Extending the Burrows–Wheeler Transform for Cartesian Tree Matching and Constructing It

Eric M. Osterkamp
University of Münster, Germany

Dominik Köppl
University of Yamanashi, Kofu, Japan

Abstract

Cartesian tree matching is a form of generalized pattern matching where a substring of the text matches with the pattern if they share the same Cartesian tree. This form of matching finds application for time series of stock prices and can be of interest for melody matching between musical scores. For the indexing problem, the state-of-the-art data structure is a Burrows–Wheeler transform based solution due to [Kim and Cho, CPM'21], which uses nearly succinct space and can count the number of substrings that Cartesian tree match with a pattern in time linear in the pattern length. The authors address the construction of their data structure with a straight-forward solution that, however, requires pointer-based data structures, resulting in $O(n \lg n)$ bits of space, where n is the text length [Kim and Cho, CPM'21, Section A.4]. We address this bottleneck by a construction that requires $O(n \lg \sigma)$ bits of space and has a time complexity of $O(n \frac{\lg \sigma \lg n}{\lg \lg n})$, where σ is alphabet size. Additionally, we can extend this index for indexing multiple circular texts in the spirit of the extended Burrows–Wheeler transform without sacrificing the time and space complexities. We present this index in a dynamic variant, where we pay a logarithmic slowdown and need space linear in the input texts in bits for the extra functionality that we can incrementally add texts. Our extended setting is of interest for finding repetitive motifs common in the aforementioned applications, independent of offsets and scaling.

2012 ACM Subject Classification Theory of computation

Keywords and phrases Cartesian tree matching, extended Burrows–Wheeler transform, construction algorithm, generalized pattern matching

Digital Object Identifier 10.4230/LIPIcs.CPM.2025.26

Related Version *Full Version*: `https://arxiv.org/abs/2411.12241`

Funding *Dominik Köppl*: JSPS KAKENHI Grant Numbers JP23H04378 and JP25K21150.

Acknowledgements We sincerely thank the anonymous reviewers for their constructive comments.

1 Introduction

String matching is ubiquitous in computer science, and its variations are custom-made to solve a wide variety of problems. We here focus on a special kind of variation called *substring consistent equivalence relation* (*SCER*) [21]. Two strings X and Y are said to SCER-match if they have the same length and all substrings of equal length starting at the same text position SCER-match, i.e., $X[i..j]$ SCER-matches with $Y[i..j]$ for all $1 \leq i \leq j \leq |X| = |Y|$. A specific instance of SCER-matching is *order-preserving matching* [19, 16], which has been studied for the analysis of numerical time series. The aim of order-preserving matching is to match two strings if the relative order of the symbols in both strings is the same. Order-preserving matching therefore can find matches independently of value offsets and scaling.

Since order-preserving matching takes the global order of the symbols in a string into account, it may be too strict in applications that primarily consider the local ordering of ranges partitioned by the peaks. In fact, time series of stock prices are such a case, where

a common pattern called the *head-and-shoulder* [8] involves one absolute peak (head) and neighboring local peaks (shoulders), where each of these neighboring peaks can be treated individually to match similar head-and-shoulder patterns with slightly changed local peaks. For such a reason, Park et al. [26] proposed *Cartesian tree matching*. Cartesian tree matching relaxes the notion of order-preserving matching in the sense that an order-preserving match is always a Cartesian tree match, but not necessarily the other way around. For instance, the two strings of digits 1537462 and 2438372 fit into the head-and-shoulder pattern, do not order-preserving match, but Cartesian tree match. For that, Cartesian tree matching compares the Cartesian trees of two strings to decide whether they match. The *Cartesian tree* [28] is a binary tree built upon an array of numbers. If all numbers are distinct, it is the min-heap whose in-order traversal retrieves the original array. Since its inception, Cartesian tree matching and variations thereof have attracted interest from researchers, whose studies include multiple pattern matching [26, 27], approximate matching [1, 18], substring matching [4], subsequence matching [24], indeterminate matching [10], cover matching [15], and palindromic matching [9].

In addition to stock prices, applications of generalized pattern matching can also be found in melody matching between musical scores. Pattern matching music scores such as differences, directions, or magnitudes have been studied [12]. However, the detection of repetitions in a piece of music has also been considered of importance [7]. The question therefore is whether we can find repetitive substrings that Cartesian tree match with a repetition of a set of melodic motifs (i.e., input texts). For that to be efficient, we want to index these motifs.

An index for Cartesian tree matching of a text string T is a data structure built upon T that can report the number of substrings of T that Cartesian tree match a given pattern. Given T has n symbols drawn from an integer alphabet of size σ, Park et al. [26] and Nishimoto et al. [23] proposed indexes for Cartesian tree matching of T that both occupy $O(n \lg n)$ bits of space and are constructed with $O(n \lg n)$ and $O(n\sigma \lg n)$ additional bits of space, respectively. Park et al. [26, Section 5.1] achieve $O(m \lg \sigma)$ time and Nishimoto et al. [23, Section 5.1] $O(m(\sigma + \lg m) + occ)$ time for answering count queries, where occ is the number of occurrences of the pattern of length m. Adapting Ferragina and Mantaci's FM-index for exact string matching [5], Kim and Cho [17] proposed an index that occupies $3n + o(n)$ bits of space, and answers a count query in $O(m)$ time for a pattern of length m. For construction, they proposed a straight-forward solution that takes as input the Cartesian suffix tree of Park et al. [26], which, however, requires $O(n \lg n)$ additional bits of space.

We address two goals in this paper. The first is a construction algorithm for Kim and Cho's index that takes $O(n \lg \sigma)$ bits of working space, which is compact for constant alphabet sizes. While we consider an index supporting access to a character of the length-n text T as compact if its space is $O(n \lg \sigma)$ bits, we here restrict queries to Cartesian-tree pattern matching, and a Cartesian tree storing n nodes can be represented in $2n$ bits, cf. [26, Section 3.5]. Therefore, a compact index for Cartesian-tree pattern matching takes $O(n)$ bits, a bound we reach for constant alphabets. Second, while all aforementioned indexes can partially address the problem by indexing multiple texts, it is hard to detect whether the pattern is a repetition of one of the input texts, which is of interest in case of indexing melodic motifs that can repeat. Here, our goal is an index that can find such matches, even if they start with different offsets of the same repetition. In concrete words, our aim is to index multiple texts for Cartesian pattern matching. The search space for a pattern are the texts that are considered to be infinite concatenations with themselves.

In this paper, we propose an extension of the index of Kim and Cho [17] for Cartesian tree matching with techniques of the extended Burrows–Wheeler transform [20], and call the resulting data structure the cBWT index. The cBWT index is an extension in the sense that

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

Figure 1 Example integer string V to illustrate queries. Here, $\mathrm{rank}_{11}(V,5) = 1$, $\mathrm{rnkcnt}_V(2,7,7,14) = 3$, $\mathrm{select}_{11}(V,2) = 7$, $\mathrm{RNV}_V(1,5,8) = 11$, and $\mathrm{MI}_V(5,6) = [4..8]$.

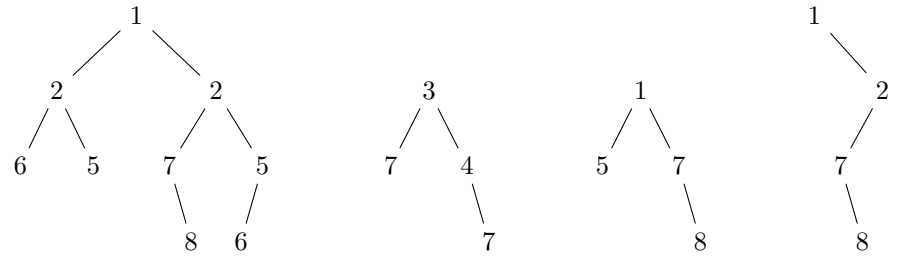
it supports indexing multiple input texts for Cartesian tree matching circularly. We show that we can compute the cBWT index in $O(n\frac{\lg\sigma\lg n}{\lg\lg n})$ time and $O(n\lg\sigma)$ bits of space, where n is the total length of all texts to index. Our construction allows to build the cBWT index incrementally, i.e., we support the incremental addition of a new string to the set of texts we index. We can also compute the original index of Kim and Cho within the same complexities. Our ideas stem from construction algorithms of indexes for parameterized matching by Hashimoto et al. [11] and Iseri et al. [14], which we recently extended for multiple circular texts [25]. During construction, the cBWT index supports the backward search and count queries for pattern strings in $O(m\frac{\lg\sigma\lg n}{\lg\lg n})$ time, where m is the pattern length.

2 Preliminaries

Let $\lg = \log_2$ denote the logarithm to base two. We assume a random access model with word size $\Omega(\lg n)$, where n denotes the input size. An interval $\{i, i+1, .., j-1, j\}$ of integers is denoted by $[i..j]$, where $[i..j] = \emptyset$ if $j < i$.

Strings. Let Σ denote an *alphabet*. We call elements of Σ *symbols*, a sequence of symbols from Σ a *string over* Σ, and denote the set of strings over Σ by Σ^*. Let $U, V, W \in \Sigma^*$. The *concatenation* of U and V is denoted by $U{\cdot}V$ or UV. We write U^k if we concatenate k instances of U for a non-negative integer k, and U^ω for the string obtained by infinitely concatenating U. We call U *primitive* if $U = X^k$ implies $U = X$ and $k = 1$. It is known that for every $U \in \Sigma^*$ there exists a unique primitive $X \in \Sigma^*$ and a unique integer k such that $U = X^k$, denoted by $\mathrm{root}(U)$ and $\exp(U)$, respectively. If $X = UVW$, then U is called a *prefix*, W a *suffix*, and U, V, W *substrings* of X. The *length* $|U|$ of U is the number of symbols in U, $\mathrm{lcp}(U,V)$ reports the length of the longest common prefix of U and V, ε denotes the unique string of length 0, and we define $\Sigma^+ = \Sigma^* - \{\varepsilon\}$. Let $X \in \Sigma^+$ and $i, j \in [1..|X|]$. Then $X[i]$ denotes the i-th symbol in X, and $X[i..j] = X[i]\cdots X[j]$, where $X[i..j] = \varepsilon$ if $j < i$, $X[..i] = X[1..i]$, and $X[i..] = X[i..|X|]$. For notational convenience, $X[..k] = \varepsilon$ if $k \le 0$, and $V[k..] = \varepsilon$ if $k \ge |V| + 1$. We call $1 \le p \le |X|$ satisfying $X[q] = X[q+p]$ for every $q \in [1..|X|-p]$ a *period* of X. Let $\mathrm{Rot}(X,0) = X$ and $\mathrm{Rot}(X,k+1) = \mathrm{Rot}(X,k)[2..]\cdot\mathrm{Rot}(X,k)[1]$ for each non-negative integer k, i.e., $\mathrm{Rot}(X,k)$ denotes the k-th *left rotation* of X. The left rotations of X for $k \in [0..|X|-1]$ are the *conjugates* of X. We write $U < V$ if and only if (a) $U = V[..|U|]$ and $|U| < |V|$ or (b) $U[\mathrm{lcp}(U,V)+1] < V[\mathrm{lcp}(U,V)+1]$.

