2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)

Resolution of the Burrows-Wheeler Transform Conjecture

Dominik Kempa
University of California,
Berkeley, USA
kempa@berkeley.edu

Tomasz Kociumaka
Bar-Ilan University,
Ramat Gan, Israel
kociumaka@mimuw.edu.pl

***Abstract*—The Burrows–Wheeler Transform (BWT) is an invertible text transformation that permutes symbols of a text according to the lexicographical order of its suffixes. BWT is the main component of popular lossless compression programs (such as `bzip2`) as well as recent powerful compressed indexes (such as r-index [Gagie et al., J. ACM, 2020]), central in modern bioinformatics. The compression ratio of BWT is quantified by the number r of equal-letter runs. Despite the practical significance of BWT, no non-trivial bound on the value of r is known. This is in contrast to nearly all other known compression methods, whose sizes have been shown to be either always within a $\operatorname{polylog} n$ factor (where n is the length of text) from z, the size of Lempel–Ziv (LZ77) parsing of the text, or significantly larger in the worst case (by a n^{ε} factor for $\varepsilon > 0$).**

In this paper, we show that $r = \mathcal{O}(z \log^2 n)$ holds for every text. This result has numerous implications for text indexing and data compression; for example: (1) it proves that many results related to BWT automatically apply to methods based on LZ77, e.g., it is possible to obtain functionality of the suffix tree in $\mathcal{O}(z \operatorname{polylog} n)$ space; (2) it shows that many text processing tasks can be solved in the optimal time assuming the text is compressible using LZ77 by a sufficiently large $\operatorname{polylog} n$ factor; (3) it implies the first non-trivial relation between the number of runs in the BWT of the text and its reverse.

In addition, we provide an $\mathcal{O}(z \operatorname{polylog} n)$-time algorithm converting the LZ77 parsing into the run-length compressed BWT. To achieve this, we develop a number of new data structures and techniques of independent interest. In particular, we introduce a notion of compressed string synchronizing sets (generalizing the recently introduced powerful technique of string synchronizing sets [STOC 2019]) and show how to efficiently construct them. Next, we propose a new variant of wavelet trees for sequences of long strings, establish a non-trivial bound on their size, and describe efficient construction algorithms. Finally, we describe new indexes that can be constructed directly from the LZ77-compressed text and efficiently support pattern matching queries on substrings of the text.

***Keywords*-data compression, compressed indexing, Burrows–Wheeler Transform, Lempel–Ziv compression**

I. Introduction

Lossless data compression aims to exploit redundancy in the input data to represent it in a small space. Despite the abundance of compression methods, nearly every existing tool falls into one of the few general frameworks, among which the three most popular are: Lempel–Ziv compression (where the nominal and most commonly used is the LZ77 variant [66]), statistical compression (this includes, for example, context mixing [43], prediction by partial matching (PPM) [16], and dynamic Markov coding [17]), and Burrows–Wheeler transform (BWT) [13]. As seen in the Large Text Compression Benchmark [42], these three frameworks underlie most existing compressors.

One of the features that best differentiates these algorithms is whether they better remove the redundancy caused by skewed symbols frequencies or by repeated fragments. The idea in LZ77 (which underlies, for example, `7-zip` [54] and `gzip` [26] compressors) is to partition the input text into long substrings, each having an earlier occurrence in the text. Every substring is then encoded as a pointer to the previous occurrence using a pair of integers. This method natively handles long repeated substrings and can achieve an exponential compression ratio given sufficiently repetitive input. Statistical compressors, on the other hand, are based on representing (predicting) symbols in the input based on their frequencies. This is formally captured by the notion of the *kth order empirical entropy* $H_k(T)$ [18]. For any sufficiently long text T, symbol frequencies (taking context into account) in any power of T (the concatenation of several copies of T) do not change significantly [38, Lemma 2.6]. Therefore, $|T^t|H_k(T^t) \approx t \cdot |T|H_k(T)$ for any $t > 1$, i.e., entropy is not sensitive to long repetitions, and hence statistical compressors are outperformed by LZ77, when k is large, i.e., when the goal is to capture long repetitions [20], [25], [34], [38], [62].

The above analysis raises the question about the nature of compressibility of the Burrows–Wheeler transform. The compression of BWT-based compressors, such as `bzip2` [60], is quantified by the number r of equal-letter blocks in the BWT. The clear picture described above no longer applies to the measure r. On one hand, Manzini [47] proved that r can be upper-bounded in terms of the kth order empirical entropy of the input string. On the other hand, already in 2008, Sirén et al. [62] observed that BWT achieves excellent compression (superior to statistical methods) on highly repetitive collections and provided probabilistic analysis exhibiting cases when r is small. Yet, after more than a decade, no upper bound on r in terms of z was discovered.

This lack of understanding is particularly frustrating due to numerous applications of BWT in the field of bioinformatics

A full version of this paper is available at arxiv.org/abs/1910.10631. Proofs of the claims marked with ♠ are presented only in the full version.

2575-8454/20/$31.00 ©2020 IEEE
DOI 10.1109/FOCS46700.2020.00097

and compressed computation. One of the most successful applications of BWT is in *compressed indexing*, which aims to store a compressed string, simultaneously supporting various queries (such as random access, pattern matching, or even suffix array queries) on the uncompressed version. While classical (uncompressed) indexes, such as suffix trees [65] and suffix arrays [45], have been successful in many applications, they are not suitable for storing and searching big highly repetitive databases. Such datasets are virtually impossible to search without preprocessing: Github databases, for example, take more than 20 terabytes, and the recently finished 100000 Human Genome Project [1] produced 75 terabytes of DNA [49]. These databases are, however, highly compressible: Github averages 20 versions per project [49], and two human genomes are 99.9% similar [59]. This area has witnessed a remarkable surge of interest in recent years [5], [9], [10], [11], [15], [21], [23], [32], [50], [58], [61]. BWT-based indexes, such as r-index [25], are among the most powerful [23], and their space usage is up to $\mathcal{O}(\operatorname{polylog} n)$ factors away from the value r. For a comprehensive overview of this field, we refer the reader to a survey by Navarro [49].

In addition to text indexing, BWT has many applications in compressed computation. For example, BWT is the main component of the popular read aligners such as `Bowtie` [39], `BWA` [40], and `Soap2` [41]. Modern textbooks spend dozens of pages describing applications of BWT [2], [44], [48], [51]. The richness of these applications has even spawned a dedicated seminar [22]. Given the importance and practical significance of BWT, one of the biggest open problems that emerged in the field of lossless data compression and compressed computation asks:

What is the upper bound on the output size of the Burrows–Wheeler transform?

With the exception of BWT, essentially every other known compression method has been proven [24], [34] to produce output whose size is always within a $\operatorname{polylog} n$ factor from z, the output size of the LZ77 algorithm (e.g., grammar compression [14], collage systems [35], or macro schemes [63]), or larger by a polynomial factor (n^{ε} for some $\varepsilon > 0$) in the worst case (e.g., LZ78 [67], compressed word graphs (CDAWGs) [12]).[1] BWT is known to never compress much better than LZ77, i.e., $z = \mathcal{O}(r \log n)$ [24]. The opposite relation (that $r = \mathcal{O}(z \operatorname{polylog} n)$) was generally conjectured to be false. For example, after presenting how to support suffix array and suffix tree queries in $\mathcal{O}(r \operatorname{polylog} n)$ space, Gagie et al. [23] speculate that "*(...) it seems unlikely that one can provide suffix array or tree functionality within space related to g, z, or γ, since these measures are not related to the structure of the suffix array: this is likely to be a specific advantage of measure r*".

[1]The choice for LZ77 as a representative in this class follows from the fact that most of the other methods are NP-hard to optimize [14], [27], while LZ77 admits a simple linear-time compression algorithm (see, e.g., [30]).

Our Contribution: We prove that $r = \mathcal{O}(z \log^2 n)$ holds for all strings, resolving the BWT conjecture in the more surprising way than anticipated, and solving an open problem by Prezza [57] and Gagie et al. [23], [24]. This result alone has multiple implications for indexing and compression:

1) It is possible to support suffix array and suffix tree functionality in $\mathcal{O}(z \operatorname{polylog} n)$ space [25].
2) It was shown in [32] that many string processing tasks (including BWT and LZ77 construction) can be solved in $\mathcal{O}(n/\log_\sigma n + r \operatorname{polylog} n)$ time (where σ is the alphabet size), i.e., if the text is sufficiently compressible by BWT (formally, when $n/r = \Omega(\operatorname{polylog} n)$), these tasks can be solved in optimal time (which is unlikely to be possible for general texts [33]). Our result loosens this assumption to $n/z = \Omega(\operatorname{polylog} n)$.
3) Until now, methods based on the Burrows–Wheeler transform were thought to be neither statistical nor dictionary (LZ-like) compression algorithms [62], [23]. Our result challenges the notion that the BWT is its own compression type: In view of our bound, BWT is much closer to LZ compressors than was previously thought.

