma 4.3, Algorithm 5 correctly computes the epBWT index of $\{T_k\}$. We represent $\mathrm{L}_{\{T_k\}}$ and $\mathrm{LCP}^{\infty}_{\{T_k\}}$ by the data structure of Lemma 2.4 and $\mathrm{F}_{\{T_k\}}$ by that of Lemma 2.3, which implies these occupy $O(n_k \lg \sigma)$ bits of space in total and their construction takes $O(n_k t_{\mathrm{query}})$ time.

Assume $k = 1$. We initialize $\mathrm{P}_{\{T_1\}} = 1^{n_1}$ and represent it by the data structure of Lemma 2.3, which implies it occupies $O(n_k)$ bits of space and is constructed in $O(n_1 t_{\mathrm{query}})$ time. In summary, we computed $\mathrm{P}_{\{T_1\}}$ and the epBWT index of $\mathcal{T}_1$ in $O(n_1 t_{\mathrm{query}})$ time and $O(n_1 \lg \sigma)$ bits of space, with correctness following by Lemma 4.8.

Assume $k \geq 2$ and that $\mathrm{P}_{\mathcal{T}_{k-1}}$, $\mathrm{F}_{\mathcal{T}_{k-1}}$, $\mathrm{L}_{\mathcal{T}_{k-1}}$ and $\mathrm{LCP}^{\infty}_{\mathcal{T}_{k-1}}$ have already been computed with $\mathrm{P}_{\mathcal{T}_{k-1}}$ and $\mathrm{F}_{\mathcal{T}_{k-1}}$ represented by the data structure of Lemma 2.3, and $\mathrm{L}_{\mathcal{T}_{k-1}}$ and $\mathrm{LCP}^{\infty}_{\mathcal{T}_{k-1}}$ by that of Lemma 2.4. We use Algorithm 9 to extend $\mathrm{P}_{\mathcal{T}_{k-1}}$, $\mathrm{F}_{\mathcal{T}_{k-1}}$, $\mathrm{L}_{\mathcal{T}_{k-1}}$ and $\mathrm{LCP}^{\infty}_{\mathcal{T}_{k-1}}$ to $\mathrm{P}_{\mathcal{T}_k}$, $\mathrm{F}_{\mathcal{T}_k}$, $\mathrm{L}_{\mathcal{T}_k}$ and $\mathrm{LCP}^{\infty}_{\mathcal{T}_k}$, respectively, by insert operations, i.e., the extension is done in $O(n_k \lg \sigma)$ bits of space.

We claim that Algorithm 9 takes $O(n_k t_{\mathrm{query}})$ time, which is readily confirmed with the exception of the function call in Line 13 due to the for-loop from Line 14 of Algorithm 6. Algorithm 6 computes $\mathrm{CNT}_{\mathcal{T}'_k}[\mathrm{LF}_{\{T'_k\}}[i]]$ for each $i \in [1..4n_k + 1]$ by invoking $O(2 + e - e')$ queries and operations, where we assigned $e' = \max\{\mathrm{TOP}_{\mathcal{T}'_k}[\mathrm{LF}_{\{T'_k\}}[i]], \mathrm{UDN}_{\mathcal{T}'_k}[\mathrm{LF}_{\{T'_k\}}[i]]\}$ and $e = \max\{\mathrm{TOP}_{\mathcal{T}'_k}[i], \mathrm{UDN}_{\mathcal{T}'_k}[i]\}$. The value e can be seen as a potential held by the current conjugate $C_{\{T'_k\}}(\mathrm{CA}_{\{T'_k\}}[i])$ of T'_k, which upper bounds the number of queries and operations for a call of Algorithm 6. Since a single call increases the potential for the next conjugate $C_{\{T'_k\}}(\mathrm{CA}_{\{T'_k\}}[\mathrm{LF}_{\{T'_k\}}[i]])$ of T'_k by at most 1 according to Lemma 2.7, the total number of invoked queries and operations by the function call in Line 13 of Algorithm 9 is bounded by $O(n_k)$. Our claim follows.

Since $k \in [1..d]$ was chosen arbitrarily and $n_i \leq n_j$ for each $i, j \in [1..d]$ with $i \leq j$, the construction of the epBWT index of $\mathcal{T}$ takes $O((n+n_d) \lg \sigma) = O(n \lg \sigma)$ bits of space and $O((\sum_{k=1}^{d} n_k) t_{\mathrm{query}}) = O(n t_{\mathrm{query}})$ time, which concludes the proof. □

The following result is then a consequence of Theorem 4.10 and Lemma 2.1

through Lemma 2.4.

Corollary 4.11. *Let $T_1, ..., T_d \in (\Sigma - \{\$\})^+$, $n = |T_1 \cdots T_d|$ and $n_k = |T_k|$ for each $k \in [1..d]$ with $d \in \mathbb{N}_{\geq 1}$. If $n_i \leq n_j$ for each $i, j \in [1..d]$ with $i \leq j$, then the static epBWT index of $\mathcal{T} = \{T_1, ..., T_d\}$ presented in Theorem 3.13 can be computed in $O(n \lg \sigma)$ bits of space and $O(n \frac{\lg \sigma \lg n}{\lg \lg n})$ time.*

Finally, we join together the main results of all sections of this article to affirm the theorem we stated in the very beginning.