Let $V \in \Sigma^+$ be a (dynamic) string, $c \le d \in \Sigma$, $i \in [1..|V|+1]$, $j \in [0..|V|]$ and $k \in [1..|V|]$. Then $\mathrm{insert}_V(i,c)$ inserts the symbol c at position i of V, $\mathrm{delete}_V(k)$ deletes the k-th entry of V, $\mathrm{rank}_c(V,j)$ returns the number of occurrences of c in $V[..j]$, $\mathrm{rnkcnt}_V(i,j,c,d)$ returns $|\{x \in [i..j] \mid c \le V[x] \le d\}|$, $\mathrm{select}_c(V,i)$ returns the index of the i-th occurrence of c in V if $i \le \mathrm{rank}_c(V,|V|)$, $\mathrm{RNV}_V(i,j,c)$ returns the smallest value in $V[i..j]$ larger than c if it exists, and $\mathrm{MI}_V(j,c)$ returns the maximal interval $[\ell..r]$ such that $0 \le \ell \le j \le r \le |V|$ and $V[x] \ge c$ for every $x \in [\ell+1..r]$. See Figure 1 for examples. We represent strings by the following dynamic data structure, which supports the aforementioned operations and queries.



(a) ct(T). **(b)** ct(P). **(c)** ct(T[3..6]). **(d)** ct(T[4..7]).

Figure 2 Cartesian trees of $T = \texttt{625178265}$, $P = \texttt{7347}$, $T[3..6] = \texttt{5178}$, and $T[4..7] = \texttt{1782}$. Here, $P \approx T[3..6]$ and $P \not\approx T[4..7]$.

▶ **Lemma 2.1** ([14, 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*,* rnkcnt*,* select*,* RNV *and* MI *in $O(\frac{\lg \sigma \lg n}{\lg \lg n})$ time.*

Alphabet. Throughout, we will work with the integer alphabet $\Sigma = [0..\sigma]$, where $\sigma \leq n^{O(1)}$, and a special symbol $\$ \notin \Sigma$ stipulated to be smaller than any symbol from Σ. The special symbol is motivated by the construction algorithm, and as a delimiter when a string should not be considered circular, as in the index of Kim and Cho [17]. Let $\Sigma_\$ = \Sigma \cup \{\$\}$.

Cartesian Tree Matching. The *Cartesian tree* ct(V) of a string $V \in \Sigma_\$^*$ is a binary tree defined as follows. If $V = \varepsilon$, then ct(V) is the empty tree. If $V \neq \varepsilon$, let i denote the position of the smallest symbol in V, where ties are broken with respect to text position. Then ct(V) has $V[i]$ as its root, ct($V[..i-1]$) as its left subtree and ct($V[i+1..]$) as its right subtree. We say that two strings $U, V \in \Sigma_\$^*$ *Cartesian tree match (ct-match)* if and only if ct(U) = ct(V), and write $U \approx V$. For instance, in Figure 2 the substring $T[3..6] = \texttt{5178}$ of $T = \texttt{625178265}$ ct-matches $P = \texttt{7347}$ while $T[4..7] = \texttt{1782}$ does not.

Parent Distance Encoding. Park et al. [26] use an encoding scheme for representing Cartesian trees that reduces the computation of a ct-match to checking whether the encoded strings exactly match. We here give a variant thereof which takes the special symbol \$ into account. Let $\aleph \notin \Sigma_\$$ denote a symbol larger than any integer. The *parent distance encoding* $\langle V \rangle$ of $V \in \Sigma_\$$ is a string of length $|V|$ over $\Sigma_\$ \cup \{\aleph\}$ such that

$$\langle V \rangle[i] = \begin{cases} \aleph & \text{if } \$ \neq V[i] < \min\{V[j] \mid j \in [..i-1]\}, \\ V[i] & \text{if } V[i] = \$, \\ i - \max\{j \in [..i-1] \mid V[j] \leq V[i]\} & \text{otherwise,} \end{cases}$$

for each $i \in [1..|V|]$. We have $\langle V \rangle[..i] = \langle V[..i] \rangle$ for each $V \in \Sigma_\$^+$ and $i \in [1..|V|]$. For example, $\langle \texttt{41327\$3} \rangle = \aleph\aleph\texttt{121\$1}$.[1]

▶ **Lemma 2.2** ([26, Theorem 1]). *Let $U, V \in \Sigma^*$. Then $U \approx V \Leftrightarrow \langle U \rangle = \langle V \rangle$.*

[1] As a side note, the parent distance encoding has been leveraged for devising compact $O(n)$-bit data structures answering range minimum queries, such as the 2d-Min-Heap by Fischer [6] and the LRM-trees by Sadakane and Navarro [22], which semantically coincide.

Problem Statement. We are interested in a solution to the following problem.

▶ **Problem** (Count). *Given $\emptyset \neq \mathcal{T} \subset \Sigma^+$ and $P \in \Sigma^*$, count each of the conjugates of the strings in $\mathcal{T}$ whose infinite iteration has a prefix ct-matching P.*

Throughout, let $\emptyset \neq \mathcal{T} = \{T_1, ..., T_d\} \subset \Sigma^+$. Our running example consists of the strings $T_1 = \mathtt{512}$, $T_2 = \mathtt{5363}$ and $T_3 = \mathtt{4478}$ over $\Sigma = [0..8]$. Given our running example, the solution to Count for $P_1 = \mathtt{643}$ and $P_2 = \mathtt{5634}$ is 0 and 2, respectively. Here, $P_2 \approx \mathrm{Rot}(T_3, 2)^\omega[..4] \approx \mathrm{Rot}(T_1, 2)^\omega[..4]$ since $\langle P_2 \rangle = \mathtt{𝕩1𝕩1} = \langle \mathrm{Rot}(T_3, 2)^\omega[..4] \rangle = \langle \mathrm{Rot}(T_1, 2)^\omega[..4] \rangle$.

3 $O(n \lg \sigma)$-bit Index

Let $n = |T_1 \cdots T_d|$, $n_k = |T_k|$ for each $k \in [1..d]$, and $C_{\mathcal{T}}(i) = \mathrm{Rot}(T_j, i - 1 - \sum_{k=1}^{j-1} n_k)$ for each $i \in [1..n]$, where $j = \min\{h \in [1..d] \mid \sum_{k=1}^{h} n_k \geq i\}$, i.e., we identify each conjugate of each text $T_1, ..., T_d$ with its start position inside the concatenation $T_1 \cdots T_d$, such that we give them ranks from 1 to n. See Figure 3 for an example. In what follows, we put these ranks in a specific order by a permutation of $[1..n]$ such that the permuted ranks of all conjugates with prefixes of their infinite concatenation ct-matching a pattern form an interval $[\ell..r] \subseteq [1..n]$.

3.1 Conjugate Array

We express the permutation of the ranks of all conjugates by the so-called conjugate array, which we will subsequently define. To achieve this, we extend the ideas of Mantaci et al. [20] to Cartesian tree matching, and introduce a preorder on $\Sigma_\$^+$. For notational convenience, we define the *rotational parent distance encoding* $\langle V \rangle_\mathrm{r}$ of $V \in \Sigma_\$^+$ by $\langle V \rangle_\mathrm{r} = \langle V^2 \rangle[|V| + 1..]$. For any $V, U \in \Sigma_\$^+$, let $V \preceq_\omega U$ if and only if there exists some natural number i such that $\langle V^\omega[..i] \rangle < \langle U^\omega[..i] \rangle$ or $\mathrm{root}(\langle V \rangle_\mathrm{r}) = \mathrm{root}(\langle U \rangle_\mathrm{r})$ holds.

▶ **Lemma 3.1.** *The relation $\preceq_\omega$ defines a total preorder on $\Sigma_\$^+$, i.e., the relation $\preceq_\omega$ is binary, reflexive, transitive and connected.*

We call this preorder the *ω-preorder*. We write $V =_\omega U$ if and only if $V \preceq_\omega U \wedge U \preceq_\omega V$, and $V \prec_\omega U$ if and only if $V \preceq_\omega U \wedge V \neq_\omega U$. For instance, $T_3 = \mathtt{4478} \prec_\omega \mathtt{125} = \mathrm{Rot}(T_1, 1)$ and $T_2 = \mathtt{5363} =_\omega \mathtt{6353} = \mathrm{Rot}(T_2, 2)$. Note that the latter example violates antisymmetry, i.e., the relation $\preceq_\omega$ is not a total order. In the following, we present a convenient but not necessarily optimal way to compute the ω-preorder of two given strings.

▶ **Lemma 3.2** (Weak Periodicity Lemma). *Let p and q be two periods of a string X. If $p + q \leq |X|$, then $\gcd(p, q)$ is also a period of X.*

▶ **Lemma 3.3** ([13, Lemma 5]). *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 3.2 and find that $\gcd(j, z)$ is a period of $\langle U \rangle_\mathrm{r} \cdot \langle U \rangle_\mathrm{r}$. As

i	$C_{\mathcal{T}}(i)$	$\langle C_{\mathcal{T}}(i)^{\omega}[..12]\rangle$	$\mathrm{CA}_{\mathcal{T}}^{-1}[i]$	$\mathrm{CA}_{\mathcal{T}}[i]$	$\langle C_{\mathcal{T}}(\mathrm{CA}_{\mathcal{T}}[i])^{\omega}[..12]\rangle$	$C_{\mathcal{T}}(\mathrm{CA}_{\mathcal{T}}[i])$
1	512	⌀⌀1131131131	9	8	⌀11131113111	4478
2	125	⌀11311311311	3	9	⌀11311131113	4784
3	251	⌀1⌀113113113	7	2	⌀11311311311	125
4	5363	⌀⌀1212121212	10	5	⌀12121212121	3635
5	3635	⌀12121212121	4	7	⌀12121212121	3536
6	6353	⌀⌀1212121212	11	10	⌀1⌀111311131	7844
7	3536	⌀12121212121	5	3	⌀1⌀113113113	251
8	4478	⌀11131113111	1	11	⌀⌀1113111311	8447
9	4784	⌀11311131113	2	1	⌀⌀1131131131	512
10	7844	⌀1⌀111311131	6	4	⌀⌀1212121212	5363
11	8447	⌀⌀1113111311	8	6	⌀⌀1212121212	6353

Figure 3 The conjugate array $\mathrm{CA}_{\mathcal{T}}$ of our running example $\mathcal{T} = \{512, 5363, 4478\}$.

$\gcd(j, z)$ 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 3.2 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 $\max\{\langle V^{\omega}[..3z]\rangle[x], \langle U^{\omega}[..3z]\rangle[x]\} < \text{⌀}$, 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 $\langle V^{\omega}[..3z]\rangle[x] = \text{⌀}$. Then $\langle U^{\omega}[..3z]\rangle[x] < \text{⌀}$, $x \leq |\mathrm{root}(\langle V\rangle_{\mathrm{r}})|$, 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, $\langle V^{\omega}[..3z]\rangle = \langle U^{\omega}[..3z]\rangle$. ◀

▶ **Corollary 3.4.** *Let $V, U \in \Sigma_{\$}^{+}$ and $z = \max\{|V|, |U|\}$. Then $V \prec_{\omega} U$ if and only if $\langle V^{\omega}[..3z]\rangle < \langle U^{\omega}[..3z]\rangle$.*

Similarly to Boucher et al. [2], we define the *conjugate array* $\mathrm{CA}_{\mathcal{T}}$ of $\mathcal{T}$ as the string of length n over $[1..n]$ such that $\mathrm{CA}_{\mathcal{T}}[i] = j$ if and only if

$$i - 1 = |\{k \in [1..n] \mid C_{\mathcal{T}}(k) \prec_{\omega} C_{\mathcal{T}}(j) \text{ or } C_{\mathcal{T}}(k) =_{\omega} C_{\mathcal{T}}(j) \wedge k < j\}|,$$

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 ensure 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. See Figure 3 for our running example's conjugate array. We define the *conjugate range* $\mathrm{CR}_{\mathcal{T}}(P)$ of a pattern $P \in \Sigma^{*}$ of length m in $\mathcal{T}$ as a maximal interval $[\ell..r] \subseteq [1..n]$ such that $P \approx C_{\mathcal{T}}(\mathrm{CA}_{\mathcal{T}}[i])^{\omega}[..m]$ for every $i \in [\ell..r]$. Leveraging Lemma 2.2, we find that the conjugate range is well-defined.