Our slightly stronger bound $r = \mathcal{O}(\delta \log^2 n)$, where $\delta \leq z$ is a symmetric (insensitive to string reversal) repetitiveness measure recently introduced in [37], further shows that:

4) The number $\bar{r}$ of BWT runs in the reverse of the text satisfies $\bar{r} = \mathcal{O}(r \log^2 n)$, which is the first non-trivial bound in terms of r. This result is of practical importance due to many algorithms whose runtime depends on $\bar{r}$ [6], [7], [8], [52], [53], [55], [56].

After proving $r = \mathcal{O}(z \log^2 n)$ and $r = \mathcal{O}(\delta \log^2 n)$, by a tighter analysis, we obtain $r = \mathcal{O}(z \log z \max(1, \log \frac{n}{z \log z}))$ and $r = \mathcal{O}(\delta \log \delta \max(1, \frac{n}{\delta \log \delta}))$, and we prove that the latter is asymptotically tight for the full spectrum of values of n and δ. As a side-result, we obtain a tight upper bound $\mathcal{O}(n \log \delta)$ on the sum of irreducible LCP values. This improves upon the previously known bound $\mathcal{O}(n \log r)$ [29].

We then describe an $\mathcal{O}(z \log^8 n)$-time algorithm converting the LZ77 parsing into run-length compressed BWT (the $\operatorname{polylog} n$ factor has not been optimized). This offers up to exponential speedup over the previously fastest space-efficient algorithms, which need $\Omega(n \log z)$ time [55], [53]. To achieve this, we develop new data structures and techniques of independent interest. In particular, we introduce a notion of *compressed string synchronizing sets*, generalizing the powerful technique introduced in [33]. We also describe a new variant of wavelet trees [28], designed to work for sequences of long strings. Finally (♠), we describe new indexes that can be built directly from the LZ77-compressed text and support fast pattern matching queries on text substrings.

Organization of the Paper: Section II introduces the basic notation. We present our upper and lower bounds in Sections III and IV. Finally, in Section V, we outline our algorithm converting LZ77 to run-length compressed BWT.

II. Preliminaries

For any string S we write $S[i\,..\,j]$, where $1 \le i, j \le |S|$, to denote a substring of S. If $i > j$, we assume $S[i\,..\,j]$ to be the empty string ε. By $\overline{S}$ we denote the *reverse* of S.

An integer $p \in [1\,..\,|S|]$ is a *period* of a string S if $S[i] = S[i+p]$ for $i \in [1\,..\,|S|-p]$. The classic *periodicity lemma* [19] states that if a string S has periods p, q such that $p + q - \gcd(p,q) \le |S|$, then $\gcd(p,q)$ is also its period.

The shortest period of S is denoted as $\mathrm{per}(S)$. A string S is called *periodic* if $\mathrm{per}(S) \le \frac{1}{2}|S|$. The following fact is a folklore consequence of the periodicity lemma.

Fact II.1 (see [4, Fact 1])**.** *Any two distinct periodic strings of the same length differ on at least two positions.*

Throughout we consider a string (text) $T[1\,..\,n]$ of $n \ge 1$ symbols from an ordered alphabet Σ of size σ. We assume $T[n] = \$$, where $\$ \in \Sigma$ is the lexicographically smallest symbol in Σ, and that $\$$ does not occur anywhere else in T.

The *suffix array* [45] of T is an array $\mathrm{SA}[1\,..\,n]$ containing a permutation of the integers $[1\,..\,n]$ such that $T[\mathrm{SA}[1]\,..\,n] \prec T[\mathrm{SA}[2]\,..\,n] \prec \cdots \prec T[\mathrm{SA}[n]\,..\,n]$, where $\prec$ denotes the lexicographical order. The closely related *Burrows–Wheeler transform* [13] $\mathrm{BWT}[1\,..\,n]$ of T is defined by $\mathrm{BWT}[i] = T[\mathrm{SA}[i]-1]$ if $\mathrm{SA}[i] > 1$ and $\mathrm{BWT}[i] = T[n]$ otherwise. The BWT is invertible; given $\mathrm{BWT}[1\,..\,n]$, the text T can be restored in $\mathcal{O}(n)$ time. For any string $S = c_1^{\ell_1} c_2^{\ell_2} \cdots c_h^{\ell_h}$, where $c_i \in \Sigma$ and $\ell_i > 0$ for $i \in [1\,..\,h]$, and $c_i \neq c_{i+1}$ for $i \in [1\,..\,h)$, we define the *run-length encoding* of S as a sequence $\mathrm{RL}(S) = ((c_1, \lambda_1), \ldots, (c_h, \lambda_h))$, where $\lambda_i = \ell_1 + \ldots + \ell_i$ for $i \in [1\,..\,h]$. Throughout, we let $r = |\mathrm{RL}(\mathrm{BWT})|$ denote the number of runs in the BWT, e.g., for the string $T = \mathtt{bbabaababababaababa\$}$ in Fig. 1, we have $\mathrm{BWT} = \mathtt{a}^1\mathtt{b}^6\mathtt{a}^1\mathtt{b}^2\mathtt{a}^6\mathtt{b}^1\mathtt{a}^2\mathtt{\$}^1$, and hence $r = 8$.

By $\mathrm{lcp}(S_1, S_2)$ we denote the length of the longest common prefix of strings S_1 and S_2. We let $\mathrm{LCE}(j_1, j_2) = \mathrm{lcp}(T[j_1\,..\,n], T[j_2\,..\,n])$, where $j_1, j_2 \in [1\,..\,n]$. The *LCP array* (see [45], [31]), $\mathrm{LCP}[1\,..\,n]$, is defined as $\mathrm{LCP}[i] = \mathrm{LCE}(\mathrm{SA}[i], \mathrm{SA}[i-1])$ for $i \in [2\,..\,n]$ and $\mathrm{LCP}[1] = 0$. We say that the value $\mathrm{LCP}[i]$ is *reducible* if $\mathrm{BWT}[i] = \mathrm{BWT}[i-1]$ and *irreducible* otherwise (including $i = 1$). Note that there are r irreducible LCP values.

Theorem II.2 (Kärkkäinen et al. [29])**.** *The sum of all irreducible LCP values is at most $n \log r$.*

The LZ77 factorization [66] uses the notion of the *longest previous factor* (LPF). The LPF at position $i \in [1\,..\,n]$ in T is a pair (p_i, ℓ_i) such that $p_i < i$ and $\ell_i = \mathrm{LCE}(p_i, i) > 0$ is maximized. In other words, $T[i\,..\,i+\ell_i-1]$ is the longest prefix of $T[i\,..\,n]$ which also occurs at some position $p_i < i$ in T. If $T[i]$ is the leftmost occurrence of a symbol in T, then such a pair does not exist. In this case, we define $p_i = T[i]$ and $\ell_i = 0$. If there is more than one possibility for p_i, we choose one arbitrarily.

i	$\mathrm{LCP}[i]$	$\mathrm{SA}[i]$	$\mathrm{BWT}[i]$	$T[\mathrm{SA}[i]\,..\,n]$
1	**0**	20	a	$
2	**0**	19	b	a$
3	1	14	b	aababa$
4	6	5	b	aabababababaababa$
5	1	17	b	aba$
6	3	12	b	abaababa$
7	8	3	b	abaababababaababa$
8	**3**	15	a	ababa$
9	**5**	10	b	ababaababa$
10	5	8	b	abababaababa$
11	**7**	6	a	abababababaababa$
12	0	18	a	ba$
13	2	13	a	baababa$
14	7	4	a	baababababaababa$
15	2	16	a	baba$
16	4	11	a	babaababa$
17	**9**	2	b	babaababababaababa$
18	**4**	9	a	bababaababa$
19	6	7	a	babababaababa$
20	**1**	1	$	bbabaababababaababa$

Figure 1. A list of lexicographically sorted suffixes of the string $T = \mathtt{bbabaababababaababa\$}$ along with the BWT, SA, and LCP tables. The irreducible LCP values are bold and underlined.

The LZ77 factorization (or the LZ77 parsing) of a string T is then just a greedy, left-to-right parsing of T into longest previous factors. More precisely, if the jth LZ factor (called the jth *phrase*) in the parsing is to start at position i, then we output (p_i, ℓ_i) (to represent the jth phrase), and then the $(j+1)$th phrase starts at position $i + \max(\ell_i, 1)$. We denote the number of phrases in the LZ77 parsing by z.

The text $\mathtt{bbabaababababaababa\$}$ of Fig. 1 has LZ77 factorization $\mathtt{b} \cdot \mathtt{b} \cdot \mathtt{a} \cdot \mathtt{ba} \cdot \mathtt{aba} \cdot \mathtt{bababa} \cdot \mathtt{ababa} \cdot \mathtt{\$}$ with $z = 8$ phrases, so its LZ77 representation is

$$(\mathtt{b}, 0), (1, 1), (\mathtt{a}, 0), (2, 2), (3, 3), (7, 6), (10, 5), (\mathtt{\$}, 0).$$

The following relation between z and r is known.