Proof of Theorem 1.1. The space complexity of the epBWT index follows from Theorem 3.16 and the claims regarding its construction from Theorem 4.10. We can switch to static data structures by Corollary 4.11 and the space complexity of the static epBWT index follows from Theorem 3.13.

Both variants support count queries in the claimed time complexities by Theorem 3.13 and Theorem 3.16. The time complexity for inserting or deleting a string of the set of strings indexed by the dynamic epBWT is due to Remark 4.9.

If $[\![T_1]\!], ..., [\![T_d]\!]$ are primitive, then the claim regarding restoration of the input text p-strings follows from Corollary 3.10. Otherwise, we augment the index with the auxiliary bit string $\mathrm{D}'_{\mathcal{T}}$ as described at the end of Section 3.2 The bit string $\mathrm{D}'_{\mathcal{T}}$ is represented by the data structure of Lemma 2.3 and constructed alongside the epBWT index of $\mathcal{T}$ without impacting the claimed complexities since $\exp([\![T_k]\!])$ can be determined by a backward search for T_k^3 on the epBWT index of $\{T_k^4\$\}$ for each $k \in [1..d]$. □

Remark 4.12. In case we want to report the locations of occurrences, we can construct $\mathrm{CA}_{\mathcal{T}}$ alongside the epBWT index and store it in $O(n + d \lg n)$ bits of space by means of sampling each $(\lg n)$-th text position [3, Sect. 3.2], where the latter term is due to the $O(d)$ cycles of the LF-mapping corresponding to the $[\![..]\!]$-encoded roots of the input text p-strings, which require at least a sample each.

5. Matching Statistics

As an application, we give a generalization of the matching statistics defined by Chang and Lawler [16]. The *circular matching statistics* between $\mathcal{T} \subset \Sigma^+$ and a pattern p-string $P \in \Sigma^+$ of length m is an array $M[1..m]$, whose entries are triples of a prefix length $M[i].q \in [0..m]$, a left interval

i	1	2	3	4	5	6	7	8	9	10	11	12
$\mathrm{E}_{\mathcal{T}}$	0	0	2	0	1	1	15	3	1	2	15	2

Figure 10: The string $\mathrm{E}_{\mathcal{T}}$ of our running example. Its values are readily confirmed by, e.g., the last column of Figure 5. Here, since $n' = n_3 = 5$, an entry i of $\mathrm{E}_{\mathcal{T}}$ evaluating to 15 indicates ω-equivalence of $C_{\mathcal{T}}(\mathrm{CA}_{\mathcal{T}}[i])$ and $C_{\mathcal{T}}(\mathrm{CA}_{\mathcal{T}}[i-1])$ by Lemma 3.4.

boundary $M[i].\ell \in [1..n]$ and a right interval boundary $M[i].r \in [1..n]$ such that for each $i \in [1..m]$

(i) $M[i].q = \max\{\mathrm{lcp}(\langle C_{\mathcal{T}}(\mathrm{CA}_{\mathcal{T}}[j])^{\omega}[..m]\rangle, \langle P[i..]\rangle) \mid j \in [1..n]\}$,

(ii) $M[i].\ell \leq M[i].r$ and

(iii) $j \in [M[i].\ell..M[i].r] \Leftrightarrow \mathrm{lcp}(\langle C_{\mathcal{T}}(\mathrm{CA}_{\mathcal{T}}[j])^{\omega}[..m]\rangle, \langle P[i..]\rangle) = M[i].q$.

See Figure 11 for an example. We adapt the results of Ohlebusch et al. [46] and compute the circular matching statistics with the backward search of the epBWT index. Their main idea is to compute sufficiently many conjugate ranges of $P[i..j]$ with $i, j \in [1..m]$. We start with $i = j = m$ and decrease i until $\mathrm{CR}_{\mathcal{T}}(P[i..j])$ is empty, and then decrease j until $\mathrm{CR}_{\mathcal{T}}(P[i..j])$ is non-empty, called a *contraction* operation. We repeat until $i = 1$ and either $j = 0$ or $\mathrm{CR}_{\mathcal{T}}(P[1..j])$ is non-empty. For the contraction, we use an auxiliary string that helps us to find the correct range if we end up with a mismatch. Let $n' = \max\{n_k \mid k \in [1..d]\}$ and denote by $\mathrm{E}_{\mathcal{T}}$ a string of length n over $[0..3n']$ such that $\mathrm{E}_{\mathcal{T}}[1] = 0$ and

$$\mathrm{E}_{\mathcal{T}}[i] = \mathrm{lcp}(\langle C_{\mathcal{T}}(\mathrm{CA}_{\mathcal{T}}[i-1])^{\omega}[..3n']\rangle, \langle C_{\mathcal{T}}(\mathrm{CA}_{\mathcal{T}}[i])^{\omega}[..3n']\rangle)$$

for each $i \in [2..n]$. Note that by Lemma 3.4 and for each $i \in [2..n]$, $C_{\mathcal{T}}(\mathrm{CA}_{\mathcal{T}}[i-1]) =_{\omega} C_{\mathcal{T}}(\mathrm{CA}_{\mathcal{T}}[i])$ if and only if $\mathrm{E}_{\mathcal{T}}[i] = 3n'$. See Figure 10 for an example.