▶ **Corollary 3.5.** *Let $\emptyset \neq \mathcal{T} = \{T_1, ..., T_d\} \subset \Sigma_{\$}^{+}$, $n = |T_1 \cdots T_d|$, $P \in \Sigma^{*}$, and $m = |P|$. 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 have $\mathrm{CR}_{\mathcal{T}}(\varepsilon) = [1..n]$ because the empty string is a prefix of every conjugate. The computation of $\mathrm{CR}_{\mathcal{T}}(P)$ for some pattern $P \in \Sigma^*$ on indexes related to the FM-index usually perform an algorithmic technique called the *backward search*. The backward search for P in $\mathcal{T}$ processes P backwards in an online manner and refines $\mathrm{CR}_{\mathcal{T}}(P[i+1..])$ to $\mathrm{CR}_{\mathcal{T}}(P[i..])$ at the i-th step on reading $P[i]$ with $P[|P|+1..] = \epsilon$. Finally, the length of $\mathrm{CR}_{\mathcal{T}}(P)$ is the solution to Count by Corollary 3.5. For our running example, $\mathrm{CR}_{\mathcal{T}}(\mathtt{643}) = \emptyset$ and $\mathrm{CR}_{\mathcal{T}}(\mathtt{5634}) = [6..7]$. Below we define the necessary tools to allow for an efficient backward search.

3.2 LF-mapping

We want to define a map that allows us to cycle backwards through each of the texts to be indexed by mapping to the position of a conjugate in the conjugate array from the position of its left rotation. However, if we want to represent this mapping space-efficiently, we have to be careful since we used tie-breaks within texts when we sorted the conjugates by a preorder. Our idea is to relax the requirements and define a map that allows us to cycle backwards through a text that is ω-equal to the original to dodge any issues arising from our tie-breaks. For that, we want to cycle backwards in text order inside the roots of each $\langle..\rangle_{\mathrm{r}}$-encoded text. Whenever we want to move backwards at the starting position of a root, we jump to its end position. We express this backwards movement with the following permutation. For every $i \in [1..n]$ with $j = \min\{h \in [1..d] \mid \sum_{k=1}^{h} n_k \geq i\}$, let

$$\mathrm{prev}_{\mathcal{T}}(i) = \begin{cases} i - 1 + |\,\mathrm{root}(\langle T_j \rangle_{\mathrm{r}})\,| & \text{if } C_{\mathcal{T}}(i) =_{\omega} T_j, \\ i - 1 & \text{otherwise.} \end{cases}$$

For our running example, $\mathrm{prev}_{\mathcal{T}} = (1\ 3\ 2)(4\ 5)(6\ 7)(8\ 11\ 10\ 9)$. Here, we observe that we have two cycles $(4\ 5)$ and $(6\ 7)$ corresponding to $T_2[1..2] = \mathtt{53}$ and $T_2[3..4] = \mathtt{63}$, respectively, which are ω-equal to the original text $T_2 = \mathtt{5363}$. Now we are ready to express the LF-mapping in terms of the function $\mathrm{prev}_{\mathcal{T}}$. 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 $\mathrm{FL}_{\mathcal{T}}$. See Figure 5 for an example. The LF-mapping is 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, which motivates the following encoding scheme. Let the *rotational Cartesian tree signature encoding* $[\![V]\!]$ of $V \in \Sigma_{\$}^{+}$ denote a string of length $|V|$ over $\Sigma_{\$}$ such that

$$[\![V]\!][i] = \begin{cases} V[i] & \text{if } V[i] = \$, \\ \mathrm{rank}_{\mathtt{x}}(\langle \mathrm{Rot}(V,i) \rangle, |V|) - \mathrm{rank}_{\mathtt{x}}(\langle V[i] \cdot \mathrm{Rot}(V,i) \rangle[2..], |V|) & \text{otherwise,} \end{cases}$$

for each $i \in [1..|V|]$, i.e., if $V[i] \neq \$$, then $[\![V]\!][i]$ reports the number of positions $j \in [1..|V|]$ such that $\mathrm{Rot}(V,i)[j] \geq V[i]$ and $\langle \mathrm{Rot}(V,i) \rangle[j] = \mathtt{x}$. See Figure 4 for an example. Note that $[\![V]\!][i..\mathrm{select}_{\$}(V,1)] = [\![V[i..\mathrm{select}_{\$}(V,1)]]\!]$ for each $V \in \Sigma_{\$}^{+}$ satisfying $\mathrm{rank}_{\$}(V,|V|) \geq 1$ and $i \in [1..\mathrm{select}_{\$}(V,1)]$.

▶ **Lemma 3.6.** *Let $V \in \Sigma_{\$}^{+}$. Then $\sum_{i=1}^{j} [\![V]\!][i] \leq |V|$, where $j = \mathrm{select}_{\$}(V,1) - 1$ if $\mathrm{rank}_{\$}(V,|V|) \geq 1$, and $j = |V|$ otherwise.*

Taking advantage of both encodings, we investigate how the ω-preorder of two strings changes if we rotate them. For convenience, we borrowed the following notation from literature [14, 11]. Let $\pi(V) = [\![V]\!][1]$ for every $V \in \Sigma_{\$}^{+}$, and $\mathrm{lcp}^{\infty}(U,W) = \mathrm{rank}_{\mathtt{x}}(\langle U \rangle, \mathrm{lcp}(\langle U \rangle, \langle W \rangle))$ for each $U, W \in \Sigma_{\$}^{*}$. For example, $\pi(\mathtt{4478}) = 1$ and $\mathrm{lcp}^{\infty}(\mathtt{4478}, \mathtt{7844}) = 1$.

i	$T_3[i]$	$\mathrm{Rot}(T_3, i)$	$\langle \mathrm{Rot}(T_3, i) \rangle$	$\langle T_3[i] \cdot \mathrm{Rot}(T_3, i) \rangle$	$[\![T_3]\!][i]$
1	4	4784	𝔵113	𝔵1113	1
2	4	7844	𝔵1𝔵1	𝔵1131	2
3	7	8447	𝔵𝔵11	𝔵1𝔵11	1
4	8	4478	𝔵111	𝔵𝔵111	0

Figure 4 Rotational Cartesian tree signature encoding $[\![T_3]\!]$ of $T_3 = 4478$.

▶ **Lemma 3.7** ([17, Lemma 3]). *Let $V, U \in \Sigma_{\$}^{+}$ such that $\mathrm{Rot}(V,1) \prec_\omega \mathrm{Rot}(U,1)$. Then $V \prec_\omega U$ if and only if (a) $\pi(V) = \$$ or (b) $\pi(V) \geq \min\{e, \pi(U)\}$ and $\pi(U) \neq \$$, where $e = \mathrm{lcp}^\infty(\mathrm{Rot}(V,1), \mathrm{Rot}(U,1))$.*

Proof. We show the statement for $\min\{\pi(V), \pi(U)\} \in \Sigma$ since it is non-trivial. Let $z = \max\{|V|, |U|\}$ and $\lambda = \mathrm{lcp}(\langle \mathrm{Rot}(V,1)^\omega[..3z] \rangle, \langle \mathrm{Rot}(U,1)^\omega[..3z] \rangle)$. Then $\langle \mathrm{Rot}(V,1)^\omega[..3z] \rangle < \langle \mathrm{Rot}(U,1)^\omega[..3z] \rangle$ by Corollary 3.4, $\lambda < 3z$, and $\mathrm{select}_{\mathbb{x}}(\langle \mathrm{Rot}(V,1) \rangle, e) \leq \min\{|V|, |U|\}$.

(⇒) Assume $\pi(V) < \min\{e, \pi(U)\}$. Let $i = \mathrm{select}_{\mathbb{x}}(\langle V \rangle, 2)$. Then $\langle V \rangle[..i-1] = \langle U \rangle[..i-1]$ and $\langle V \rangle[i] = \mathbb{x} \neq i - 1 = \langle U \rangle[i]$, i.e. $\langle U \rangle < \langle V \rangle$. Consequently, $\langle U^\omega[..3z] \rangle < \langle V^\omega[..3z] \rangle$, which implies $U \prec_\omega V$ by Corollary 3.4. The statement follows by contraposition.

(⇐) We exhaust all possible cases for $\pi(V) \geq \min\{e, \pi(U)\}$.

Case 1. Assume $\min\{e, \pi(V)\} > \pi(U)$. Let $i = \mathrm{select}_{\mathbb{x}}(\langle U \rangle, 2)$. Then $\langle V \rangle[..i-1] = \langle U \rangle[..i-1]$ and $\langle V \rangle[i] = i - 1 \neq \mathbb{x} = \langle U \rangle[i]$, i.e. $\langle V \rangle < \langle U \rangle$. Consequently, $\langle V^\omega[..3z] \rangle < \langle U^\omega[..3z] \rangle$, which implies $V \prec_\omega U$ by Corollary 3.4.

Case 2. Assume $\min\{\pi(V), \pi(U)\} \geq e$. Then we have $\langle V^\omega[..3z] \rangle[..\lambda+1] = \langle U^\omega[..3z] \rangle[..\lambda+1]$. Since $\mathrm{Rot}(V,1) \prec_\omega \mathrm{Rot}(U,1)$, $\lambda + 2 \leq 3z$ by Lemma 3.3. If $\langle \mathrm{Rot}(U,1)^\omega[..3z] \rangle[\lambda+1] = \mathbb{x}$, $\langle V^\omega[..3z] \rangle[\lambda+2] \leq \lambda < \lambda + 1 = \langle U^\omega[..3z] \rangle[\lambda+2]$. If $\langle \mathrm{Rot}(U,1)^\omega[..3z] \rangle[\lambda] \neq \mathbb{x}$, then $\langle V^\omega[..3z] \rangle[\lambda+2] = \langle \mathrm{Rot}(V,1)^\omega[..3z] \rangle[\lambda+1] < \langle \mathrm{Rot}(U,1)^\omega[..3z] \rangle[\lambda+1] = \langle U^\omega[..3z] \rangle[\lambda+2]$. Thus, $V \prec_\omega U$ by Corollary 3.4.

Case 3. Assume $e > \pi(V) = \pi(U)$. Then $\langle V^\omega[..3z] \rangle[..\lambda+1] = \langle U^\omega[..3z] \rangle[..\lambda+1]$ and $\lambda \leq z$. Moreover, $\langle V^\omega[..3z] \rangle[\lambda+2] = \langle \mathrm{Rot}(V,1)^\omega[..3z] \rangle[\lambda+1] < \langle \mathrm{Rot}(U,1)^\omega[..3z] \rangle[\lambda+1] = \langle U^\omega[..3z] \rangle[\lambda+2]$. Hence, $V \prec_\omega U$ by Corollary 3.4. ◀

Our representation of the LF- and FL-mapping consists of two strings $\mathrm{L}_{\mathcal{T}}$ and $\mathrm{F}_{\mathcal{T}}$, which are defined as follows. First, $\mathrm{L}_{\mathcal{T}}$ is the string of length n over $\Sigma_{\$}$ such that $\mathrm{L}_{\mathcal{T}}[i] = \pi(C_{\mathcal{T}}(\mathrm{CA}_{\mathcal{T}}[\mathrm{LF}_{\mathcal{T}}[i]])) = [\![C_{\mathcal{T}}(\mathrm{CA}_{\mathcal{T}}[i])]\!][|C_{\mathcal{T}}(\mathrm{CA}_{\mathcal{T}}[i])|]$ for each $i \in [1..n]$.