Theorem II.3 (Gagie et al. [24])**.** *Every string of length n satisfies $z = \mathcal{O}(r \log n)$.*

III. Upper Bounds

A. Basic Upper Bound

To illustrate the main idea of our proof technique, we first show the upper bound in its simplest form $r = \mathcal{O}(z \log^2 n)$. The following lemma stands at the heart of our proof.

Lemma III.1. *For every $\ell \in [1\,..\,n]$, the number of irreducible LCP values in $[\ell\,..\,2\ell)$ is $\mathcal{O}(z \log n)$.*

Proof: Let T^∞ be an infinite string defined so that $T^\infty[i] = T[1 + (i-1) \bmod n]$ for $i \in \mathbb{Z}$; in particular, $T^\infty[1\,..\,n] = T[1\,..\,n]$. Note that $T^\infty[\mathrm{SA}[1]\,..] \prec \cdots \prec T^\infty[\mathrm{SA}[n]\,..]$ and $\mathrm{BWT}[i] = T^\infty[\mathrm{SA}[i]-1]$ for $i \in [1\,..\,n]$.

Denote $\mathcal{S}_m = \{S \in \Sigma^m : S \text{ is a substring of } T^\infty\}$ for $m \ge 1$. Observe that $|\mathcal{S}_m| \le mz$ since every length-m substring of T^∞ has an occurrence crossing or beginning at a phrase boundary of the LZ77 parsing of T. This includes

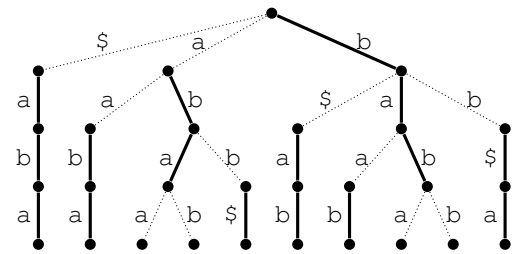

Figure 2. The trie $\mathcal{T}$ of reversed length-4 substrings of T^∞ for $T =$ bbabaababababaababa$ of Fig. 1. Light edges are thin and dotted.

substrings overlapping two copies of T, which cross the boundary between the last and the first phrase.

The idea of the proof is as follows. With each irreducible value LCP$[i] \in [\ell\,..\,2\ell)$, we associate a cost ℓ which is charged to individual characters of strings in $\mathcal{S}_{3\ell}$. We then show that each of the strings in $\mathcal{S}_{3\ell}$ is charged at most $2\log n$ times. The number of irreducible LCP values in $[\ell\,..\,2\ell)$ equals $\frac{1}{\ell}$ times the total cost, which is at most

$$|\mathcal{S}_{3\ell}| \cdot 2\log n \le 6\ell z \log n.$$

To devise the announced assignment of cost to characters of strings in $\mathcal{S}_{3\ell}$, consider the trie $\mathcal{T}$ of all reversed strings in $\mathcal{S}_\ell$ (see Fig. 2 for an example). By v_X denote the node of $\mathcal{T}$ whose path from the root of $\mathcal{T}$ is labelled by a string X.

Let LCP$[i] \in [\ell\,..\,2\ell)$ be an irreducible LCP value; note that $i > 1$ due to LCP$[i] > 0$. Let $j_0 = \mathrm{SA}[i-1]$ and $j_1 = \mathrm{SA}[i]$ so that LCP$[i] = \mathrm{LCE}(j_0, j_1)$. Since LCP$[i]$ is irreducible, we have $T^\infty[j_0-1] = \mathrm{BWT}[i-1] \neq \mathrm{BWT}[i] = T^\infty[j_1-1]$. For $k \in [1\,..\,\ell]$, the kth unit of the cost associated with LCP$[i]$ is charged to the kth character ($T^\infty[j_t-1]$) of the string $T^\infty[j_t-k\,..\,j_t-k+3\ell) \in \mathcal{S}_{3\ell}$, where $t \in \{0,1\}$ is such that the subtree of $\mathcal{T}$ rooted at $v_{\overline{T^\infty[j_t-1\,..\,j_t-k+\ell)}}$ contains less leaves than the subtree rooted at $v_{\overline{T^\infty[j_{1-t}-1\,..\,j_{1-t}-k+\ell)}}$ (we choose $t=0$ in case of ties).

Note that at most $\log n$ characters of each $S \in \mathcal{S}_{3\ell}$ can be charged during the above procedure: whenever $S[k]$, $k \in [1\,..\,\ell]$, is charged, the subtree of $\mathcal{T}$ rooted at $v_{\overline{S[k+1\,..\,\ell]}}$ has at least twice as many leaves as the subtree rooted at $v_{\overline{S[k\,..\,\ell]}}$, and this can happen for at most $\log|\mathcal{S}_\ell| \le \log n$ positions k.

It remains to show that, for every $S \in \mathcal{S}_{3\ell}$, a single position $S[k]$, $k \in [1\,..\,\ell]$, can be charged at most twice. First, observe that characters charged for a single irreducible value LCP$[i]$ are at different positions (of strings in $\mathcal{S}_{3\ell}$). Hence, to analyze the total charge assigned to $S[k]$, we only need to bound the number of possible candidate positions i. Let $[b\,..\,e]$ be the set of indices i' such that $T^\infty[\mathrm{SA}[i']\,..]$ starts with $S[k+1\,..\,3\ell]$. In the above procedure, if a character $S[k]$ is charged a unit of cost corresponding to LCP$[i]$, then $S[k+1\,..\,3\ell]$ is a prefix of either $T^\infty[\mathrm{SA}[i-1]\,..] = T^\infty[j_0\,..]$ or $T^\infty[\mathrm{SA}[i]\,..] = T^\infty[j_1\,..]$. Hence, $\{i-1, i\} \cap [b\,..\,e] \neq \emptyset$. At the same time, $\mathrm{LCE}(\mathrm{SA}[i-1], \mathrm{SA}[i]) < 2\ell$ and all strings $T^\infty[\mathrm{SA}[i']\,..]$ with $i' \in [b\,..\,e]$ share a common prefix $S[k+1\,..\,3\ell]$ of length $3\ell - k \ge 2\ell$. Consequently, $i = b$ or $i = e+1$. ■

Theorem III.2. *All length-n strings satisfy $r = \mathcal{O}(z\log^2 n)$.*

Proof: Recall that r is the total number of irreducible LCP values. Thus, the claim follows by applying Lemma III.1 for $\ell_i = 2^i$ with $i \in [0\,..\,\lfloor \log n \rfloor]$, and observing that the number of LCP values 0 is $\sigma \le z$. ■

B. Tighter Upper Bound

To obtain a tighter bound, we refine the ideas from Section III-A, starting with a counterpart of Lemma III.1.

Lemma III.3. *For every $\ell \in [1\,..\,n]$, the number of irreducible LCP values in $[\ell\,..\,2\ell)$ is $\mathcal{O}(z\log z)$.*

Proof: The proof follows closely that of Lemma III.1. However, with each irreducible value LCP$[i] \in [\ell\,..\,2\ell)$, we associate cost $\lceil \frac{1}{2}\ell \rceil$ rather ℓ. We then show that each of the strings in $\mathcal{S}_{3\ell}$ is charged at most $2\cdot(3+\log z)$ times (rather than $2\log n$ times). Then, the number of irreducible LCP values in the range $[\ell\,..\,2\ell)$ does not exceed $\frac{2}{\ell}$ times the total cost, which is bounded by

$$|\mathcal{S}_{3\ell}| \cdot 2 \cdot (3+\log z) \le 6\ell z(3+\log z).$$

Recall the trie $\mathcal{T}$ of all reversed length-ℓ strings in $\mathcal{S}_\ell$. For a node v of $\mathcal{T}$, by $\mathrm{size}(v)$ we denote the number of leaves in the subtree of $\mathcal{T}$ rooted in v. An edge connecting $v \neq \mathrm{root}(\mathcal{T})$ to its parent in $\mathcal{T}$ is called *light* if v has a sibling v' satisfying $\mathrm{size}(v') \ge \mathrm{size}(v)$ (see Fig. 2). In the proof of Lemma III.1, we observed that the characters $S[k]$ of $S \in \mathcal{S}_{3\ell}$ that can be charged correspond to light edges on the root-to-leaf path in $\mathcal{T}$ spelling $\overline{S[1\,..\,\ell]}$: whenever $S[k]$ with $k \in [1\,..\,\ell]$ is charged, the edge connecting $v_{\overline{S[k\,..\,\ell]}}$ to its parent $v_{\overline{S[k+1\,..\,\ell]}}$ is light. We then noted that there are at most $\log|\mathcal{S}_\ell| \le \log n$ light edges on each root-to-leaf path in $\mathcal{T}$. Here, we perform the same assignment of cost to the characters of strings in $\mathcal{S}_{3\ell}$ as in Lemma III.1, but only for units $k \in [1\,..\,\lceil \frac{1}{2}\ell \rceil]$. This implies that only characters $S[k]$ of $S \in \mathcal{S}_{3\ell}$ with $k \le \lceil \frac{1}{2}\ell \rceil$ are charged. It remains to show that any root-to-leaf path in $\mathcal{T}$ contains at most $3+\log z$ light edges between a node at depth at least $\lfloor \frac{1}{2}\ell \rfloor$ and its child.