Lemma 5.1. *Algorithm 10 correctly computes the circular matching statistics between $\mathcal{T} \subset \Sigma^+$ and a pattern p-string $P \in \Sigma^+$.*

The statement is a consequence of Ohlebusch et al. [46, Section 3], but for completeness we give a detailed proof.

Proof. Algorithm 10 takes a pattern $P \in \Sigma^+$, the static epBWT index of $\mathcal{T}$ presented in Theorem 3.13 and $\mathrm{E}_{\mathcal{T}}$ as input. We address the computation in

Algorithm 10: Computing the circular matching statistics between $\mathcal{T} \subset \Sigma^+$ and a pattern p-string $P \in \Sigma^+$.

```
1  Function cms(P, F_T, L_T, LF_T, E_T):
2      initialize M and the binary search tree of Lemma 3.12;
3      (q, ℓ, r, i) ← (0, 1, n, |P|);
4      while i ≥ 1 do
5          compute π(P[i..i+q]) and |P[i+1..i+q]|_p;
6          [ℓ'..r'] ← crangeupdrmq([ℓ..r], π(P[i..i+q]), |P[i+1..i+q]|_p,
               F_T, L_T, LF_T);                                  // Alg. 1
7          if ℓ' ≤ r' then                          // expand previous match
8              q ← q + 1;
9              M[i] ← (q, ℓ', r');
10             [ℓ..r] ← [ℓ'..r'];
11             i ← i − 1;
12         else if q = 0 then                                    // reset
13             M[i] ← (0, 1, n);
14             i ← i − 1;
15         else                                  // contract previous match
16             q ← q − 1;
17             [ℓ..r] ← MI_{E_T}(r, q);
18     return M;
```

Line 5 in the proof of Corollary 5.2. All branches except Line 15 decrement i. However, this branch decrements q. Since $q \leq m$, the algorithm terminates.

For correctness, we consider the loop invariant that $M[i+1..]$ has been correctly computed and $[\ell..r] = \mathrm{CR}_{\mathcal{T}}(P[i+1..i+q])$. Clearly, this invariant holds initially. Assume the invariant holds for a tuple (i, q) with $i \in [1..m]$, $q \in [0..m-i]$ and $[\ell..r] = \mathrm{CR}_{\mathcal{T}}(P[i+1..i+q])$, i.e., if $i < m$, then $P[i+1..i+M[i].q]$ is the longest substring of P starting at position i that p-matches some prefix of an infinitely iterated conjugate considered in $\mathcal{T}$.

Case 1. Assume $i = m$. Then $q = 0$. If $[\ell'..r'] = \mathrm{CR}_{\mathcal{T}}(P[m])$ is non-empty, then $P[m]$ is the longest substring of $P[m..] = P[m]$ starting at position $i = m$ that p-matches some prefix of an infinitely iterated conjugate considered in $\mathcal{T}$, i.e., we enter Line 7 where we set $M[m] = (q+1, \ell', r')$, adjust i, q and update $[\ell..r]$. If on the other hand $[\ell'..r']$ is empty, then ε

is the longest substring of $P[m..] = P[m]$ starting at position $i = m$ that p-matches some prefix of an infinitely iterated conjugate considered in $\mathcal{T}$, i.e., we enter Line 12, where we set $M[m] = (0, 1, n)$, decrement i by 1 and leave $q = 0$ and $[\ell..r] = [1..n]$ as is. In either case, the invariant holds after the iteration of the loop.

Case 2. Assume $i < m$ and $q = M[i+1].q$. If $q = 0$, then the proof is similar to that of Case 1. So assume $q \geq 1$. If $[\ell'..r'] = \mathrm{CR}_{\mathcal{T}}(P[i..i+q])$ is non-empty, then $P[i..i+q]$ is the longest substring of P starting at position i that p-matches some prefix of an infinitely iterated conjugate considered in $\mathcal{T}$ by assumption on $M[i+1..]$, i.e., we enter Line 7, where we set $M[i]$ and maintain the invariant. If $[\ell'..r'] = \mathrm{CR}_{\mathcal{T}}(P[i..i+q])$ is empty, then we will consider $P[i..i+q-1]$ as a candidate for the longest substring of P starting at position i that p-matches some prefix of an infinitely iterated conjugate considered in $\mathcal{T}$ in the next iteration, i.e., we enter Line 15 and maintain the invariant by adjusting q and $[\ell..r]$.

Case 3. Assume $i < m$ and $0 \leq q < M[i+1].q$. Note that we only enter this case if $\mathrm{CR}_{\mathcal{T}}(P[i..i+q+k])$ does not p-match any prefix of an infinitely iterated conjugate considered in $\mathcal{T}$ for each $k \in \mathbb{N}$. If $[\ell'..r'] = \mathrm{CR}_{\mathcal{T}}(P[i..i+q])$ is empty, then we argue similarly to Case 1 if $q = 0$ and similarly to Case 2 if $q \neq 0$. Thus, assume $[\ell'..r'] = \mathrm{CR}_{\mathcal{T}}(P[i..i+q])$ is non-empty. Then $P[i..i+q]$ is the longest substring of P starting at position i that p-matches some prefix of an infinitely iterated conjugate considered in $\mathcal{T}$ by our remark, i.e., we enter Line 7, where we set $M[i]$ and maintain the invariant.