▶ **Corollary 3.8.** *Let $\emptyset \neq \mathcal{T} \subset \Sigma_{\$}^{+}$, n the accumulated length of all texts in $\mathcal{T}$, $i, j \in [1..n]$, and $i < j$. If $\mathrm{L}_{\mathcal{T}}[i] = \mathrm{L}_{\mathcal{T}}[j]$, then $\mathrm{LF}_{\mathcal{T}}[i] < \mathrm{LF}_{\mathcal{T}}[j]$.*

Second, $\mathrm{F}_{\mathcal{T}}$ is the string of length n over $\Sigma_{\$}$ such that $\mathrm{F}_{\mathcal{T}}[\mathrm{LF}_{\mathcal{T}}[i]] = \mathrm{L}_{\mathcal{T}}[i]$ for each $i \in [1..n]$. By what follows, $\mathrm{L}_{\mathcal{T}}$ and $\mathrm{F}_{\mathcal{T}}$ suffice to compute both LF- and FL-mapping of $\mathcal{T}$.

▶ **Corollary 3.9.** *Let $\emptyset \neq \mathcal{T} \subset \Sigma_{\$}^{+}$, n the accumulated length of all texts in $\mathcal{T}$, and $i \in [1..n]$. Then $\mathrm{LF}_{\mathcal{T}}[i] = \mathrm{select}_{\mathrm{L}_{\mathcal{T}}[i]}(\mathrm{F}_{\mathcal{T}}, \mathrm{rank}_{\mathrm{L}_{\mathcal{T}}[i]}(\mathrm{L}_{\mathcal{T}}, i))$ and $\mathrm{FL}_{\mathcal{T}}[i] = \mathrm{select}_{\mathrm{F}_{\mathcal{T}}[i]}(\mathrm{L}_{\mathcal{T}}, \mathrm{rank}_{\mathrm{F}_{\mathcal{T}}[i]}(\mathrm{F}_{\mathcal{T}}, i))$.*

In Figure 5 we present $\mathrm{F}_{\mathcal{T}}$ and $\mathrm{L}_{\mathcal{T}}$ of our running example.

▶ Remark 3.10. Let $d = 1$, $\mathrm{rank}_{\$}(T_1, n_1) = 1$ and $T_1[n_1] = \$$. If we substitute the occurrence of $\$$ in $\mathrm{F}_{\mathcal{T}}$ and $\mathrm{L}_{\mathcal{T}}$ for -1, then the modified strings constitute the integer-based representation of Kim and Cho's index [17, Section 4].

Algorithm 1 Computing the conjugate range $\mathrm{CR}_{\mathcal{T}}(P[i..])$. Here, $\emptyset \neq \mathcal{T} \subset \Sigma_{\$}^{+}$, $P \in \Sigma^{+}$, $m = |P|$, $i \in [1..m]$, $[\ell..r] = \mathrm{CR}_{\mathcal{T}}(P[i{+}1..])$, $h = \pi(P[i..]{\cdot}\$)$, and $e = \mathrm{rank}_{\infty}(\langle P[i..]\rangle, m{-}i{+}1)$.

```
1  Function crangeupd(e, h, [ℓ..r], L_T, F_T, LCP^∞_T):
2      r' ← r;
3      if e > 1 then                                          // Case 1 in Lemma 3.13
4          if c ← rnkcnt_{L_T}(ℓ, r, h, h) ≥ 1 then
5              r' ← LF_T[select_h(L_T, rank_h(L_T, r))];
6      else                                                   // Case 2 in Lemma 3.13
7          if c ← rnkcnt_{L_T}(ℓ, r, h, σ) ≥ 1 then
8              v ← RNV_{L_T}(ℓ, r, h − 1);
9              x ← select_v(L_T, rank_v(L_T, r));
10             [ℓ''..r''] ← MI_{LCP^∞_T}(x, v + 1);
11             y ← rnkcnt_{L_T}(ℓ, x − 1, h, σ) + rnkcnt_{L_T}(x + 1, r'', h, σ);
12             r' ← LF_T[x] + c − (y + 1);
13     return [r' − c + 1..r'];                               // returns [r + 1..r] = ∅ if c = 0
```

3.3 Backward Search

The LF-mapping can be leveraged for the backward search by the following result, which is due to Lemma 2.2 and Corollary 3.5.

▶ **Lemma 3.11** ([17, Lemma 6]). *Let $\emptyset \neq \mathcal{T} \subset \Sigma_{\$}^{+}$, $P \in \Sigma^{+}$, $|P| = m$, $i \in [1..m]$, $h = \pi(P[i..]\cdot\$)$ and $e = \mathrm{rank}_{\$}(\langle P[i..]\rangle, m-i+1)$. For $j \in \mathrm{CR}_{\mathcal{T}}(P[i+1..])$, $\mathrm{LF}_{\mathcal{T}}[j] \in \mathrm{CR}_{\mathcal{T}}(P[i..])$ if and only if (a) $e > 1$ and $\mathrm{L}_{\mathcal{T}}[j] = h$ or (b) $e = 1$ and $\mathrm{L}_{\mathcal{T}}[j] \geq h$.*

At this point it is straight-forward to apply the techniques developed by Kim and Cho [17, Sections 5 and 6] to define a static index of $\mathcal{T} \subset \Sigma^{+}$ that occupies $3n + o(n)$ bits of space and that solves Count in $O(m)$ time, where m is the pattern length. However, for brevity and in view of a space efficient construction of our proposed index and its extension, we will represent $\mathrm{F}_{\mathcal{T}}$ and $\mathrm{L}_{\mathcal{T}}$ by the dynamic data structure of Lemma 2.1, and introduce an auxiliary string for the backward search. Let $\mathrm{LCP}_{\mathcal{T}}^{\infty}$ denote a string of length n over Σ such that $\mathrm{LCP}_{\mathcal{T}}^{\infty}[1] = 0$ and $\mathrm{LCP}_{\mathcal{T}}^{\infty}[i] = \mathrm{lcp}^{\infty}(C_{\mathcal{T}}(\mathrm{CA}_{\mathcal{T}}[i]), C_{\mathcal{T}}(\mathrm{CA}_{\mathcal{T}}[i-1]))$ for each $i \in [2..n]$.

▶ **Lemma 3.12.** *Let $\emptyset \neq \mathcal{T} = \{T_1, ..., T_d\} \subset \Sigma_{\$}^{+}$, $n = |T_1 \cdots T_d|$, $i, j \in [1..n]$, and $i < j$. Then $\mathrm{lcp}^{\infty}(C_{\mathcal{T}}(\mathrm{CA}_{\mathcal{T}}[i]), C_{\mathcal{T}}(\mathrm{CA}_{\mathcal{T}}[j])) = \min\{\mathrm{LCP}_{\mathcal{T}}^{\infty}[k] \mid k \in [i+1..j]\} = \mathrm{RNV}_{\mathrm{LCP}_{\mathcal{T}}^{\infty}}(i+1, j, -1)$.*

▶ **Lemma 3.13.** *Let $\emptyset \neq \mathcal{T} \subset \Sigma_{\$}^{+}$, $P \in \Sigma^{+}$, $m = |P|$, $i \in [1..m]$, $[\ell..r] = \mathrm{CR}_{\mathcal{T}}(P[i+1..])$, $h = \pi(P[i..]\cdot\$)$, and $e = \mathrm{rank}_{\$}(\langle P[i..]\rangle, m-i+1)$. Then Algorithm 1 correctly computes $[\ell'..r'] = \mathrm{CR}_{\mathcal{T}}(P[i..])$.*

Proof. Let $c = |\mathrm{CR}_{\mathcal{T}}(P[i..])|$. We show how Algorithm 1 computes c and r' to obtain $[\ell'..r']$.

Case 1. Assume $e > 1$. This case is handled from Line 3 through Line 5. By Lemma 3.11, $\mathrm{LF}_{\mathcal{T}}[j] \in \mathrm{CR}_{\mathcal{T}}(P[i..])$ if and only if $\mathrm{L}_{\mathcal{T}}[j] = h$ for each $j \in [\ell..r]$. We compute c in Line 4 and return an empty interval in Line 13 due to Line 2 if $c \leq 0$. Assume $c \geq 1$. We compute the largest $x \in [\ell..r]$ such that $\mathrm{L}_{\mathcal{T}}[x] = h$, and then $r' = \mathrm{LF}_{\mathcal{T}}[x]$ in Line 5, with correctness following by Corollary 3.8.

i	$\mathrm{CA}_{\mathcal{T}}[i]$	$C_{\mathcal{T}}(\mathrm{CA}_{\mathcal{T}}[i])$	$\mathrm{LF}_{\mathcal{T}}[i]$	$\mathrm{F}_{\mathcal{T}}[i]$	$\mathrm{L}_{\mathcal{T}}[i]$	$\mathrm{LCP}^{\infty}_{\mathcal{T}}[i]$	$\langle C_{\mathcal{T}}(\mathrm{CA}_{\mathcal{T}}[i])\rangle$
1	8	4478	8	1	0	0	𝔵111
2	9	4784	1	2	1	1	𝔵113
3	2	125	9	2	0	1	𝔵11
4	5	3635	10	2	0	1	𝔵121
5	7	3536	11	2	0	1	𝔵121
6	10	7844	2	1	2	1	𝔵1𝔵1
7	3	251	3	1	2	2	𝔵1𝔵
8	11	8447	6	0	1	1	𝔵𝔵11
9	1	512	7	0	1	2	𝔵𝔵1
10	4	5363	4	0	2	2	𝔵𝔵12
11	6	6353	5	0	2	2	𝔵𝔵12

Figure 5 The cBWT index of our running example $\mathcal{T} = \{\mathtt{512}, \mathtt{5363}, \mathtt{4478}\}$.

Case 2. Assume $e = 1$. This case is handled from Line 6 through Line 12. By Lemma 3.11, $\mathrm{LF}_{\mathcal{T}}[j] \in \mathrm{CR}_{\mathcal{T}}(P[i..])$ if and only if $\mathrm{L}_{\mathcal{T}}[j] \geq h$ for each $j \in [\ell..r]$. Thus, we correctly compute c in Line 7, and return an empty interval in Line 13 due to Line 2 if $c \leq 0$. Assume $c \geq 1$. We compute the lowest value v in $\mathrm{L}_{\mathcal{T}}[\ell..r]$ greater than $h-1$ in Line 8 and determine the largest $x \in [\ell..r]$ such that $\mathrm{L}_{\mathcal{T}}[x] = v$ in Line 9. Let $[\ell''..r'']$ denote the interval computed in Line 10. Then $\mathrm{lcp}^{\infty}(C_{\mathcal{T}}(\mathrm{CA}_{\mathcal{T}}[j]), C_{\mathcal{T}}(\mathrm{CA}_{\mathcal{T}}[k])) \geq v+1$ for each $j, k \in [\ell''..r'']$ by Lemma 3.12. We compute $y = \mathrm{LF}_{\mathcal{T}}[x] - \ell'$. The following statements hold due to Lemma 3.7.