Consider a light edge from a node v to its parent u at depth at least $\lfloor \frac{1}{2}\ell \rfloor$. Let v' be a sibling of v satisfying $\mathrm{size}(v') \ge \mathrm{size}(v)$, and let $S_v, S_{v'}$ be the labels of the paths from the root to v and v'. These labels differ on the last position only so, by Fact II.1, they cannot be both periodic. Let $\widetilde{v} \in \{v, v'\}$ be such that $S_{\widetilde{v}}$ is not periodic, and let $\widetilde{m} = \mathrm{size}(\widetilde{v})$.

Consider the set $\mathcal{S}$ of length-ℓ strings corresponding to the leaves in the subtree rooted at $\widetilde{v}$ (i.e., labels of the root-to-leaf paths passing through $\widetilde{v}$). Define $\overline{\mathcal{S}} := \{\overline{P} : P \in \mathcal{S}\}$ and note that $\overline{\mathcal{S}} \subseteq \mathcal{S}_\ell$ because $\mathcal{T}$ is the trie of *reversed* strings from $\mathcal{S}_\ell$. Let $e_1 < \cdots < e_{\widetilde{m}}$ denote the ending positions of the leftmost occurrences in $T^\infty[1\,..)$ of strings in $\overline{\mathcal{S}}$. By definition, we have an occurrence of $\overline{S_{\widetilde{v}}}$ ending in T^∞ at

every position e_i, $i \in [1\,..\,\widetilde{m}]$. Now, $\mathrm{per}(S_{\widetilde{v}}) > \frac{1}{2}|S_{\widetilde{v}}| \geq \frac{1}{4}\ell$ implies that $e_{i+1} - e_i > \frac{1}{4}\ell$ for every $i \in [1\,..\,\widetilde{m}-1]$ (since otherwise the two close occurrences of $\overline{S_{\widetilde{v}}}$ would yield $\mathrm{per}(\overline{S_{\widetilde{v}}}) = \mathrm{per}(S_{\widetilde{v}}) \leq \frac{1}{4}\ell$). Consequently, at least $\frac{1}{4}|\mathcal{S}| = \frac{1}{4}\widetilde{m}$ length-ℓ substrings of $T^\infty[1\,..)$ have disjoint leftmost occurrences. Since each leftmost occurrence crosses or begins at a phrase boundary of the LZ77 parsing of T, we conclude that $z \geq \frac{1}{4}\widetilde{m}$, and therefore $\mathrm{size}(v) \leq \mathrm{size}(\widetilde{v}) = \widetilde{m} \leq 4z$.

The reasoning above shows that once a root-to-leaf path encounters a light edge connecting a node u at depth at least $\lfloor \frac{1}{2}\ell \rfloor$ to its child v, we have $\mathrm{size}(v) \leq 4z$. The number of remaining light edges on the path is at most $\log(\mathrm{size}(v)) \leq 2 + \log z$ by the classic bound applied to the subtree of $\mathcal{T}$ rooted at v. ∎

Theorem III.4. *Every string of length n satisfies $r = \mathcal{O}(z \log z \max(1, \log \frac{n}{z \log z}))$.*

Proof: To obtain tighter bounds on the number of irreducible LCP values in $[\ell\,..\,2\ell)$, we consider three cases:

1) $\ell \leq \log z$. We repeat the proof of Lemma III.1, except that we observe that the number of light edges on each root-to-leaf path in $\mathcal{T}$ is bounded by ℓ. Thus, the number of irreducible LCP values in $[\ell\,..\,2\ell)$ is $\mathcal{O}(z\ell)$.
2) $\log z < \ell \leq \frac{n}{z}$. We use the bound of Lemma III.3.
3) $\frac{n}{z} < \ell$. We repeat the proof of Lemma III.3, except that we observe that $|\mathcal{S}_{3\ell}| \leq n$. Thus, the number of irreducible LCP values in $[\ell\,..\,2\ell)$ is $\mathcal{O}(\frac{n \log z}{\ell})$.

The above upper bounds for $\ell = 2^i$, $i \in [0\,..\,\lfloor \log n \rfloor]$, yield

$$r \leq \sigma + \sum_{i=0}^{\lfloor \log n \rfloor} \left| \left\{ j \in [2\,..\,n] : \substack{\mathrm{LCP}[j] \in [2^i\,..\,2^{i+1}),\\ \mathrm{BWT}[j] \neq \mathrm{BWT}[j-1]} \right\} \right| =$$

$$= \mathcal{O}\left(\sigma + \sum_{i=0}^{\lfloor \log \log z \rfloor} z 2^i + \sum_{i=\lfloor \log \log z \rfloor + 1}^{\lfloor \log \frac{n}{z} \rfloor} z \log z + \sum_{i=\lfloor \log \frac{n}{z} \rfloor + 1}^{\lfloor \log n \rfloor} \frac{n \log z}{2^i} \right)$$

$$= \mathcal{O}\left(\sigma + z \log z + z \log z \max\left(0, \log \tfrac{n}{z \log z}\right) + z \log z \right)$$

$$= \mathcal{O}\left(z \log z \max\left(1, \log \tfrac{n}{z \log z}\right) \right).$$

∎

C. Upper Bound in Terms of δ

Let $\delta = \max_{m=1}^{n} \frac{1}{m}|\mathcal{S}_m|$ denote the *substring complexity* of T [37]. Note that letting $\delta = \sup_{m=1}^{\infty} \frac{1}{m}|\mathcal{S}_m|$ is equivalent, because $|\mathcal{S}_m| \leq n$ holds for $m \geq 1$, which implies $\frac{1}{m}|\mathcal{S}_m| \leq 1 \leq |\mathcal{S}_1|$ for $m \geq n$. We start by noting that $\delta \leq z$ because $|\mathcal{S}_m| \leq mz$ holds for every $m \geq 1$, as observed in the proof of Lemma III.1. Furthermore, $|\mathcal{S}_m| \leq m\delta$ holds by definition of δ, so we can replace z with δ in the proof of Lemma III.1.

To adapt the proof Lemma III.3, we need the following fact that generalizes the observation that $T^\infty[1\,..)$ contains at most $2z\ell$ positions covered by the leftmost occurrence of a substring from $\mathcal{S}_\ell$. This is easy to see since the LZ77 parsing yields a set of positions (phrase boundaries) in T. The substring complexity δ does not provide such information. As shown below, this property nevertheless generalizes to δ. Its proof is a straightforward modification of the argument used in [37, Lemma 10]. For completeness, below we write down the proof, with technical details tailored to our notation (e.g., $\mathcal{S}_\ell$ defined in terms of T^∞ rather than T).

Lemma III.5 (based on [37])**.** *For any positive integer ℓ, the total number of positions in $T^\infty[1\,..)$ covered by the leftmost occurrences of strings from $\mathcal{S}_\ell$ is at most $3\delta\ell$.*

Proof: Let C denote the set of positions in $T^\infty[1\,..)$ covered by the leftmost occurrences of strings from $\mathcal{S}_\ell$, and let $C' = C \setminus [1\,..\,\ell)$. For any $i \in C'$ denote $S_i = T^\infty[i-\ell+1\,..\,i+\ell]$, and let $\mathcal{S} = \{S_i : i \in C'\} \subseteq \mathcal{S}_{2\ell}$. We will show that $|\mathcal{S}| = |C'|$. Let $i \in C'$. Observe first that, due to $i \geq \ell$, the fragment S_i is entirely contained in $T^\infty[1\,..)$. Furthermore, by definition, S_i contains the leftmost occurrence of some $S \in \mathcal{S}_\ell$. Thus, this occurrence of S_i in $T^\infty[1\,..)$ must also be the leftmost one in $T^\infty[1\,..)$. Consequently, the substrings S_i for $i \in C'$ are distinct.

We have thus shown that $|C'| = |\mathcal{S}| \leq |\mathcal{S}_{2\ell}|$. Since $|\mathcal{S}_{2\ell}| \leq 2\delta\ell$ holds by definition of δ, we obtain $|C| < |C'| + \ell \leq |\mathcal{S}_{2\ell}| + \ell \leq (2\delta+1)\ell \leq 3\delta\ell$. ∎

Lemma III.6. *For every $\ell \in [1\,..\,n]$, the number of irreducible LCP values in $[\ell\,..\,2\ell)$ is $\mathcal{O}(\delta \log \delta)$.*

Proof: Compared to the proof of Lemma III.3, we use the bound $|\mathcal{S}_{3\ell}| \leq 3\ell\delta$. The only other modification needed is that for every light edge connecting a node u at depth at least $\lfloor \frac{1}{2}\ell \rfloor$ to its child v, we need to prove $\mathrm{size}(v) = \mathcal{O}(\delta)$.