□

Corollary 5.2. *There exists a* $2n\lg\sigma + n\lg 3n' + 4n + O(n\frac{\lg\lg n}{\lg n})$ *bits of space data structure that supports the computation of circular matching statistics between* $\mathcal{T} \subset \Sigma^+$ *and any pattern p-string in* $O(m(\lg\sigma + \lg n))$ *time, where* m *is the pattern length and* n' *is the maximum length of all texts in* $\mathcal{T}$.

Proof. The data structure consists of the static epBWT index from Theorem 3.13, $\mathrm{E}_{\mathcal{T}}$ represented by the data structure of Lemma 2.2 and an additional copy of $\mathrm{E}_{\mathcal{T}}$ as a simple string for access occupying $n\lg 3n'$ bits of space, which results in the claimed space complexity.

We examine Algorithm 10 for the claim regarding time complexity. To obtain the values in Line 5 in $O(\lg\sigma)$ time, we proceed similarly to the proof of Lemma 3.12, but possibly remove nodes in the binary search tree if we enter Line 12 or Line 15. Then Line 6 is computed in $O(\lg\sigma)$ time by the proof of Theorem 3.13. We can perform the MI query of Line 17

i	1	2	3	4	5	6	7	8
$P[i]$	C	B	A	A	b	C	$	b
$M[i].q$	3	3	2	3	2	1	0	1
$M[i].\ell$	12	10	6	5	2	4	1	2
$M[i].r$	12	11	8	5	3	12	12	3

Figure 11: The circular matching statistics M between the pattern p-string $P = \texttt{CBAAbC\$b}$ and our running example $\mathcal{T} = \{\texttt{Bab}, \texttt{ABBA}, \texttt{CAbBB}\}$.

by exponential search on the RmQ values of ranges in $\mathrm{E}_{\mathcal{T}}$. The exponential search requires $O(\lg n)$ RmQ queries, and thus the computation takes $O(\lg n)$ time by Lemma 2.2. As each of the cases in the while-loop is entered at most m times until termination, the claim follows. □

We compute the circular matching statistics M between our running example $\mathcal{T} = \{\texttt{Bab}, \texttt{ABBA}, \texttt{CAbBB}\}$ and the pattern p-string $P = \texttt{CBAAbC\$a}$. The solution is also presented in Figure 11.

We begin by computing $\mathrm{CR}_{\mathcal{T}}(P[8..]) = \mathrm{CR}_{\mathcal{T}}(\texttt{b}) = [2..3]$, which is non-empty. Thus, $M[8] = (1, 2, 3)$. Next, $\mathrm{CR}_{\mathcal{T}}(P[7..]) = \mathrm{CR}_{\mathcal{T}}(\texttt{\$b}) = \emptyset$, and consequently we contract the previous match and compute $\mathrm{CR}_{\mathcal{T}}(P[7..7]) = \mathrm{CR}_{\mathcal{T}}(\$) = \emptyset$. We conclude that ε is the longest prefix of $P[7..]$ p-matching an infinitely iterated conjugate considered in $\mathcal{T}$, set $M[7] = (0, 1, 12)$, and reset. Afterwards, we compute $\mathrm{CR}_{\mathcal{T}}(P[6..6]) = \mathrm{CR}_{\mathcal{T}}(\texttt{C}) = [4..12]$, $\mathrm{CR}_{\mathcal{T}}(P[5..6]) = \mathrm{CR}_{\mathcal{T}}(\texttt{bC}) = [2..3]$ and $\mathrm{CR}_{\mathcal{T}}(P[4..6]) = \mathrm{CR}_{\mathcal{T}}(\texttt{AbC}) = [5..5]$, which result in $M[6] = (1, 4, 12)$, $M[5] = (2, 5, 6)$ and $M[4] = (3, 5, 5)$, respectively. We get empty intervals for $\mathrm{CR}_{\mathcal{T}}(P[3..6]) = \mathrm{CR}_{\mathcal{T}}(\texttt{AAbC})$ and $\mathrm{CR}_{\mathcal{T}}(P[3..5]) = \mathrm{CR}_{\mathcal{T}}(\texttt{AAb})$, but then $\mathrm{CR}_{\mathcal{T}}(P[3..4]) = \mathrm{CR}_{\mathcal{T}}(\texttt{AA}) = [6..8]$, which yields $M[3] = (2, 6, 8)$. We have $\mathrm{CR}_{\mathcal{T}}(P[2..4]) = \mathrm{CR}_{\mathcal{T}}(\texttt{BAA}) = [10..11]$ and consequently $M[2] = (3, 10, 11)$. The conjugate range $\mathrm{CR}_{\mathcal{T}}(P[1..4]) = \mathrm{CR}_{\mathcal{T}}(\texttt{CBAA})$ is empty again, but $\mathrm{CR}_{\mathcal{T}}(P[1..3]) = \mathrm{CR}_{\mathcal{T}}(\texttt{CBA}) = [12..12]$ and thus $M[1] = (3, 12, 12)$.