- If $j \in [\ell..x-1]$ satisfies $\mathrm{L}_{\mathcal{T}}[j] \geq h$, then $\mathrm{LF}_{\mathcal{T}}[j] < \mathrm{LF}_{\mathcal{T}}[x]$ since $v \leq \mathrm{L}_{\mathcal{T}}[j]$.
- If $j \in [x+1..r'']$ satisfies $\mathrm{L}_{\mathcal{T}}[j] \geq h$, then $\mathrm{LF}_{\mathcal{T}}[j] < \mathrm{LF}_{\mathcal{T}}[x]$ since $v < \mathrm{L}_{\mathcal{T}}[j]$ and $v < v+1 \leq \mathrm{lcp}^{\infty}(C_{\mathcal{T}}(\mathrm{CA}_{\mathcal{T}}[j]), C_{\mathcal{T}}(\mathrm{CA}_{\mathcal{T}}[x]))$.
- If $j \in [r''+1..r]$ satisfies $\mathrm{L}_{\mathcal{T}}[j] \geq h$, then $\mathrm{LF}_{\mathcal{T}}[j] > \mathrm{LF}_{\mathcal{T}}[x]$ since we have $v \geq \mathrm{lcp}^{\infty}(C_{\mathcal{T}}(\mathrm{CA}_{\mathcal{T}}[j]), C_{\mathcal{T}}(\mathrm{CA}_{\mathcal{T}}[x]))$.

We apply these results to compute y in Line 11, and then infer $r' = \mathrm{LF}_{\mathcal{T}}[x] + c - (y+1)$ in Line 12. ◀

The next result is obtained from the computation of Demaine and colleagues' [3] Cartesian tree signature encoding of the given string.

▶ **Lemma 3.14.** *Given $P \in \Sigma^*_{\$}$ of length m, we can process P in $O(m)$ time such that we can subsequently compute $\pi(P[i..] \cdot \$)$ and $\mathrm{rank}_{\mathtt{𝔵}}(P[i..], m-i+1)$ in $O(1)$ time, for every $i \in [1..m]$.*

▶ **Theorem 3.15.** *Let $\emptyset \neq \mathcal{T} = \{T_1, ..., T_d\} \subset \Sigma^+_{\$}$ and $n = |T_1 \cdots T_d|$. There exists a data structure occupying $O(n \lg \sigma)$ bits of space that solves* Count *in $O(m \frac{\lg \sigma \lg n}{\lg \lg n})$ time, where m is the pattern length.*

Proof. We represent $\mathrm{F}_{\mathcal{T}}$, $\mathrm{L}_{\mathcal{T}}$ and $\mathrm{LCP}^{\infty}_{\mathcal{T}}$ by the data structure of Lemma 2.1, which leads to the claimed space complexity. We preprocess a pattern $P \in \Sigma^+$ of length m with Lemma 3.14, and compute $\mathrm{CR}_{\mathcal{T}}(P[i..])$ from $\mathrm{CR}_{\mathcal{T}}(P[i+1..])$ for each $i \in [1..m]$ in descending order leveraging Lemma 3.13. Since each conjugate range update takes $O(1)$ queries, the claimed complexity for solving Count follows from Lemma 2.1. ◀

We call the data structure of Theorem 3.15 the *cBWT index* of $\mathcal{T}$. The cBWT index of the running example is presented in Figure 5.

4 Construction in $O(n \lg \sigma)$ Bits of Space

We will show how to construct the cBWT index of a single text, and how an existing cBWT index can be extended to also index an additional text. We then leverage these two results to iteratively construct the cBWT index of any set of texts, adding one new text per iteration step.

4.1 Single Text cBWT Index

Before we can tackle the construction of the cBWT index of a single text, we need another technical result, which follows from an examination of the definitions and Lemma 3.7.

▶ **Lemma 4.1.** *Let* $V, U \in \Sigma_{\$}^{+}$, $\mathrm{Rot}(V,1) \prec_\omega \mathrm{Rot}(U,1)$, *and* $e = \mathrm{lcp}^\infty(\mathrm{Rot}(V,1), \mathrm{Rot}(U,1))$. *Then*

$$\mathrm{lcp}^\infty(V,U) = \begin{cases} 0 & \text{if } \pi(U) = \$ \text{ or } \pi(V) = \$, \\ e - \pi(V) + 1 & \text{if } V \prec_\omega U \text{ and } \$ \neq \pi(V) = \pi(U) < e, \\ 1 & \text{otherwise.} \end{cases}$$

We will first show how to construct the cBWT index of a text from $\Sigma_{\$}^{+}$ in which \$ occurs exactly once, and then show how to construct the cBWT index of an arbitrary text from Σ^+ (without the requirement on \$).

▶ **Lemma 4.2.** *Let* $R \in \Sigma_{\$}^{+}$, $\rho = |R| \geq 2$, $R[\rho] = \$$, *and* $\mathrm{rank}_\$(R,\rho) = 1$. *Algorithm 2 correctly computes* $\mathrm{CA}^{-1}_{\{bR\}}[1]$ *and the cBWT index of* $\{bR\}$ *for each* $b \in \Sigma$.

Proof. For each $i \in [0..\pi(bR)]$, let $J_i = [\ell_i..r_i]$ maximal such that $r_i \leq \mathrm{CA}^{-1}_{\{R\}}[1]$ and $\mathrm{lcp}^\infty(R, C_{\{R\}}(\mathrm{CA}_{\{R\}}[j])) = i$ for each $j \in J_i$, and $J_{\pi(bR)+1} = [\ell_{\pi(bR)+1}..r_{\pi(bR)+1}]$ maximal such that $\mathrm{lcp}^\infty(R, C_{\{R\}}(\mathrm{CA}_{\{R\}}[j])) \geq \pi(bR)+1$ for each $j \in J_{\pi(bR)+1}$, i.e., the left and right boundary of the former are computed in Line 6 and Line 5, respectively, and the latter in Line 2. The following statements are due to the location of the single \$ in each conjugate of R and Lemma 3.7.

- If $j \in [r_{\pi(bR)+1} + 1..\rho]$, then $bR \prec_\omega C_{\{R\}}(\mathrm{CA}_{\{R\}}[\mathrm{LF}_{\{R\}}[j]])$.
- If $j \in [\mathrm{CA}^{-1}_{\{R\}}[1]+1..r_{\pi(bR)+1}]$, then $C_{\{R\}}(\mathrm{CA}_{\{R\}}[\mathrm{LF}_{\{R\}}[j]]) \prec_\omega bR$ if and only if $\mathrm{L}_{\{R\}}[j] \geq \pi(bR)+1$.
- If $j = \mathrm{CA}^{-1}_{\{R\}}[1]$, then $C_{\{R\}}(\mathrm{CA}_{\{R\}}[\mathrm{LF}_{\{R\}}[j]]) = (\$ \cdot R)[..\rho] \prec_\omega bR$.
- If $j \in [\ell_{\pi(bR)+1}..\mathrm{CA}^{-1}_{\{R\}}[1]-1]$, then $C_{\{R\}}(\mathrm{CA}_{\{R\}}[\mathrm{LF}_{\{R\}}[j]]) \prec_\omega bR$ if and only if $\mathrm{L}_{\{R\}}[j] \geq \pi(bR)$.
- If $i \in [0..\pi(bR)]$ and $j \in J_i$, then $C_{\{R\}}(\mathrm{CA}_{\{R\}}[\mathrm{LF}_{\{R\}}[j]]) \prec_\omega bR$ if and only if $\mathrm{L}_{\{R\}}[j] \geq i$.

We apply these results from Line 2 through Line 6 to compute the number c of conjugates of R *smaller* than bR according to ω-preorder. Since $C_{\{R\}}(k)[..\mathrm{select}_\$(C_{\{R\}}(k),1)] = C_{\{bR\}}(k+1)[..\mathrm{select}_\$(C_{\{R\}}(k),1)]$ for each $k \in [1..\rho]$ by assumption on R and b,

$$C_{\{bR\}}(\mathrm{CA}_{\{bR\}}[i])[..x] = \begin{cases} C_{\{R\}}(\mathrm{CA}_{\{R\}}[i])[..x] & \text{if } i \in [1..c], \\ bR & \text{if } i = c+1, \\ C_{\{R\}}(\mathrm{CA}_{\{R\}}[i-1])[..x] & \text{otherwise,} \end{cases}$$

for each $i \in [1..\rho+1]$ with $x = \mathrm{select}_\$(C_{\{bR\}}(\mathrm{CA}_{\{bR\}}[i]),1)$. Thus, c is also the number of conjugates of bR smaller than bR according to ω-preorder. Subsequently, Algorithm 2

Algorithm 2 Computing $c+1 = \mathrm{CA}^{-1}_{\{bR\}}[1]$ and updating the cBWT of $\{R\}$ to that of $\{bR\}$ for $b \in \Sigma$. Here, $R \in \Sigma_{\$}^{+}$, $\rho = |R|$, $R[\rho] = \$$, $\mathrm{rank}_{\$}(R, \rho) = 1$, and $y = \mathrm{CA}^{-1}_{\{R\}}[1]$.

```
1  Function extend(π(bR), y, F_{R}, L_{R}, LCP^∞_{R}):
2      [ℓ..r] ← MI_{LCP^∞_{R}}(y, π(bR) + 1);
3      c ← rnkcnt_{L_{R}}(ℓ, y − 1, π(bR), σ) + 1 + rnkcnt_{L_{R}}(y + 1, r, π(bR) + 1, σ);
4      for i ← π(bR) to 0 do
5          r′ ← ℓ − 1;
6          [ℓ..r] ← MI_{LCP^∞_{R}}(y, i);
7          c ← c + rnkcnt_{L_{R}}(ℓ, r′, i, σ);
8      p, s ← 1;          // p stores LCP^∞_{bR}[c+1], and, if c < ρ, s stores LCP^∞_{bR}[c+2]
9      if FL_{R}[c] < y and π(bR) = F_{R}[c] then
10         p ← RNV_{LCP^∞_{R}}(FL_{R}[c] + 1, y, −1) − π(bR) + 1;
11     if c < ρ and FL_{R}[c+1] > y and π(bR) = F_{R}[c+1] then
12         s ← RNV_{LCP^∞_{R}}(y + 1, FL_{R}[c+1], −1) − π(bR) + 1;
13     insert_{LCP^∞_{R}}(c + 1, p);
14     if c < ρ then LCP^∞_{R}[c+2] ← s ;
15     insert_{F_{R}}(c + 1, π(bR));
16     L_{R}[y] ← π(bR);
17     insert_{L_{R}}(c + 1, $);
18     return c + 1, F_{R}, L_{R}, LCP^∞_{R};        // CA^{-1}_{bR}[1] and the cBWT index of {bR}
```

updates the cBWT index of $\{R\}$ from Line 8 through Line 17. By assumption on b and R, $\mathrm{LCP}^{\infty}_{\{R\}}[..c] = \mathrm{LCP}^{\infty}_{\{bR\}}[..c]$ and $\mathrm{LCP}^{\infty}_{\{R\}}[c+2..] = \mathrm{LCP}^{\infty}_{\{bR\}}[c+3..]$. From Line 8 through Line 14 we leverage Lemma 3.12 and Lemma 4.1 to compute $\mathrm{LCP}^{\infty}_{\{bR\}}[c+1]$ and, if $c < \rho$, $\mathrm{LCP}^{\infty}_{\{bR\}}[c+2]$, and update $\mathrm{LCP}^{\infty}_{\{R\}}$. By assumption on R, $\pi(C_{\{R\}}(j)) = \pi(C_{\{bR\}}(j+1))$ for each $j \in [1..\rho]$. Consequently, $\mathrm{F}_{\{R\}}[..c] = \mathrm{F}_{\{bR\}}[..c]$ and $\mathrm{F}_{\{R\}}[c+1..] = \mathrm{F}_{\{bR\}}[c+2..]$. Moreover, if we set $\mathrm{L}_{\{R\}}[\mathrm{CA}^{-1}_{\{R\}}[1]] = \pi(bR)$, then $\mathrm{L}_{\{R\}}[..c] = \mathrm{L}_{\{bR\}}[..c]$ and $\mathrm{L}_{\{R\}}[c+1..] = \mathrm{L}_{\{bR\}}[c+2..]$. It remains to compute $\mathrm{F}_{\{bR\}}[c+1]$ and $\mathrm{L}_{\{bR\}}[c+1]$, which are $\pi(bR)$ and \$, respectively. The update of both $\mathrm{F}_{\{R\}}$ and $\mathrm{L}_{\{R\}}$ is done from Line 15 through Line 17. ◀