Let $\widetilde{m} \geq \mathrm{size}(v)$ be defined as in the proof of Lemma III.3. Recall that we have identified at least $\frac{\widetilde{m}}{4}$ strings in $\mathcal{S}_\ell$ whose leftmost occurrences in $T^\infty[1\,..)$ are disjoint. By Lemma III.5, there are at most 3δ such substrings. Thus, $\mathrm{size}(v) \leq \widetilde{m} \leq 12\delta$. ∎

By replacing the thresholds $\log z$ and $\frac{n}{z}$ with $\log \delta$ and $\frac{n}{\delta}$, respectively, in the proof of Theorem III.4, we immediately obtain a bound in terms of δ.

Theorem III.7. *Every string of length n satisfies $r = \mathcal{O}(\delta \log \delta \max(1, \log \frac{n}{\delta \log \delta}))$.*

Note that the trivial upper bound $r = \mathcal{O}(n)$ is tighter if $\delta \log \delta > n$. In Section IV, we show that a combination of these two upper bounds is asymptotically optimal. For this, we construct tight examples in which the values δ cover the whole spectrum between $\mathcal{O}(1)$ and $\Omega(n)$.

By combining the Theorem III.7 with known properties of the substring complexity δ, we obtain the first bound relating the number of BWT runs in the string and its reverse. No such bounds (even polynomial in n) were known before.

Corollary III.8. *If r and $\bar{r}$ denote the number of runs in the BWT of a length-n text and its reverse, respectively, then $\bar{r} = \mathcal{O}(r \log r \max(1, \frac{n}{r \log r}))$.*

Proof: Since the value of δ is the same for the text and its reverse, we obtain $\bar{r} = \mathcal{O}(\delta \log \delta \max(1, \log \frac{n}{\delta \log \delta}))$. Combining [34, Theorem 3.9] and [37, Lemma 2] gives $\delta \leq r$. Consequently, we obtain $\bar{r} = \mathcal{O}(r \log r \max(1, \frac{n}{r \log r}))$. ■

Our technique also lets us strengthen the bound of Theorem II.2 on the sum of irreducible LCP values.

Theorem III.9 (♠). *The sum of all irreducible LCP values is $\mathcal{O}(n \log \delta)$.*

Due to $\delta \leq r$, the presented upper bound is always (asymptotically) at least as strong as the bound of Theorem II.2; furthermore, it can be strictly stronger since $\log \delta = o(\log r)$ is possible when $\delta = \log^{o(1)} n$. In Section IV, we construct strings proving tightness of the new bound.

IV. Lower Bounds

In this section, we present examples showing asymptotic tightness of the upper bounds in Section III-C. We give two constructions, corresponding to the bound of Theorem III.7 and the trivial bound $r = \mathcal{O}(n)$, respectively.

For $\ell \geq 1$, let $\mathrm{bin}_\ell(x) \in \{\mathtt{0}, \mathtt{1}\}^\ell$ be the binary representation of $x \in [0 \,..\, 2^\ell)$.

Lemma IV.1 (♠). *For all integers $\ell \geq 2$ and $K \geq 1$, the length n, the substring complexity δ, and the number of runs r in the BWT of a string $T_{\ell,K} \in \{\$, \mathtt{0}, \mathtt{1}, \mathtt{2}\}^+$, defined with*

$$T_{\ell,K} = \left(\bigodot_{k=0}^{K-1} \bigodot_{i=0}^{2^\ell - 1} \left(2^{2^k \ell} \cdot \mathrm{bin}_\ell(i) \right) \right) \cdot \$,$$

satisfy $n = \Theta(2^{K+\ell}\ell)$, $\delta = \Theta(2^\ell)$, and $r = \Omega(2^\ell \ell K)$.

Lemma IV.2 (♠). *For all integers $\ell \geq 2$ and $\Delta \in \Omega(2^\ell) \cap \mathcal{O}(2^\ell \ell)$, the length n, the substring complexity δ, and the number of runs r in the BWT of a string $T'_{\ell,\Delta} \in \{\$_1, \ldots, \$_\Delta, \mathtt{0}, \mathtt{1}, \mathtt{2}\}^+$, defined with*

$$T'_{\ell,\Delta} = \left(\bigodot_{i=0}^{2^\ell - 1} \left(2^\ell \cdot \mathrm{bin}_\ell(i) \right) \right) \cdot \$_1 \$_2 \cdots \$_\Delta,$$

satisfy $n = \Theta(2^\ell \ell)$, $\delta = \Theta(\Delta)$, and $r = \Omega(2^\ell \ell)$.

Combining Lemmas IV.1 and IV.2, we obtain the following

Theorem IV.3 (♠). *For every $N \geq 1$ and $\Delta \in [1 \,..\, N]$, there is a string T whose length n, substring complexity δ, and number of runs r in the BWT satisfy $n = \Theta(N)$, $\delta = \Theta(\Delta)$, and $r = \Theta(\min(n, \delta \log \delta \max(1, \log \frac{n}{\delta \log \delta})))$.*

The same strings show that our upper bound $\mathcal{O}(n \log \delta)$ on the sum of irreducible LCP values is also tight.

Theorem IV.4 (♠). *For every $N \geq 1$ and $\Delta \in [1 \,..\, N]$, there is a string T whose length n, substring complexity δ, and sum r_Σ of irreducible LCP values satisfy $n = \Theta(N)$, $\delta = \Theta(\Delta)$, and $r_\Sigma = \Theta(n \log \delta)$.*

V. Converting LZ77 to Run Length BWT

In this section, we outline an algorithm that, given the LZ77 parsing of a text $T \in \Sigma^n$, computes its run-length compressed BWT in $\mathcal{O}(z \,\mathrm{polylog}\, n)$ time. We start with an overview that explains the key concepts. Then, we present two new data structures utilized in our algorithm: the compressed string synchronizing set (Section V-A) and the compressed wavelet tree (Section V-B). The conversion algorithm is then presented in Section V-C.

For any substring Y of T^∞, we define $\mathrm{lpos}(Y) = \min\{i \in [1 \,..\, n] : T^\infty[i \,..\, i + |Y|) = Y\}$. If Y is a substring of $\overline{T}^\infty$, we define $\overline{\mathrm{lpos}}(Y)$ by replacing T^∞ in the definition with $\overline{T}^\infty$. We say that Y is *left-maximal* if there exist distinct symbols $a, b \in \Sigma$ such that the strings aY and bY are also substrings of T^∞. The following definition, assuming $\Sigma \cap \mathbb{N} = \emptyset$, plays a key role in our construction.

Definition V.1 (BWT modulo ℓ)**.** *Let $T \in \Sigma^n$, $\ell \geq 1$, and $Y_i = T^\infty[\mathrm{SA}[i] \,..\, \mathrm{SA}[i] + \ell)$ for $i \in [1 \,..\, n]$. We define the string $\mathrm{BWT}_\ell \in (\Sigma \cup \mathbb{N})^n$, called the* BWT modulo ℓ *(of T), as follows. For $i \in [1 \,..\, n]$,*

$$\mathrm{BWT}_\ell[i] = \begin{cases} \mathrm{lpos}(Y_i) & \text{if } Y_i \text{ is left-maximal,} \\ \mathrm{BWT}[i] & \text{otherwise.} \end{cases}$$

The algorithm runs in $k = \lceil \log n \rceil$ rounds. For $q \in [0 \,..\, k)$, the input to the qth round is $\mathrm{RL}(\mathrm{BWT}_\ell)$, where $\ell = 2^q$, and the output is $\mathrm{RL}(\mathrm{BWT}_{2\ell})$. At the end of the algorithm, we have $\mathrm{RL}(\mathrm{BWT}_{2^k}) = \mathrm{RL}(\mathrm{BWT})$ because $X \in \mathcal{S}_{2^k}$ is never left-maximal for $2^k \geq n$.

Informally, in round q, we are given a (run-length compressed) subsequence of BWT that can be determined based on sorting the suffixes only up to their prefixes of length 2^q. $\mathrm{BWT}_\ell[b \,..\, e] \in \Sigma^+$ implies $\mathrm{BWT}_{\ell+1}[b \,..\, e] \in \Sigma^+$ (because a prefix of a left-maximal substring is left-maximal). Hence, these subsequences need not be modified until the end of the algorithm (except possibly merging their runs with adjacent runs). For the remaining positions, BWT_ℓ identifies the (leftmost occurrences of) substrings to be inspected in the qth round with the aim of replacing their corresponding runs in BWT_ℓ with previously unknown BWT symbols (as defined in $\mathrm{BWT}_{2\ell}$).

We call a block $\mathrm{BWT}[b \,..\, e]$ *uniform* if all symbols in $\mathrm{BWT}[b \,..\, e]$ are equal, and *non-uniform* otherwise. The following lemma ensures feasibility of the above construction.