Remark 5.3. We can construct $\mathrm{E}_{\mathcal{T}}$ alongside the epBWT index of $\mathcal{T}$ as follows. We represent $\mathrm{E}_{\mathcal{T}}$ by the data structure of Lemma 2.4, occupying $O(n \lg n)$ bits of space at the expense of compact space. Using the setting of Section 4, we assume to have already built the epBWT index of $\mathcal{R} \subset (\Sigma - \{\$\})^+$ and $\mathrm{E}_{\mathcal{R}}$ and want to extend it by some $S \in (\Sigma - \{\$\})^+$ with S not shorter than the longest text in $\mathcal{R}$. Initially $\mathrm{E}_{\emptyset} = \varepsilon$. Given $S' = S^4\$$, we compute both $\mathrm{E}_{\{S'\}}$ and the circular matching statistics between $\mathcal{R}$ and S', where we note that the former is simply the parameterized LCP array [30]

i	1	2	3	4	5
$P[i]$	A	A	C	C	b
$M[i].q$	4	3	2	2	1
$M[i].\ell$	7	10	6	5	2
$M[i].r$	8	11	8	5	3

i	1	2	3	4	5
$P[i]$	A	A	C	C	b
$M'[i].q$	4	3	2	4	4
$M'[i].\ell$	7	10	6	5	3
$M'[i].r$	8	11	8	5	3

Figure 12: Comparison of the matching statistics M and M' between the pattern p-string $P = \texttt{AACCb}$ and our running example $\mathcal{T} = \{\texttt{Bab}, \texttt{ABBA}, \texttt{CAbBB}\}$.

of S' by choice of S'. We use the circular matching statistics to compute the equivalent of $\mathrm{TOP}_{\mathcal{S}}$ and $\mathrm{UDN}_{\mathcal{S}}$, and then proceed similarly to Line 3 through Line 9 of Algorithm 4 to obtain $\mathrm{E}_{\mathcal{S}}$.

5.1. Circular Pattern p-Strings

We can also consider a pattern p-string to be circular, which motivates the following definition. Given $\mathcal{T} \subset \Sigma^+$ and a pattern p-string $P \in \Sigma^+$ of length m, the *modified circular matching statistics* M' denotes an array whose entries are triples of a prefix length $M'[i].q \in [0..m]$, a left interval boundary $M'[i].\ell \in [1..n]$ and a right interval boundary $M'[i].r \in [1..n]$ such that for each $i \in [1..m]$

(i) $M'[i].q = \max\{\mathrm{lcp}(\langle C_{\mathcal{T}}(\mathrm{CA}_{\mathcal{T}}[j])^{\omega}[..m]\rangle, \langle P[i..] \cdot P[..i-1]\rangle) \mid j \in [1..n]\}$,

(ii) $M'[i].\ell \leq M'[i].r$ and

(iii) $j \in [M'[i].\ell..M'[i].r] \Leftrightarrow \mathrm{lcp}(\langle C_{\mathcal{T}}(\mathrm{CA}_{\mathcal{T}}[j])^{\omega}[..m]\rangle, \langle P[i..] \cdot P[..i-1]\rangle) = M'[i].q$.

We give an example in Figure 12. To compute M', we slightly modify Algorithm 10. In detail, we first compute M and then initialize the values in Line 3 with $(M[1].q, M[1].\ell, M[1].r, |P|)$. Also, in the while-loop we first check for $q = |P|$ and contract the match as we do in Line 15 if the statement holds.

As a side product, we also solve the circular pattern matching problem in the parameterized setting, where we consider texts and the pattern as circular strings.

6. Conclusion

We introduced the epBWT-index as a new type of index capable to search a pattern in a set of circular texts in the parameterized pattern matching setting. The main goal of the epBWT-index is to solve the problem of counting the number of pattern occurrences, which we formalized as EP-COUNT. The main ideas stem from combining core concepts of (a) the pBWT, which is a parameterized index for a single non-circular text, and (b) the eBWT, which is an index for multiple circular texts. We could show that it is possible to solve EP-COUNT with our epBWT within the same space and time complexities as Kim and Cho's pBWT variant [14]. Further, we considered the input set of circular texts to be dynamic, supporting the insertion and the removal of a text. To that end, we followed the approach of Iseri et al. [15] who proposed a construction algorithm of Kim and Cho's pBWT, and extended this construction algorithm to construct and extend the epBWT. We inherited the time complexity of the technique of Iseri et al., which makes the epBWT construction faster for general parameterized alphabets to construction algorithms for other parameterized indexes, whose time complexities usually have a multiplicative dependency on the input length and the size of the parameterized alphabet. Finally, as an application of the epBWT, we show how to compute generalized versions of matching statistics to parameterized and circular matching. We conclude this article with future work spawning two different research directions: lowering the space and adaptation to other generalized pattern matchings.

6.1. Lowering Space

First, we did not address the pBWT of Ganguly et al. [13] in detail, who can escape the requirement of storing $\mathrm{F}_{\mathcal{T}}$ by having a compressed suffix tree topology at hand. It is unclear whether we can make use of their approach, in particular if we want to emphasize on construction.

Second, to actually locate all the reported occurrences within compact space, we want to shrink the conjugate array sampling addressed in Remark 4.12 to fit into $O(n)$ bits of space. The main question here is how to get rid of the $O(d \lg n)$ term, which is due to the LF-mapping having at least one cycle for each of the indexed texts.