▶ **Corollary 4.3.** *Let $R \in \Sigma_{\$}^{+}$, $\rho = |R|$, $R[\rho] = \$$, and $\mathrm{rank}_{\$}(R, \rho) = 1$. The cBWT index of $\{R\}$ can be constructed in $O(\rho \lg \sigma)$ bits of space and $O(\rho \frac{\lg \sigma \lg \rho}{\lg \lg \rho})$ time.*

Proof. We show the case where $\rho \geq 2$. We represent $\mathrm{F}_{\{R[\rho-1..]\}} = \$0$, $\mathrm{L}_{\{R[\rho-1..]\}} = 0\$$ and $\mathrm{LCP}^{\infty}_{\{R[\rho-1..]\}} = 00$ by the data structure of Lemma 2.1. Initially, $\mathrm{CA}^{-1}_{\{R[\rho-1..]\}}[1] = 2$. We preprocess R with Lemma 3.14 in $O(\rho)$ time to have access to $\pi(R[i..]) = \pi(\mathrm{Rot}(R, i-1))$ for each $i \in [1..\rho]$ in $O(1)$ time. For each $i \in [1..\rho-2]$ in descending order, we apply Algorithm 2 to compute $\mathrm{CA}^{-1}_{\{R[i..]\}}[1]$ and extend the cBWT index of $\{R[i+1..]\}$ to that of $\{R[i..]\}$. Since the extension of the cBWT index of $\{R[i+1..]\}$ to that of $\{R[i..]\}$ takes $O(\pi(R[i..]))$ queries for each $i \in [1..\rho-2]$, the construction of the cBWT index of $\{R[1..]\} = \{R\}$ takes a total of $O(\rho)$ queries by Lemma 3.6. The claimed complexities then follow by Lemma 2.1. ◀

▶ Remark 4.4. Due to Remark 3.10, the previous statement yields the integer-based representation of Kim and Cho's index [17, Section 4] by substituting \$ with -1 in both $\mathrm{F}_{\mathcal{T}}$ and $\mathrm{L}_{\mathcal{T}}$. Thus, we can construct their compact index [17, Section 5] directly [17, Section A.4] while retaining the complexities as stated in Corollary 4.3.

▶ **Corollary 4.5.** *Let $T \in \Sigma^+$ and $|T| = n$. Then the cBWT index of $\{T\}$ can be constructed in $O(n \lg \sigma)$ bits of space and $O(n \frac{\lg \sigma \lg n}{\lg \lg n})$ time.*

Proof. Let $R = T^4\$$. We construct the cBWT index of $\{R\}$ within the claimed complexities by Corollary 4.3. During construction, we build a bit string Y of length $4n+1$ such that $Y[i] = 1$ if and only if $\mathrm{CA}_{\{R\}}[i] \in [2..n+1]$. By choice of R, $C_{\{T\}}(i)^\omega[..3n] = C_{\{R\}}(i)[..3n]$ for each $i \in [1..n]$, and $C_{\{T\}}(1)^\omega[..3n] = C_{\{R\}}(n+1)[..3n]$. The following statements hold for each $i \in [1..n]$ by choice of Y and Lemma 3.3.

- $C_{\{T\}}(\mathrm{CA}_{\{T\}}[i])^\omega[..3n] = C_{\{R\}}(\mathrm{CA}_{\{R\}}[\mathrm{select}_1(Y,i)])[..3n]$.
- $\pi(C_{\{T\}}(\mathrm{CA}_{\{T\}}[i])) = \pi(C_{\{R\}}(\mathrm{CA}_{\{R\}}[\mathrm{select}_1(Y,i)]))$.
- $\pi(C_{\{T\}}(\mathrm{CA}_{\{T\}}[\mathrm{LF}_{\{T\}}[i]])) = \pi(C_{\{R\}}(\mathrm{CA}_{\{R\}}[\mathrm{LF}_{\{R\}}[\mathrm{select}_1(Y,i)]]))$.

Thus, $\mathrm{F}_{\{T\}}[i] = \mathrm{F}_{\{R\}}[\mathrm{select}_1(Y,i)]$ and $\mathrm{L}_{\{T\}}[i] = \mathrm{L}_{\{R\}}[\mathrm{select}_1(Y,i)]$ for each $i \in [1..n]$. Moreover, $\mathrm{LCP}^\infty_{\{T\}}[i] = \mathrm{RNV}_{\mathrm{LCP}^\infty_{\{R\}}}(\mathrm{select}_1(Y,i-1)+1, \mathrm{select}_1(Y,i), -1)$ for each $i \in [2..n]$ due to Lemma 3.12, and $\mathrm{LCP}^\infty_{\{T\}}[1] = 0$. Hence, we can transform the cBWT index of $\{R\}$ to index $\{T\}$ with $O(n)$ queries and operations. The claimed complexities now follow by Lemma 2.1. ◀

4.2 Extending an Existing cBWT Index

Let $\emptyset \neq \mathcal{R} \subset \Sigma^+$ be a non-empty set of strings and $\rho = |\mathrm{CA}_\mathcal{R}|$ denote the accumulated length of all strings in $\mathcal{R}$. In this subsection, we assume we have the cBWT index of $\mathcal{R}$ at hand, and want to extend it by another text $S \in \Sigma^+$ of length λ to index $\mathcal{S} = \mathcal{R} \cup \{S\}$. To achieve this, we compute the integers $\mathrm{cnt}_\mathcal{R}(\mathrm{Rot}(S,i))$, $\mathrm{plcp}^\infty_\mathcal{R}(\mathrm{Rot}(S,i))$ and $\mathrm{slcp}^\infty_\mathcal{R}(\mathrm{Rot}(S,i))$ for each $i \in [1..\lambda]$, whose definitions follow. Let $\mathrm{cnt}_\mathcal{R}(V) = |\{i \in [1..\rho] \mid C_\mathcal{R}(\mathrm{CA}_\mathcal{R}[i]) \preceq_\omega V\}|$,

$$\mathrm{plcp}^\infty_\mathcal{R}(V) = \begin{cases} -1 & \text{if } \mathrm{cnt}_\mathcal{R}(V) = 0, \\ \mathrm{lcp}^\infty(C_\mathcal{R}(\mathrm{CA}_\mathcal{R}[\mathrm{cnt}_\mathcal{R}(V)]), V) & \text{otherwise, and} \end{cases}$$

$$\mathrm{slcp}^\infty_\mathcal{R}(V) = \begin{cases} \mathrm{lcp}^\infty(C_\mathcal{R}(\mathrm{CA}_\mathcal{R}[\mathrm{cnt}_\mathcal{R}(V)+1]), V) & \text{if } \mathrm{cnt}_\mathcal{R}(V) < \rho, \\ -1 & \text{otherwise,} \end{cases}$$

for each $V \in \Sigma^+_\$$. We can apply the techniques exhibited in Lemma 4.2 to compute the helper values.

▶ **Lemma 4.6.** *Let $\emptyset \neq \mathcal{R} \subset \Sigma^+$, $\rho = |\mathrm{CA}_\mathcal{R}|$, and $V \in \Sigma^+_\$$ with $\mathrm{rank}_\$(V, |V|) \geq 1$. Then Algorithm 3 correctly computes $\mathrm{cnt}_\mathcal{R}(V)$, $\mathrm{plcp}^\infty_\mathcal{R}(V)$ and $\mathrm{slcp}^\infty_\mathcal{R}(V)$.*

Proof. Algorithm 3 takes $\pi(V)$, $y = \mathrm{cnt}_\mathcal{R}(\mathrm{Rot}(V,1))$, $\mathrm{plcp}^\infty_\mathcal{R}(\mathrm{Rot}(V,1))$, $\mathrm{slcp}^\infty_\mathcal{R}(\mathrm{Rot}(V,1))$, and the cBWT index of $\mathcal{R}$ as input. If $\pi(V) = \$$, then we return $\mathrm{cnt}_\mathcal{R}(V) = 0$, $\mathrm{plcp}^\infty_\mathcal{R}(V) = -1$ and $\mathrm{slcp}^\infty_\mathcal{R}(V) = 0$ in Line 2 since $\mathcal{R} \subset \Sigma^+$. Thus, assume $\pi(V) > \$$. For each $i \in [0..\pi(V)]$, let $J_i = [\ell_i..r_i]$ maximal such that $r_i \leq y$ and $\mathrm{lcp}^\infty(V, C_\mathcal{R}(\mathrm{CA}_\mathcal{R}[j])) = i$ for each $j \in J_i$. The following statements are due to Lemma 3.7 and $\mathcal{R} \subset \Sigma^+$.

- If $j \in [y+1..\rho]$, then $C_\mathcal{R}(\mathrm{CA}_\mathcal{T}[\mathrm{LF}_\mathcal{R}[j]]) \prec_\omega V$ if and only if $\mathrm{L}_\mathcal{R}[j] \geq \pi(V)+1$ and $\mathrm{lcp}^\infty(C_\mathcal{R}(\mathrm{CA}_\mathcal{T}[i]), \mathrm{Rot}(V,j)) \geq \pi(V)+1$.
- If $j \in [r_{\pi(V)}+1..y]$, then $C_\mathcal{R}(\mathrm{CA}_\mathcal{T}[\mathrm{LF}_\mathcal{R}[j]]) \prec_\omega V$ if and only if $\mathrm{L}_\mathcal{R}[j] \geq \pi(V)$.
- If $i \in [0..\pi(V)]$ and $j \in J_i$, then $C_\mathcal{R}(\mathrm{CA}_\mathcal{T}[\mathrm{LF}_\mathcal{R}[j]]) \prec_\omega V$ if and only if $\mathrm{L}_\mathcal{T}[j] \geq i$.

We apply these results from Line 3 through Line 10 to compute $\mathrm{cnt}_\mathcal{R}(V)$. Leveraging Lemma 4.1 and (the proof of) Lemma 3.12, we then compute $\mathrm{plcp}^\infty_\mathcal{R}(V)$ and $\mathrm{slcp}^\infty_\mathcal{R}(V)$ from Line 11 through Line 16 and from Line 17 through Line 23, respectively. ◀

Algorithm 3 Computing $c = \mathrm{cnt}_{\mathcal{R}}(V)$, $p = \mathrm{plcp}^{\infty}_{\mathcal{R}}(V)$ and $s = \mathrm{slcp}^{\infty}_{\mathcal{R}}(V)$. Here, $\emptyset \neq \mathcal{R} \subset \Sigma^+$, $\rho = |\mathrm{CA}_{\mathcal{R}}|$, $V \in \Sigma^+_{\$}$, $\mathrm{rank}_{\infty}(V, |V|) \geq 1$, and $y = \mathrm{cnt}_{\mathcal{R}}(\mathrm{Rot}(V, 1))$.