Lemma V.2. *For any $\ell \geq 1$, it holds $|\mathrm{RL}(\mathrm{BWT}_\ell)| < 2r$.*

Proof: Denote $\mathrm{RL}(\mathrm{BWT}_\ell) = ((c_1, \lambda_1), \ldots, (c_h, \lambda_h))$, letting $\lambda_0 = 0$. By definition of BWT_ℓ, if $c_i \in \mathbb{N}$, then the block $\mathrm{BWT}(\lambda_{i-1} \,..\, \lambda_i]$ is non-uniform. Thus, there are at most $r - 1$ runs of symbols from $\mathbb{N}$ in BWT_ℓ.

On the other hand, $c_i \in \Sigma$ and $c_j \in \Sigma$, with $i < j$, cannot both belong to the same run in BWT. If this was true, then either $c_{i+1} \in \Sigma$ (which implies $c_{i+1} = c_i$, contradicting the definition of $\mathrm{RL}(\mathrm{BWT}_\ell)$), or $c_{i+1} \in \mathbb{N}$, which is impossible

since then $\mathrm{BWT}(\lambda_i \,..\, \lambda_{i+1}]$ is non-uniform. Thus, there are at most r runs of symbols from Σ in BWT_ℓ. ∎

A. Compressed String Synchronizing Sets

Our algorithm builds on the notion of *string synchronizing sets*, recently introduced in [33]. Synchronizing sets are one of the most powerful techniques for sampling suffixes. As demonstrated in [36], in the uncompressed setting, they are the key in obtaining time-optimal solutions to many problems, and their further applications are still being discovered [3], [33]. In this section, we introduce a notion of *compressed string synchronizing sets*. Our construction is the first implementation of synchronizing sets in the compressed setting and thus of independent interest.

We start with the definition of basic synchronizing sets.

Definition V.3 (τ-synchronizing set [33])**.** *Let T be a string of length n, and let $\tau \in [1 \,..\, \lfloor \frac{n}{2} \rfloor]$. A set $\mathsf{S} \subseteq [1 \,..\, n-2\tau+1]$ is called a* τ-synchronizing set *of T if it satisfies the following* consistency *and* density *conditions:*

1) *If $T[i \,..\, i+2\tau) = T[j \,..\, j+2\tau)$, then $i \in \mathsf{S}$ holds if and only if $j \in \mathsf{S}$ (for $i, j \in [1 \,..\, n-2\tau+1]$),*
2) *$\mathsf{S} \cap [i \,..\, i+\tau) = \emptyset$ if and only if $\mathrm{per}(T[i \,..\, i+3\tau-2]) \leq \frac{1}{3}\tau$ (for $i \in [1 \,..\, n-3\tau+2]$).*

In most applications, we want to minimize $|\mathsf{S}|$. Observe that the Thue–Morse sequence T_{TM} [64] does not contain any *cube* (substring of the form W^3). Thus, by density condition, any synchronizing set S of the length-n prefix of T_{TM} satisfies $|\mathsf{S}| = \Omega\left(\frac{n}{\tau}\right)$ unless $n < 3\tau$. Therefore, in general, we cannot hope to achieve an upper bound improving in the worst case upon the following one.

Theorem V.4 ([33])**.** *For any string T of length n and parameter $\tau \in [1 \,..\, \lfloor \frac{n}{2} \rfloor]$, there exists a τ-synchronizing set S of size $|\mathsf{S}| = \mathcal{O}\left(\frac{n}{\tau}\right)$. Moreover, such S can be (deterministically) constructed from T in $\mathcal{O}(n)$ time.*

Storing S for compressible strings presents the following challenge: As shown in [46], a length-n prefix of T_{TM} satisfies $z = \mathcal{O}(\log n)$ and yet, as discussed above, every synchronizing set of T_{TM} satisfies $|\mathsf{S}| = \Omega\left(\frac{n}{\tau}\right)$. Thus, although $|\mathsf{S}|$ *can* be smaller than $\frac{n}{\tau}$, $z \ll n$ does not imply $|\mathsf{S}| \ll n$, preventing us from keeping plain S when $\tau = o(\frac{n}{z})$.

We thus exploit a different property of compressible strings: their substrings Y satisfy $\mathrm{lpos}(Y) \in \bigcup_{j=1}^{z} (e_j - |Y| \,..\, e_j]$, where e_j is the last position of the jth phrase in the LZ77 parsing of T. By consistency of S, it suffices to store $\bigcup_{j=1}^{z} \mathsf{S} \cap (e_j - 2\tau \,..\, e_j]$. To check if $i \in \mathsf{S}$, we then locate $i' = \mathrm{lpos}(T[i \,..\, i+2\tau))$ and check if $i' \in \bigcup_{j=1}^{z} \mathsf{S} \cap (e_j - 2\tau \,..\, e_j]$. This motivates the following (more general) definition.

Definition V.5 (Compressed τ-synchronizing set)**.** *Let S be a τ-synchronizing set of string $T[1 \,..\, n]$ for some $\tau \in [1 \,..\, \lfloor \frac{n}{2} \rfloor]$, and, for every $j \in [1 \,..\, z]$, let e_j denote the last position of the jth phrase in the LZ77 parsing of T. For $k \in \mathbb{N}_{\geq 2}$, we define the* compressed representation *of S as*

$$\mathrm{comp}_k(\mathsf{S}) := \bigcup_{j=1}^{z} \mathsf{S} \cap \Big(e_j - k\tau \,..\, e_j + k\tau\Big].$$

Next, we prove that every text T has a synchronizing set S with a small compressed representation, and we show how to efficiently compute such S from the LZ77 parsing of T.

1) The Nonperiodic Case: We initially assume that $\mathrm{per}(T[i \,..\, i+\tau)) > \frac{1}{3}\tau$ holds for all $i \in [1 \,..\, n-\tau+1]$.

Theorem V.6. *Let T be a string of length n and let $\tau \in [1 \,..\, \lfloor \frac{n}{2} \rfloor]$. Assume that $\mathrm{per}(T[i \,..\, i+\tau)) > \frac{1}{3}\tau$ holds for all $i \in [1 \,..\, n-\tau+1]$. Then, for every $k \in \mathbb{N}_{\geq 2}$, there exists a τ-synchronizing set S of T with $\mathrm{comp}_k(\mathsf{S}) \leq 12kz$.*

Proof: Let $h : \mathcal{S}_\tau \to [0, 1]$ be a function mapping strings to real values in $[0, 1]$ independently and uniformly at random. Note that h is collision-free almost surely (with probability 1). Let us define $\mathrm{id} : [1 \,..\, n-\tau+1] \to [0, 1]$ with $\mathrm{id}(i) = h(T[i \,..\, i+\tau))$. Observe that (almost surely) id is an *identifier function*, that is, $\mathrm{id}(i) = \mathrm{id}(j)$ holds if and only if $T[i \,..\, i+\tau) = T[j \,..\, j+\tau)$. In [33, Lemma 8.2], it is proved that then

$$\mathsf{S} := \{i \in [1 \,..\, n-2\tau+1] : \min\{\mathrm{id}(j) : j \in [i \,..\, i+\tau]\} \in \{\mathrm{id}(i), \mathrm{id}(i+\tau)\}\}$$

is a τ-synchronizing set of T. Moreover, $\mathbb{E}[|\mathsf{S}|] = \mathcal{O}\left(\frac{n}{\tau}\right)$. To see this, observe that, for $j, j' \in [i \,..\, i+\tau]$, $\mathrm{id}(j) = \mathrm{id}(j')$ implies $|j' - j| > \frac{1}{3}\tau$ (otherwise, assuming $j < j'$, we have $\mathrm{per}(T[j \,..\, j' + \tau)) \leq \frac{1}{3}\tau$, which contradicts $\mathrm{per}(T[j \,..\, j+\tau)) > \frac{1}{3}\tau$). Thus, $T[i \,..\, i+2\tau-1]$ contains $d \geq \frac{\tau}{3}$ distinct length-τ substrings. Since each has equal chance of having the smallest id, we have $\mathbb{P}[i \in \mathsf{S}] \leq \frac{2}{d} \leq \frac{6}{\tau}$, and consequently, by linearity of expectation, $\mathbb{E}[|\mathsf{S}|] \leq \frac{6n}{\tau}$. More generally, $\mathbb{E}[|\mathsf{S} \cap (i \,..\, i+\ell]|] \leq \frac{6\ell}{\tau}$, and therefore

$$\mathbb{E}\big[\,|\mathrm{comp}_k(\mathsf{S})|\,\big] = \mathbb{E}\left[\left|\bigcup_{j=1}^{z} \mathsf{S} \cap (e_j - k\tau \,..\, e_j + k\tau]\right|\right] \leq \sum_{j=1}^{z} \mathbb{E}\big[\,|\mathsf{S} \cap (e_j - k\tau \,..\, e_j + k\tau]|\,\big] \leq 12kz.$$

In particular, $|\mathrm{comp}_k(\mathsf{S})| \leq 12kz$ holds for some h. ∎

The above proof does not lead to an efficient algorithm for constructing S as it relies on the random assignment of unique names to *all* substrings in $\mathcal{S}_\tau$ and, since $|\mathcal{S}_\tau| = \Theta(z\tau)$ holds in the worst case, we cannot hope to achieve $\mathcal{O}(z \,\mathrm{polylog}\, n)$ time this way. Next, we prove that assigning unique names to all elements of $\mathcal{S}_\tau$ is in fact not necessary.