Finally, for matching statistics to work in compact space, the string $\mathrm{E}_{\mathcal{T}}$ remains the bottleneck. An idea is to investigate new compression techniques for $\mathrm{E}_{\mathcal{T}}$. However, a major problem is that the values of $\mathrm{E}_{\mathcal{T}}$ are based on the

prev-encoded conjugates, for which a permuted LCP-like compression [47] seems not to work. Also, we are unaware of how to compute $\mathrm{E}_{\mathcal{T}}$ efficiently since we rely on the construction of the parameterized LCP array, whose time is only linear for constant alphabets [34].

6.2. Practical Considerations

Due to the fact that the standard FM-index is a well-perceived data structure for practical application, especially in bioinformatics, it is natural to investigate the practical performance of our epBWT-index. For that it looks promising to first gather more ideas on algorithmic improvements on the practical side, which may not have an impact on the asymptotic time complexity — there are certainly several screws to turn, such as improving Lemma 3.4 to compare $2z$ instead of $3z$ characters. (This was pointed out by one reviewer of this manuscript.)

Acknowledgements. This research was supported by JSPS KAKENHI with grant numbers `JP23H04378` and `JP25K21150`.

Appendix A. Variables

The table below gives an overview of variables used throughout this article.

variable name	*description*
$i, j, k, v, x, y, \alpha, \beta$	index or natural number
$\Sigma, \Sigma_{\mathrm{s}}, \Sigma_{\mathrm{p}}$	alphabet
ψ	permutation of Σ_{p}
$\sigma, \sigma_{\mathrm{s}}, \sigma_{\mathrm{p}}$	size of $\Sigma, \Sigma_{\mathrm{s}}, \Sigma_{\mathrm{p}}$
$\$, \aleph, f, g$	symbol
ε	empty string
$U, W, V, V_1, ..$	string
$S, T, T_1, ..$	text string
$\lambda, n, n_1, ..$	length of $S, T, T_1, ..$
$S', T', T'_1, ..$	$S^4\$, T^4\$, T_1^4\$, ..$
$\mathcal{S}, \mathcal{R}, \mathcal{T}, \mathcal{T}_1, ..$	set of text strings
d	number of texts in $\mathcal{T}$
ρ	total length of all texts in $\mathcal{R}$
z, n'	maximum length of text in a set of texts

$\mathcal{S}', \mathcal{T}'_1, \mathcal{T}'_2, ..$	$\mathcal{R} \cup \{S'\}, \{T'_1\}, \mathcal{T}_1 \cup \{T'_2\}, ..$
$P, P_1, ..$	pattern string
m	length of P
$\mathrm{SA}_T, \mathrm{SA}_T^{-1}$	parameterized suffix array and its inverse
$\mathrm{SR}_T(P)$	suffix range of P in T
$\ell, \ell', \ell'', r, r', r''$	interval boundary
$\mathrm{F}_T, \mathrm{L}_T, \mathrm{LCP}_T^{\infty}$	the pBWT index of T
LF_T	LF-mapping on the suffix array of T
$\mathrm{A}_T, \mathrm{B}_T$	augmenting pBWT index to solve EP-COUNT
$C_{\mathcal{T}}(i)$	a conjugate of a text in $\mathcal{T}$
$\mathrm{CA}_{\mathcal{T}}, \mathrm{CA}_{\mathcal{T}}^{-1}$	conjugate array of $\mathcal{T}$ and its inverse
$\mathrm{CR}_{\mathcal{T}}(P)$	conjugate range of P in $\mathcal{T}$
δ	length of a conjugate range
$\mathrm{LEN}_{\mathcal{T}}, \mathrm{PRV}_{\mathcal{T}}, \mathrm{ENC}_{\mathcal{T}}$	conceptual string for defining the epBWT of $\mathcal{T}$
$\mathrm{LF}_{\mathcal{T}}, \mathrm{FL}_{\mathcal{T}}$	LF-/FL-mapping of the conjugate array of $\mathcal{T}$
$\mathrm{F}_{\mathcal{T}}, \mathrm{L}_{\mathcal{T}}, \mathrm{LCP}_{\mathcal{T}}^{\infty}$	the epBWT index of $\mathcal{T}$
h	a return value of $\pi(..)$
c	a return value of $\lvert..\rvert_{\mathrm{p}}$
$\mathrm{CNT}_{\mathcal{S}}$	conceptual string for extending epBWT index
$\mathrm{P}_S, \mathrm{Y}_S, \mathrm{TOP}_{\mathcal{S}}, \mathrm{UDN}_{\mathcal{S}}$	auxiliary string for epBWT index extension
$q, \gamma, t', t_1, ..$	length of a prefix
e, e'	number of $\mathbb{x}$ in a prefix
$\mathrm{D}_{\mathcal{T}}, \mathrm{D}'_{\mathcal{T}}$	auxiliary string for restoring indexed texts
M, M'	matching statistics
$\mathrm{E}_{\mathcal{T}}$	conceptual string for contraction operation

References

[1] E. M. Osterkamp, D. Köppl, Extending the parameterized Burrows–Wheeler transform, in: Proc. DCC, 2024, pp. 143–152.

[2] G. Bell, T. Hey, A. Szalay, Beyond the data deluge, Science 323 (5919) (2009) 1297–1298. doi:10.1126/science.1170411.

[3] P. Ferragina, G. Manzini, Opportunistic data structures with applications, in: Proc. FOCS, 2000, pp. 390–398. doi:10.1109/SFCS.2000.892127.