1 **Function** `next`$(\pi(V), y, \mathrm{plcp}^{\infty}_{\mathcal{R}}(\mathrm{Rot}(V,1)), \mathrm{slcp}^{\infty}_{\mathcal{R}}(\mathrm{Rot}(V,1)), \mathrm{F}_{\mathcal{R}}, \mathrm{L}_{\mathcal{R}}, \mathrm{LCP}^{\infty}_{\mathcal{R}})$**:**
2 **if** $\pi(V) = \$$ **then return** $0, -1, 0$;
3 **if** $\mathrm{slcp}^{\infty}_{\mathcal{R}}(\mathrm{Rot}(V,1)) \geq \pi(V) + 1$ **then** $[\ell..r] \leftarrow \mathrm{MI}_{\mathrm{LCP}^{\infty}_{\mathcal{R}}}(y+1, \pi(V)+1)$;
4 **else if** $\mathrm{plcp}^{\infty}_{\mathcal{R}}(\mathrm{Rot}(V,1)) \geq \pi(V) + 1$ **then** $[\ell..r] \leftarrow \mathrm{MI}_{\mathrm{LCP}^{\infty}_{\mathcal{R}}}(y, \pi(V)+1)$;
5 **else** $[\ell..r] \leftarrow [y+1..y]$;
6 $c \leftarrow \mathrm{rnkcnt}_{\mathrm{L}_{\mathcal{R}}}(y+1, r, \pi(V)+1, \sigma) + \mathrm{rnkcnt}_{\mathrm{L}_{\mathcal{R}}}(\ell, y, \pi(V), \sigma)$;
7 **for** $i \leftarrow \min\{\pi(V), \mathrm{plcp}^{\infty}_{\mathcal{R}}(\mathrm{Rot}(V,1))\}$ **to** 0 **do**
8 $r' \leftarrow \ell - 1$;
9 $[\ell..r] \leftarrow \mathrm{MI}_{\mathrm{LCP}^{\infty}_{\mathcal{R}}}(y, i)$;
10 $c \leftarrow c + \mathrm{rnkcnt}_{\mathrm{L}_{\mathcal{R}}}(\ell, r', i, \sigma)$;
11 **if** $c = 0$ **then** $p \leftarrow -1$;
12 **else if** $\mathrm{FL}_{\mathcal{R}}[c] \leq y$ *and* $\mathrm{F}_{\mathcal{R}}[c] = \pi(V) < \mathrm{plcp}^{\infty}_{\mathcal{R}}(\mathrm{Rot}(V,1))$ **then**
13 **if** $\mathrm{FL}_{\mathcal{R}}[c] = y$ **then** $p \leftarrow \mathrm{plcp}^{\infty}_{\mathcal{R}}(\mathrm{Rot}(V,1)) - \pi(V) + 1$;
14 **else if** $\pi(V) \geq \mathrm{RNV}_{\mathrm{LCP}^{\infty}_{\mathcal{R}}}(\mathrm{FL}_{\mathcal{R}}[c]+1, y, -1)$ **then** $p \leftarrow 1$;
15 **else** $p \leftarrow \min\{\mathrm{plcp}^{\infty}_{\mathcal{R}}(\mathrm{Rot}(V,1)), \mathrm{RNV}_{\mathrm{LCP}^{\infty}_{\mathcal{R}}}(\mathrm{FL}_{\mathcal{R}}[c]+1, y, -1)\} - \pi(V) + 1$;
16 **else** $p \leftarrow 1$;
17 **if** $c = \rho$ **then** $s \leftarrow -1$;
18 **else if** $\mathrm{FL}_{\mathcal{R}}[c+1] \geq y+1$ *and* $\mathrm{F}_{\mathcal{R}}[c+1] = \pi(V) < \mathrm{slcp}^{\infty}_{\mathcal{R}}(\mathrm{Rot}(V,1))$ **then**
19 **if** $\mathrm{FL}_{\mathcal{R}}[c+1] = y+1$ **then** $s \leftarrow \mathrm{slcp}^{\infty}_{\mathcal{R}}(\mathrm{Rot}(V,1)) - \pi(V) + 1$;
20 **else if** $\pi(V) < \mathrm{RNV}_{\mathrm{LCP}^{\infty}_{\mathcal{R}}}(y+2, \mathrm{FL}_{\mathcal{R}}[c+1], -1)$ **then**
21 $s \leftarrow \min\{\mathrm{slcp}^{\infty}_{\mathcal{R}}(\mathrm{Rot}(V,1)), \mathrm{RNV}_{\mathrm{LCP}^{\infty}_{\mathcal{R}}}(y+2, \mathrm{FL}_{\mathcal{R}}[c+1], -1)\} - \pi(V) + 1$;
22 **else** $s \leftarrow 1$;
23 **else** $s \leftarrow 1$;
24 **return** c, p, s

▶ **Lemma 4.7.** *Let* $\emptyset \neq \mathcal{R} \subset \Sigma^+$, $\rho = |\mathrm{CA}_{\mathcal{R}}|$, $V \in \Sigma^+_{\$}$, *and* $\mathrm{rank}_{\mathrm{x}}(V, |V|) \geq 1$. *If* $\pi(V) \neq \$$, *then* $\mathrm{cnt}_{\mathcal{R}}(V)$, $\mathrm{plcp}^{\infty}_{\mathcal{R}}(V)$ *and* $\mathrm{slcp}^{\infty}_{\mathcal{R}}(V)$ *can be computed from* $\pi(V)$, $\mathrm{cnt}_{\mathcal{R}}(\mathrm{Rot}(V,1))$, $\mathrm{plcp}^{\infty}_{\mathcal{R}}(\mathrm{Rot}(V,1))$ *and* $\mathrm{slcp}^{\infty}_{\mathcal{R}}(\mathrm{Rot}(V,1))$ *in* $O((1+\pi(V))\frac{\lg\sigma\lg\rho}{\lg\lg\rho})$ *time with the cBWT index of* $\mathcal{R}$.

Proof. We apply Algorithm 3. Correctness follows from Lemma 4.6, and the time complexity is due to the loop of Algorithm 3 in Line 7 and Lemma 2.1. ◀

For the iterative construction, we augment the cBWT index by a dynamic bit string E with the invariant that E is zeroed before and after each extension with a new string. We use E to temporarily mark modified parts in the cBWT such that we retain the functionality of the index even during construction. The length of E is the number ρ of characters indexed by the cBWT index, which we call in the next lemma the *augmented cBWT index* for clarity.

▶ **Lemma 4.8.** *Let* $\emptyset \neq \mathcal{R} \subset \Sigma^+$, $\rho = |\mathrm{CA}_{\mathcal{R}}|$, $S \in \Sigma^+$, $\mathcal{S} = \mathcal{R} \cup \{S\}$, $\lambda = |S|$, *and* $z = \max\{|V| \mid V \in \mathcal{S}\}$. *The augmented cBWT index of* $\mathcal{R}$ *can be extended to index* $\mathcal{S}$ *in* $O((\lambda+\rho)\lg\sigma)$ *bits of space and* $O(z\frac{\lg\sigma\lg(\lambda+\rho)}{\lg\lg(\lambda+\rho)})$ *time.*

Proof. We compute the cBWT index of $\{S\}$ within the claimed complexities by Corollary 4.5. Then we compute $\mathrm{cnt}_{\mathcal{R}}(\mathrm{Rot}(S,i))$, $\mathrm{plcp}^{\infty}_{\mathcal{R}}(\mathrm{Rot}(S,i))$ and $\mathrm{slcp}^{\infty}_{\mathcal{R}}(\mathrm{Rot}(S,i))$ for each $i \in [1..\lambda]$.

Case 1. Assume $\mathrm{Rot}(S,i) =_\omega R$ for some $R \in \mathcal{R}$ and $i \in [1..\lambda]$. Then $\mathrm{cnt}_\mathcal{R}(\mathrm{Rot}(S,i)) = r$, $\mathrm{plcp}^\infty_\mathcal{R}(\mathrm{Rot}(S,i)) = \mathrm{rank}_\infty(\langle \mathrm{Rot}(S,i)\rangle, \lambda)$, and, if $r < \rho$, $\mathrm{slcp}^\infty_\mathcal{R}(\mathrm{Rot}(S,i)) = \mathrm{LCP}^\infty_\mathcal{R}[r+1]$ by Lemma 3.3, where $i \in [1..\lambda]$ and r is the right boundary of $\mathrm{CR}_\mathcal{R}(\mathrm{Rot}(S,i)^\omega[..3z])$. Since $\mathrm{rank}_\infty(\langle \mathrm{Rot}(S,i-1)\rangle, \lambda) = \mathrm{rank}_\infty(\langle S^\omega[i..4z]\rangle, 4z-i+1)$ for each $i \in [2..\lambda+1]$, the last λ steps of the backward search for $S^\omega[2..4z]$ yield the necessary values to compute $\mathrm{cnt}_\mathcal{R}(\mathrm{Rot}(S,i))$, $\mathrm{plcp}^\infty_\mathcal{R}(\mathrm{Rot}(S,i))$ and $\mathrm{slcp}^\infty_\mathcal{R}(\mathrm{Rot}(S,i))$ for all $i \in [1..\lambda]$ in descending order within the claimed complexities by Theorem 3.15.

Case 2. Assume $\mathrm{Rot}(S,i) \neq_\omega R$ for each $R \in \mathcal{R}$ and $i \in [1..\lambda]$. Let $V = S^\omega[..4z] \cdot \$$. By assumption and Corollary 3.4, $\mathrm{cnt}_\mathcal{R}(\mathrm{Rot}(S,i)) = \mathrm{cnt}_\mathcal{R}(\mathrm{Rot}(V,i))$, $\mathrm{plcp}^\infty_\mathcal{R}(\mathrm{Rot}(S,i)) = \mathrm{plcp}^\infty_\mathcal{R}(\mathrm{Rot}(V,i))$, and $\mathrm{slcp}^\infty_\mathcal{R}(\mathrm{Rot}(S,i)) = \mathrm{slcp}^\infty_\mathcal{R}(\mathrm{Rot}(V,i))$ for each $i \in [1..\lambda]$. We leverage Lemma 3.14 to preprocess V within the claimed complexities. Since $\mathrm{cnt}_\mathcal{R}(\mathrm{Rot}(V,4z)) = 0$, $\mathrm{plcp}^\infty_\mathcal{R}(\mathrm{Rot}(V,4z)) = -1$ and $\mathrm{slcp}^\infty_\mathcal{R}(\mathrm{Rot}(V,4z)) = 0$ by construction, we can now use Lemma 4.7 to compute $\mathrm{cnt}_\mathcal{R}(\mathrm{Rot}(V,j))$, $\mathrm{plcp}^\infty_\mathcal{R}(\mathrm{Rot}(V,j))$ and $\mathrm{slcp}^\infty_\mathcal{R}(\mathrm{Rot}(V,j))$ for each $j \in [1..4z-1]$ in descending order. Hence, we obtain $\mathrm{cnt}_\mathcal{R}(\mathrm{Rot}(S,i))$, $\mathrm{plcp}^\infty_\mathcal{R}(\mathrm{Rot}(S,i))$ and $\mathrm{slcp}^\infty_\mathcal{R}(\mathrm{Rot}(S,i))$ for all $i \in [1..\lambda]$ in descending order within the claimed complexities by Lemma 3.6.

Leveraging the backward search on the cBWT index of $\{S\}$, we store the $\mathrm{plcp}^\infty_\mathcal{R}$-values and $\mathrm{slcp}^\infty_\mathcal{R}$-values in lex-order, which takes $O(\lambda \lg \sigma)$ bits of space. To store the $\mathrm{cnt}_\mathcal{R}$-values in lex-order and stay within the claimed time and space complexity, we utilize E, which we assume to be represented by the data structure of Lemma 2.1. For each $i \in [1..\lambda]$ in descending order, we call $\mathrm{insert}_E(\mathrm{select}_0(E, \mathrm{cnt}_\mathcal{R}(\mathrm{Rot}(S,i))) + 1, 1)$, and discard $\mathrm{cnt}_\mathcal{R}(\mathrm{Rot}(S,i))$ once $\mathrm{cnt}_\mathcal{R}(\mathrm{Rot}(S,i-1))$ has been computed. Then $\mathrm{cnt}_\mathcal{R}(C_{\{S\}}(\mathrm{CA}_{\{S\}}[i])) = \mathrm{rank}_0(E, \mathrm{select}_1(E,i))$. The following statements for each $i \in [1..\lambda+\rho]$ are due to construction.