Lemma V.7. *Let T be a string of length n and let $\tau \in [1 \,..\, \lfloor \frac{n}{2} \rfloor]$. Assume that $\mathrm{per}(T[i \,..\, i+\tau)) > \frac{1}{3}\tau$ holds for all $i \in [1 \,..\, n-\tau+1]$. For $i \in [1 \,..\, n-\tau+1]$, let $\mathrm{id}(i) :=$*

$h(T[i\,..\,i+\tau))$, *where* $h : \mathcal{S}_\tau \to [0,1]$ *assigns independent and uniformly random values.*

If $\kappa = \max(1, \frac{\tau}{3c\ln n})$ *for* $c > 1$*, then, with probability at least* $1 - n^{1-c}$*, all positions* $i \in [1\,..\,n-2\tau+1]$ *satisfy* $\min\{\mathrm{id}(j) : j \in [i\,..\,i+\tau]\} \leq \frac{1}{\kappa}$.

Proof: Recall from the proof of Theorem V.6 that $T[i\,..\,i+2\tau-1]$ contains $d \geq \frac{\tau}{3}$ distinct length-τ substrings. Since the values of h are independent and uniformly distributed, we have $\mathbb{P}\left[\min\{\mathrm{id}(j) : j \in [i\,..\,i+\tau]\} > \frac{1}{\kappa}\right] = \left(1-\frac{1}{\kappa}\right)^d \leq \exp(-\frac{d}{\kappa}) \leq \exp(-\frac{\tau}{3\kappa})$. The probability above is trivially 0 if $\kappa = 1$, so we can bound it by n^{-c} if $\kappa = \max(1, \frac{\tau}{3c\ln n})$. Taking the union bound across all positions i, we derive the final claim. ∎

If κ is set as in the above lemma for a sufficiently large constant c, then, with high probability, each window contains at least one substring with a "small" identifier $\leq \frac{1}{\kappa}$. The "large" identifiers of other substrings are never used in the construction of the synchronizing set S and hence need not be specified. Consequently, to carry out the randomized construction of S using Theorem V.6, rather than choosing a random function $h : \mathcal{S}_\tau \to [0,1]$, it suffices to select a random subset $\mathcal{S}_{\mathrm{sample}} \subseteq \mathcal{S}_\tau$ with rate $\frac{1}{\kappa}$ (each string in $\mathcal{S}_\tau$ is included in $\mathcal{S}_{\mathrm{sample}}$ independently with probability $\frac{1}{\kappa}$) and then construct a uniformly random function $h_{\mathrm{sample}} : \mathcal{S}_{\mathrm{sample}} \to [0, \frac{1}{\kappa}]$ (mapping strings in $\mathcal{S}_{\mathrm{sample}}$ to real values in $[0, \frac{1}{\kappa}]$ independently and uniformly at random).

Clearly, the element-wise sampling of $\mathcal{S}_\tau$ is equivalent to sampling the set P_{left} containing the starting positions of the leftmost occurrences of strings in $\mathcal{S}_\tau$. Sampling P_{left} directly is still hard, though. The key observation is that instead of P_{left} (which is difficult to compute), we can sample (at the same rate) elements of its superset $P_{\mathrm{close}} := \bigcup_{j=1}^{z}(e_j - \tau\,..\,e_j]$, which is readily available, and yet still sufficiently small. Let $P'_{\mathrm{sample}} \subseteq P_{\mathrm{close}}$ be a resulting sample. We then define the desired sample with $P_{\mathrm{sample}} := P'_{\mathrm{sample}} \cap P_{\mathrm{left}}$. Crucially, however, we have

$$\mathbb{E}\left[|P'_{\mathrm{sample}}|\right] = \tfrac{1}{\kappa}|P_{\mathrm{close}}| \leq \tfrac{3c\ln n}{\tau} \cdot z\tau = \mathcal{O}(z\log n).$$

To finish the construction, it suffices to pick a random function $h_{\mathrm{sample}} : \mathcal{S}_{\mathrm{sample}} \to [0, \frac{1}{\kappa}]$ to obtain $\mathrm{id}(i) := h_{\mathrm{sample}}(T[i\,..\,i+\tau))$ (letting $\mathrm{id}(i) = 1$ if $T[i\,..\,i+\tau) \notin \mathcal{S}_{\mathrm{sample}}$). Then, by Lemma V.7 and the discussion above, using h_{sample} is with high probability equivalent to using a uniformly random function $h : \mathcal{S}_\tau \to [0,1]$ during the construction behind Theorem V.6. Moreover, we can also detect failures (that $\min\{\mathrm{id}(j) : j \in [i\,..\,i+\tau]\} = 1$ for some i), so the algorithm is Las-Vegas randomized.

Theorem V.8 (♠). *Let* T *be a string of length* n *and let* $\tau \in [1\,..\,\lfloor\frac{n}{2}\rfloor]$*. Assume that* $\mathrm{per}(T[i\,..\,i+\tau)) > \frac{1}{3}\tau$ *holds for all* $i \in [1\,..\,n-\tau+1]$*. There exists a Las-Vegas randomized algorithm that, for any constant* $k \in \mathbb{N}_{\geq 2}$*, given the LZ77 parsing of* T*, constructs in* $\mathcal{O}(z\log^5 n)$ *time a compressed representation* $\mathrm{comp}_k(\mathsf{S})$ *of a* τ*-synchronizing set* S *of* T *satisfying* $\mathrm{comp}_k(\mathsf{S}) \leq 24kz$.

2) The General Case: Periodic fragments are handled similarly as in [33]. This yields the following two results, which constitute the main outcome of this section.

Theorem V.9 (♠). *Let* T *be a string of length* n *and let* $\tau \in [1\,..\,\lfloor\frac{n}{2}\rfloor]$*. For any* $k \in \mathbb{N}_{\geq 2}$*, there exists a* τ*-synchronizing set* S *of* T *satisfying* $\mathrm{comp}_k(\mathsf{S}) \leq 36kz$.

Theorem V.10 (♠). *Let* T *be a string of length* n *and let* $\tau \in [1\,..\,\lfloor\frac{n}{2}\rfloor]$*. There exists a Las-Vegas randomized algorithm that, for any constant* $k \in \mathbb{N}_{\geq 2}$*, given the LZ77 parsing of* T*, constructs in* $\mathcal{O}(z\log^5 n)$ *time a compressed representation* $\mathrm{comp}_k(\mathsf{S})$ *of a* τ*-synchronizing set* S *of* T *satisfying* $|\mathrm{comp}_k(\mathsf{S})| \leq 72kz$.

B. Compressed Wavelet Trees

Along with string synchronizing sets, wavelet trees [28], originally invented for text indexing, play a central role in our algorithm. Unlike virtually all prior applications of wavelet trees, ours uses a sequence of very long strings (up to $\Theta(n)$ symbols). This approach is feasible since all strings are substrings of the text, which is stored in the LZ77-compressed form. In this section, we describe this novel variant of wavelet trees, dubbed here *compressed wavelet trees*. In particular, we prove the upper bound on their size, describe an efficient construction from the LZ77-compressed text, and show how to augment them to support some fundamental queries.

Let Σ be an alphabet of size $\sigma \geq 1$. Consider a string $W[1\,..\,m]$ over the alphabet Σ^ℓ so that W is a sequence of $m \geq 0$ strings of length $\ell \geq 0$ over the alphabet Σ. The wavelet tree of W is defined as follows. Let $\mathcal{T}$ be a perfect σ-ary rooted tree of height ℓ with edges labelled by symbols of Σ such that, for every $Y \in \Sigma^\ell$, there exists a root-to-leaf path in $\mathcal{T}$ whose edges are labelled $Y[1], \ldots, Y[\ell]$. We define the label of a node as the concatenation of the edge labels on the path from the root. For $X \in \Sigma^d$, where $d \in [0\,..\,\ell]$, by v_X we denote the node of $\mathcal{T}$ labelled X. We let $V(\mathcal{T}) = \bigcup_{d=0}^{\ell}\{v_X : X \in \Sigma^d\}$ denote the node set of $\mathcal{T}$.