[4] M. Burrows, D. J. Wheeler, A block sorting lossless data compression algorithm, Tech. Rep. 124, Digital Equipment Corporation, Palo Alto, California (1994).

[5] R. Grossi, J. S. Vitter, B. Xu, Wavelet trees: From theory to practice, in: Proc. CCP, 2011, pp. 210–221. doi:10.1109/CCP.2011.16.

[6] C. S. Iliopoulos, M. S. Rahman, Indexing circular patterns, in: Proc. WALCOM, Vol. 4921 of LNCS, 2008, pp. 46–57. doi:10.1007/978-3-540-77891-2_5.

[7] B. L. Strang, N. D. Stow, Circularization of the herpes simplex virus type 1 genome upon lytic infection, Journal of virology 79 (19) (2005) 12487–12494.

[8] T. B. H. Karoline K. Ebbesen, J. Kjems, Insights into circular RNA biology, RNA Biology 14 (8) (2017) 1035–1045. doi:10.1080/15476286.2016.1271524.

[9] A. F. Nielsen, A. Bindereif, I. Bozzoni, M. Hanan, T. B. Hansen, M. Irimia, S. Kadener, L. S. Kristensen, I. Legnini, M. Morlando, M. T. Jarlstad Olesen, R. J. Pasterkamp, S. Preibisch, N. Rajewsky, C. Suenkel, J. Kjems, Best practice standards for circular rna research, Nature Methods 19 (10) (2022) 1208–1220. doi:10.1038/s41592-022-01487-2.

[10] S. Mantaci, A. Restivo, G. Rosone, M. Sciortino, An extension of the Burrows–Wheeler transform, Theor. Comput. Sci. 387 (3) (2007) 298–312. doi:10.1016/j.tcs.2007.07.014.

[11] B. S. Baker, A theory of parameterized pattern matching: algorithms and applications, in: Proc. STOC, 1993, pp. 71–80. doi:10.1145/167088.167115.

[12] T. Shibuya, Generalization of a suffix tree for RNA structural pattern matching, Algorithmica 39 (1) (2004) 1–19. doi:10.1007/S00453-003-1067-9.

[13] A. Ganguly, R. Shah, S. V. Thankachan, pBWT: Achieving succinct data structures for parameterized pattern matching and related problems, in: Proc. SODA, 2017, pp. 397–407. doi:10.1137/1.9781611974782.25.

[14] S. Kim, H. Cho, Simpler FM-index for parameterized string matching, Inf. Process. Lett. 165 (2021) 106026. doi:10.1016/j.ipl.2020.106026.

[15] K. Iseri, T. I, D. Hendrian, D. Köppl, R. Yoshinaka, A. Shinohara, Breaking a barrier in constructing compact indexes for parameterized pattern matching, in: Proc. ICALP, Vol. 297 of LIPIcs, 2024, pp. 89:1–89:19. doi:10.4230/LIPICS.ICALP.2024.89.

[16] W. I. Chang, E. L. Lawler, Sublinear approximate string matching and biological applications, Algorithmica 12 (4/5) (1994) 327–344. doi:10.1007/BF01185431.

[17] P. Weiner, Linear pattern matching algorithms, in: Proc. SWAT, 1973, pp. 1–11. doi:10.1109/SWAT.1973.13.

[18] I. M. Gessel, C. Reutenauer, Counting permutations with given cycle structure and descent set, J. Comb. Theory, Ser. A 64 (2) (1993) 189–215.

[19] W. Hon, C. Lu, R. Shah, S. V. Thankachan, Succinct indexes for circular patterns, in: Proc. ISAAC, Vol. 7074 of LNCS, 2011, pp. 673–682. doi:10.1007/978-3-642-25591-5_69.

[20] W. Hon, T. Ku, C. Lu, R. Shah, S. V. Thankachan, Efficient algorithm for circular Burrows–Wheeler transform, in: Proc. CPM, Vol. 7354 of LNCS, 2012, pp. 257–268. doi:10.1007/978-3-642-31265-6_21.

[21] N. Cotumaccio, Improved circular dictionary matching (2025). arXiv:2504.03394.
URL https://arxiv.org/abs/2504.03394

[22] S. Bonomo, S. Mantaci, A. Restivo, G. Rosone, M. Sciortino, Sorting conjugates and suffixes of words in a multiset, Int. J. Found. Comput. Sci. 25 (8) (2014) 1161. doi:10.1142/S0129054114400309.

[23] A. Policriti, N. Prezza, Computing LZ77 in run-compressed space, in: Proc. DCC, 2016, pp. 23–32. doi:10.1109/DCC.2016.30.

[24] T. Ohno, Y. Takabatake, T. I, H. Sakamoto, A faster implementation of online run-length Burrows–Wheeler transform, in: Proc. IWOCA, Vol. 10765 of LNCS, 2017, pp. 409–419. doi:10.1007/978-3-319-78825-8_33.

[25] H. Bannai, J. Kärkkäinen, D. Köppl, M. Piątkowski, Constructing and indexing the bijective and extended Burrows–Wheeler transform, Inf. Comput. 297 (105153) (2024) 1–30. doi:10.1016/j.ic.2024.105153.

[26] C. Boucher, D. Cenzato, Z. Lipták, M. Rossi, M. Sciortino, r-indexing the eBWT, Inf. Comput. 298 (2024) 105155. doi:10.1016/J.IC.2024.105155.