- If $E[i] = 0$, then $\mathrm{F}_\mathcal{S}[i] = \mathrm{F}_\mathcal{R}[\mathrm{rank}_0(E,i)]$ and $\mathrm{L}_\mathcal{S}[i] = \mathrm{L}_\mathcal{R}[\mathrm{rank}_0(E,i)]$.
- If $E[i] = 1$, then $\mathrm{F}_\mathcal{S}[i] = \mathrm{F}_{\{S\}}[\mathrm{rank}_1(E,i)]$ and $\mathrm{L}_\mathcal{S}[i] = \mathrm{L}_{\{S\}}[\mathrm{rank}_1(E,i)]$.
- $\mathrm{LCP}^\infty_\mathcal{S}[1] = 0$.
- If $i \geq 2$, $E[i-1] = 1$ and $E[i] = 1$, then $\mathrm{LCP}^\infty_\mathcal{S}[i] = \mathrm{LCP}^\infty_{\{S\}}[\mathrm{rank}_1(E,i)]$.
- If $i \geq 2$, $E[i-1] = 1$ and $E[i] = 0$, then $\mathrm{LCP}^\infty_\mathcal{S}[i] = \mathrm{slcp}^\infty_\mathcal{R}(C_{\{S\}}(\mathrm{CA}_{\{S\}}[\mathrm{rank}_1(E,i)]))$.
- If $i \geq 2$, $E[i-1] = 0$ and $E[i] = 1$, then $\mathrm{LCP}^\infty_\mathcal{S}[i] = \mathrm{plcp}^\infty_\mathcal{R}(C_{\{S\}}(\mathrm{CA}_{\{S\}}[\mathrm{rank}_1(E,i)]))$.
- If $i \geq 2$, $E[i-1] = 0$ and $E[i] = 0$, then $\mathrm{LCP}^\infty_\mathcal{S}[i] = \mathrm{LCP}^\infty_\mathcal{R}[\mathrm{rank}_0(E,i)]$.

Consequently, $O(\lambda)$ queries and operations are needed to update the cBWT index of $\mathcal{R}$ to index $\mathcal{S}$. Finally, we zero E. The claimed complexities then follow from Lemma 2.1. ◀

We are finally able to state the main result of this section.

▶ **Theorem 4.9.** *Let $\emptyset \neq \mathcal{T} = \{T_1, ..., T_d\} \subset \Sigma^+$ and $n = |T_1 \cdots T_d|$. The cBWT index of $\mathcal{T}$ can be constructed in $O(n \lg \sigma)$ bits of space and $O(n \frac{\lg \sigma \lg n}{\lg \lg n})$ time.*

Proof. Let $|T_{k-1}| \leq |T_k|$ for each $k \in [2..d]$, and let E_j denote a zeroed bit string of length $|T_1 \cdots T_j|$ for each $j \in [1..d]$. First, we construct E_1 and the cBWT index of $\{T_1\}$ within the claimed complexities by Lemma 4.5, where each string is represented by the data structure of Lemma 2.1. Subsequently, we iteratively extend the cBWT index of $\{T_1, ..., T_{k-1}\}$ augmented by E_{k-1} to the cBWT index of $\{T_1, ..., T_k\}$ augmented by E_k for each $k \in [2..d]$ in ascending order leveraging Lemma 4.8. Then the cBWT index of $\mathcal{T}$ is constructed in $O((n+|T_d|) \lg \sigma) = O(n \lg \sigma)$ bits of space and $O((\sum_{k=1}^d |T_k|) \frac{\lg \sigma \lg n}{\lg \lg n}) = O(n \frac{\lg \sigma \lg n}{\lg \lg n})$ time. ◀

References

1 Bastien Auvray, Julien David, Richard Groult, and Thierry Lecroq. Approximate cartesian tree matching: An approach using swaps. In *Proc. SPIRE*, volume 14240 of *LNCS*, pages 49–61, 2023. doi:10.1007/978-3-031-43980-3_5.

2 Christina Boucher, Davide Cenzato, Zsuzsanna Lipták, Massimiliano Rossi, and Marinella Sciortino. r-indexing the eBWT. In *Proc. SPIRE*, volume 12944 of *LNCS*, pages 3–12, 2021. doi:10.1007/978-3-030-86692-1_1.

3 Erik D. Demaine, Gad M. Landau, and Oren Weimann. On Cartesian trees and range minimum queries. *Algorithmica*, 68(3):610–625, 2014. doi:10.1007/s00453-012-9683-x.

4 Simone Faro, Thierry Lecroq, Kunsoo Park, and Stefano Scafiti. On the longest common Cartesian substring problem. *The Computer Journal*, 66(4):907–923, 2022. doi:10.1093/COMJNL/BXAB204.

5 Paolo Ferragina and Giovanni Manzini. Opportunistic data structures with applications. In *Proc. FOCS*, pages 390–398, 2000. doi:10.1109/SFCS.2000.892127.

6 Johannes Fischer. Optimal succinctness for range minimum queries. In *Proc. LATIN*, volume 6034 of *LNCS*, pages 158–169, 2010. doi:10.1007/978-3-642-12200-2_16.

7 Peter Foster, Anssi Klapuri, and Simon Dixon. A method for identifying repetition structure in musical audio based on time series prediction. In *Proc. EUSIPCO*, pages 1299–1303. IEEE, 2012. URL: https://ieeexplore.ieee.org/document/6334323/.

8 Tak-Chung Fu, Korris Fu-Lai Chung, Robert Wing Pong Luk, and Chak-man Ng. Stock time series pattern matching: Template-based vs. rule-based approaches. *Eng. Appl. Artif. Intell.*, 20(3):347–364, 2007. doi:10.1016/J.ENGAPPAI.2006.07.003.

9 Mitsuru Funakoshi, Takuya Mieno, Yuto Nakashima, Shunsuke Inenaga, Hideo Bannai, and Masayuki Takeda. Computing maximal palindromes in non-standard matching models. In *Proc. IWOCA*, volume 14764 of *LNCS*, pages 165–179, 2024. doi:10.1007/978-3-031-63021-7_13.

10 Paweł Gawrychowski, Samah Ghazawi, and Gad M. Landau. On indeterminate strings matching. In *Proc. CPM*, volume 161 of *LIPIcs*, pages 14:1–14:14, 2020. doi:10.4230/LIPICS.CPM.2020.14.

11 Daiki Hashimoto, Diptarama Hendrian, Dominik Köppl, Ryo Yoshinaka, and Ayumi Shinohara. Computing the parameterized Burrows–Wheeler transform online. In *Proc. SPIRE*, volume 13617 of *LNCS*, pages 70–85, 2022. doi:10.1007/978-3-031-20643-6_6.

12 Yuzuru Hiraga. Structural recognition of music by pattern matching. In *Proc. ICMC*. Michigan Publishing, 1997. URL: https://hdl.handle.net/2027/spo.bbp2372.1997.113.

13 Wing-Kai Hon, Tsung-Han Ku, Chen-Hua Lu, Rahul Shah, and Sharma V. Thankachan. Efficient algorithm for circular Burrows–Wheeler transform. In *Proc. CPM*, volume 7354 of *LNCS*, pages 257–268, 2012. doi:10.1007/978-3-642-31265-6_21.

14 Kento Iseri, Tomohiro I, Diptarama Hendrian, Dominik Köppl, Ryo Yoshinaka, and Ayumi Shinohara. Breaking a barrier in constructing compact indexes for parameterized pattern matching. In *Proc. ICALP*, volume 297 of *LIPIcs*, pages 89:1–89:19, 2024. doi:10.4230/LIPICS.ICALP.2024.89.

15 Natsumi Kikuchi, Diptarama Hendrian, Ryo Yoshinaka, and Ayumi Shinohara. Computing covers under substring consistent equivalence relations. In *Proc. SPIRE*, volume 12303 of *LNCS*, pages 131–146, 2020. doi:10.1007/978-3-030-59212-7_10.

16 Jinil Kim, Peter Eades, Rudolf Fleischer, Seok-Hee Hong, Costas S. Iliopoulos, Kunsoo Park, Simon J. Puglisi, and Takeshi Tokuyama. Order-preserving matching. *Theor. Comput. Sci.*, 525:68–79, 2014. doi:10.1016/J.TCS.2013.10.006.

17 Sung-Hwan Kim and Hwan-Gue Cho. A compact index for Cartesian tree matching. In *Proc. CPM*, volume 191 of *LIPIcs*, pages 18:1–18:19, 2021. doi:10.4230/LIPIcs.CPM.2021.18.

18 Sungmin Kim and Yo-Sub Han. Approximate Cartesian tree pattern matching. In *Proc. DLT*, volume 14791 of *LNCS*, pages 189–202, 2024. doi:10.1007/978-3-031-66159-4_14.

19 Marcin Kubica, Tomasz Kulczyński, Jakub Radoszewski, Wojciech Rytter, and Tomasz Waleń. A linear time algorithm for consecutive permutation pattern matching. *Inf. Process. Lett.*, 113(12):430–433, 2013. doi:10.1016/J.IPL.2013.03.015.

20 Sabrina Mantaci, Antonio Restivo, Giovanna Rosone, and Marinella Sciortino. An extension of the Burrows–Wheeler transform. *Theor. Comput. Sci.*, 387(3):298–312, 2007. doi:10.1016/j.tcs.2007.07.014.

21 Yoshiaki Matsuoka, Takahiro Aoki, Shunsuke Inenaga, Hideo Bannai, and Masayuki Takeda. Generalized pattern matching and periodicity under substring consistent equivalence relations. *Theor. Comput. Sci.*, 656:225–233, 2016. doi:10.1016/j.tcs.2016.02.017.

22 Gonzalo Navarro and Kunihiko Sadakane. Fully functional static and dynamic succinct trees. *ACM Trans. Algorithms*, 10(3):16:1–16:39, 2014. doi:10.1145/2601073.

23 Akio Nishimoto, Noriki Fujisato, Yuto Nakashima, and Shunsuke Inenaga. Position heaps for Cartesian-tree matching on strings and tries. In *Proc. SPIRE*, volume 12944 of *LNCS*, pages 241–254, 2021. doi:10.1007/978-3-030-86692-1_20.

24 Tsubasa Oizumi, Takeshi Kai, Takuya Mieno, Shunsuke Inenaga, and Hiroki Arimura. Cartesian tree subsequence matching. In *Proc. CPM*, volume 223 of *LIPIcs*, pages 14:1–14:18, 2022. doi:10.4230/LIPICS.CPM.2022.14.

25 Eric M. Osterkamp and Dominik Köppl. Extending the parameterized Burrows–Wheeler transform. In *Proc. DCC*, pages 143–152, 2024. doi:10.1109/DCC58796.2024.00022.

26 Sung Gwan Park, Magsarjav Bataa, Amihood Amir, Gad M. Landau, and Kunsoo Park. Finding patterns and periods in Cartesian tree matching. *Theor. Comput. Sci.*, 845:181–197, 2020. doi:10.1016/J.TCS.2020.09.014.

27 Siwoo Song, Geonmo Gu, Cheol Ryu, Simone Faro, Thierry Lecroq, and Kunsoo Park. Fast algorithms for single and multiple pattern Cartesian tree matching. *Theor. Comput. Sci.*, 849:47–63, 2021. doi:10.1016/J.TCS.2020.10.009.

28 Jean Vuillemin. A unifying look at data structures. *Commun. ACM*, 23(4):229–239, 1980. doi:10.1145/358841.358852.