[27] J. Olbrich, E. Ohlebusch, T. Büchler, Generic non-recursive suffix array construction, ACM Trans. Algorithms 20 (2) (2024) 18. doi:10.1145/3641854.

[28] B. S. Baker, Parameterized duplication in strings: Algorithms and an application to software maintenance, SIAM J. Comput. 26 (5) (1997) 1343–1362. doi:10.1137/S0097539793246707.

[29] J. Mendivelso, S. V. Thankachan, Y. J. Pinzón, A brief history of parameterized matching problems, Discret. Appl. Math. 274 (2020) 103–115. doi:10.1016/j.dam.2018.07.017.

[30] S. Deguchi, F. Higashijima, H. Bannai, S. Inenaga, M. Takeda, Parameterized suffix arrays for binary strings, in: Proc. PSC, 2008, pp. 84–94.

[31] T. I, S. Deguchi, H. Bannai, S. Inenaga, M. Takeda, Lightweight parameterized suffix array construction, in: Proc. IWOCA, Vol. 5874 of LNCS, 2009, pp. 312–323. doi:10.1007/978-3-642-10217-2_31.

[32] R. Beal, D. A. Adjeroh, Variations of the parameterized longest previous factor, J. Discrete Algorithms 16 (2012) 129–150. doi:10.1016/j.jda.2012.05.004.

[33] D. Hendrian, T. Katsura, Y. Otomo, K. Narisawa, A. Shinohara, Position heaps for parameterized strings, in: Proc. CPM, Vol. 78 of LIPIcs, 2017, pp. 8:1–8:13. doi:10.4230/LIPICS.CPM.2017.8.

[34] N. Fujisato, Y. Nakashima, S. Inenaga, H. Bannai, M. Takeda, Direct linear time construction of parameterized suffix and LCP arrays for constant alphabets, in: Proc. SPIRE, Vol. 11811 of LNCS, 2019, pp. 382–391. doi:10.1007/978-3-030-32686-9_27.

[35] K. Nakashima, D. Hendrian, R. Yoshinaka, A. Shinohara, An extension of linear-size suffix tries for parameterized strings, in: Proc. SOFSEM, Vol. 2568 of CEUR Workshop Proceedings, 2020, pp. 97–108.

[36] N. Fujisato, Y. Nakashima, S. Inenaga, H. Bannai, M. Takeda, The parameterized suffix tray, in: Proc. CIAC, Vol. 12701 of LNCS, 2021, pp. 258–270. doi:10.1007/978-3-030-75242-2_18.

[37] A. Ganguly, R. Shah, S. V. Thankachan, Fully functional parameterized suffix trees in compact space, in: Proc. ICALP, Vol. 229 of LIPIcs, 2022, pp. 65:1–65:18. doi:10.4230/LIPIcs.ICALP.2022.65.

[38] K. Nakashima, N. Fujisato, D. Hendrian, Y. Nakashima, R. Yoshinaka, S. Inenaga, H. Bannai, A. Shinohara, M. Takeda, Parameterized DAWGs: Efficient constructions and bidirectional pattern searches, Theor. Comput. Sci. 933 (2022) 21–42. doi:10.1016/J.TCS.2022.09.008.

[39] D. Hashimoto, D. Hendrian, D. Köppl, R. Yoshinaka, A. Shinohara, Computing the parameterized Burrows–Wheeler transform online, in: Proc. SPIRE, Vol. 13617 of LNCS, 2022, pp. 70–85. doi:10.1007/978-3-031-20643-6_6.

[40] E. Ohlebusch, J. Fischer, S. Gog, CST++, in: Proc. SPIRE, Vol. 6393 of LNCS, 2010, pp. 322–333. doi:10.1007/978-3-642-16321-0_34.

[41] V. Mäkinen, G. Navarro, Position-restricted substring searching, in: Proc. LATIN, Vol. 3887 of LNCS, 2006, pp. 703–714. doi:10.1007/11682462_64.

[42] P. G. S. da Fonseca, I. B. F. da Silva, Online construction of wavelet trees, in: 16th International Symposium on Experimental Algorithms, SEA 2017, June 21-23, 2017, London, UK, Vol. 75 of LIPIcs, 2017, pp. 16:1–16:14. doi:10.4230/LIPIcs.SEA.2017.16.

[43] J. Fischer, V. Heun, Space-efficient preprocessing schemes for range minimum queries on static arrays, SIAM J. Comput. 40 (2) (2011) 465–492. doi:10.1137/090779759.

[44] J. I. Munro, Y. Nekrich, J. S. Vitter, Dynamic data structures for document collections and graphs, in: Proc. PODS, 2015, pp. 277–289. doi:10.1145/2745754.2745778.

[45] M. Crochemore, W. Rytter, Jewels of Stringology: Text Algorithms, World Scientific, 2002.

[46] E. Ohlebusch, S. Gog, A. Kügel, Computing matching statistics and maximal exact matches on compressed full-text indexes, in: Proc. SPIRE, Vol. 6393 of LNCS, 2010, pp. 347–358. doi:10.1007/978-3-642-16321-0_36.

[47] K. Sadakane, Succinct representations of lcp information and improvements in the compressed suffix arrays, in: Proc. SODA, 2002, pp. 225–232.