With each node $v_X \in V(\mathcal{T})$ we associate an increasing sequence $I_X[1\,..\,h]$ of *primary indices* such that

$$\{I_X[i] : i \in [1\,..\,h]\} = \{j \in [1\,..\,m] : W[j][1\,..\,|X|] = X\}.$$

Based on I_X, we define $B_X \in \Sigma^*$ such that, for $i \in [1\,..\,h]$,

$$B_X[i] = W[I_X[i]][|X|+1],$$

if $|X| < \ell$ and $B_X = \varepsilon$ if $|X| = \ell$. In other words, B_X is a string containing the symbol at position $|X|+1$ for each string of W that is prefixed by X. Importantly, the symbols in B_X occur in the same order as these strings occur in W.

As typically done in the applications of wavelet trees, we only explicitly store the strings B_X. The values of primary indices I_X are retrieved using additional data structures, based on the following observation.

Lemma V.11 ([28]). *Let $X \in \Sigma^d$, where $d \in [0\,..\,\ell)$. For every $c \in \Sigma$ and $j \in [1\,..\,|I_{Xc}|]$, we have $I_{Xc}[j] = I_X[i]$, where $B_X[i]$ is the jth occurrence of c in B_X.*

We define the *compressed wavelet tree* $\mathcal{T}_c$ of W as the wavelet tree of W in which all strings B_X have been run-length compressed and, with the exception of $\{v_\varepsilon\} \cup \{v_{W[i]}\}_{i=1}^m$, all nodes v_X satisfying $|\mathrm{RL}(B_X)| \le 1$, have been removed (the unary paths are collapsed into single edges). The shape and edge labels of the resulting tree are identical to the compact trie of strings $W[1], \ldots, W[m]$.

We store edge labels of $\mathcal{T}_c$ as pointers to substrings in W. We assume that values of ℓ and m fit into a single machine word so that each edge of $\mathcal{T}_c$ and each element of $\mathrm{RL}(B_X)$ can be encoded in $\mathcal{O}(1)$ space. Since $|\mathrm{RL}(B_Y)| \ge 1$ holds for every internal node $v_Y \in V(\mathcal{T}_c)$, and unless $|V(\mathcal{T}_c)| = 1$, each leaf v_Z in $\mathcal{T}_c$ can be injectively mapped to an element of $\mathrm{RL}(B_{Z'})$ for the parent $v_{Z'}$ of v_Z, the space to store $\mathcal{T}_c$ is dominated by the run-length compressed strings B_X, i.e., $\mathcal{T}_c$ needs $\mathcal{O}(1 + \sum_{v_X \in V(\mathcal{T}_c)} |\mathrm{RL}(B_X)|)$ space.

Theorem V.12. *Let W be a non-empty sequence of equal-length strings and let $\mathcal{T}_c$ be its compressed wavelet tree. Then, $\sum_{v_X \in V(\mathcal{T}_c)} |\mathrm{RL}(B_X)| = \mathcal{O}(1 + |\mathrm{RL}(W)| \log |\mathrm{RL}(W)|)$.*

Proof: Let $m = |W|$, $k = |\mathrm{RL}(W)| \le m$, and $k' = |\{W[i] : i \in [1\,..\,m]\}| \le k$. Due to $|V(\mathcal{T}_c)| \le 2k' = \mathcal{O}(k)$, we can focus on nodes $v_X \in V(\mathcal{T}_c)$ such that $|\mathrm{RL}(B_X)| \ge 2$.

The proof resembles that of Lemma III.1. With each $X \in \Sigma^*$ such that $|\mathrm{RL}(B_X)| \ge 2$, we associate $|\mathrm{RL}(B_X)| - 1$ units of cost and charge them to individual elements of W. We then show that each run in $\mathrm{RL}(W)$ is in total charged at most $2 \log k'$ units of cost. Consequently,

$$\sum_{\substack{v_X \in V(\mathcal{T}_c) \\ |\mathrm{RL}(B_X)| \ge 2}} |\mathrm{RL}(B_X)| \le 4k \log k' = \mathcal{O}(k \log k).$$

Consider $X \in \Sigma^d$ with $|\mathrm{RL}(B_X)| \ge 2$; note that $d < \ell$. Let $\mathrm{RL}(B_X) = ((c_1, \lambda_1), \ldots, (c_h, \lambda_h))$. Observe that if we let $p_0 = I_X[\lambda_i]$ and $p_1 = I_X[\lambda_i + 1]$ for some $i \in [1\,..\,h)$, then $W[p_0][d+1] = c_i \ne c_{i+1} = W[p_1][d+1]$. Moreover, $B_X[\lambda_i] \ne B_X[\lambda_i + 1]$ implies $W[p_0 + 1] \ne W[p_0]$ and $W[p_1 - 1] \ne W[p_1]$. The ith unit of cost is charged to $W[p_t]$, where $t \in \{0, 1\}$ is chosen depending on the sizes of subtrees of $\mathcal{T}_c$ rooted at the children of v_X, so that the subtree containing $v_{W[p_t]}$ has at most as many leaves as the subtree containing $v_{W[p_{1-t}]}$.

Now, consider a run $W[b\,..\,b'] = Y^\delta$ in $\mathrm{RL}(W)$. For a single depth d, the run could be charged at most twice, with at most one unit assigned to $W[b]$ due to $p_1 = b$ and at most one unit assigned to $W[b']$ due to $p_0 = b'$, both for $X = Y[1\,..\,d]$. Moreover, note that the subtree size on the path from v_Y to the root v_ε of $\mathcal{T}_c$ doubles for every depth d for which the run was charged. Thus, the total charge of the run is at most $2 \log k'$ units. ∎

Let $W[1\,..\,m]$ be a sequence of substrings of T^∞ of the same length ℓ. Observe that if we have access to T, then the sequence W can be compactly encoded in $\mathcal{O}(1 + |\mathrm{RL}(W)|)$ space. Namely, it suffices to store the length ℓ and the sequence $\mathrm{RL}((\mathrm{lpos}(W[i]))_{i \in [1\,..\,m]})$. If W is a sequence of substrings of $\overline{T}^\infty$, we can similarly encode it using $\mathrm{RL}((\overline{\mathrm{lpos}}(W[i]))_{i \in [1\,..\,m]})$.

The key operation that we want to support on $\mathcal{T}_c$, given a pointer to $v_X \in V(\mathcal{T}_c)$ and an integer $q \in [1\,..\,|I_X|]$, is to compute the value $I_X[q]$. The following theorem shows that given the compact encoding of W and the LZ77 parsing of T, the compressed wavelet tree of W supporting these *primary index queries* can be constructed efficiently.

Theorem V.13 (♠). *Given the LZ77 parsing of $T[1\,..\,n]$ and a sequence $W[1\,..\,m]$ of $m \le n$ same-length substrings of T^∞ (resp. $\overline{T}^\infty$), represented as $\mathrm{RL}((\mathrm{lpos}(W[i]))_{i \in [1\,..\,m]})$ (resp. $\mathrm{RL}((\overline{\mathrm{lpos}}(W[i]))_{i \in [1\,..\,m]})$), the compressed wavelet tree of W, supporting primary index queries in $\mathcal{O}(\log^4 n)$ time, can be constructed in $\mathcal{O}((z + |\mathrm{RL}(W)|) \log^2 n)$ time.*

C. The Algorithm

We are now ready to show how to construct the sequences $\mathrm{RL}(\mathrm{BWT}_\ell)$, where $\ell = 2^q$ for $q \in [0\,..\,\lceil \log n \rceil]$.

For small ℓ, constructing $\mathrm{RL}(\mathrm{BWT}_\ell)$ reduces to sorting and computing frequencies of length-$\Theta(\ell)$ substrings of T^∞.

Proposition V.14 (♠). *Let $\ell = \mathcal{O}(1)$. Given the LZ77 parsing of $T[1\,..\,n]$, the sequence $\mathrm{RL}(\mathrm{BWT}_\ell)$ can be constructed in $\mathcal{O}(z \log^4 n)$ time.*

Let $q \ge 4$. We show how to compute $\mathrm{RL}(\mathrm{BWT}_{2\ell})$, given the LZ77 parsing of T and $\mathrm{RL}(\mathrm{BWT}_\ell)$. The main idea of the algorithm is as follows.

Let S be a τ-synchronizing set of T, where $\tau = \lfloor \frac{\ell}{3} \rfloor$. As noted earlier, $\mathrm{BWT}_\ell[j] \in \Sigma$ implies $\mathrm{BWT}_{2\ell}[j] \in \Sigma$. Let $\mathrm{BWT}_\ell[y\,..\,y'] \in \mathbb{N}^+$ be a run in BWT_ℓ. By definition of BWT_ℓ, the suffixes of T^∞ starting at positions $i \in \mathrm{SA}[y\,..\,y']$ share a common prefix of length $\ell \ge 3\tau$. Thus, assuming that $\mathsf{S} \cap [i\,..\,i+\tau) \ne \emptyset$ holds for all $i \in \mathrm{SA}[y\,..\,y']$ (the periodic case is handled separately), by the consistency of S, all text positions $i \in \mathrm{SA}[y\,..\,y']$ share a common offset Δ with $i + \Delta = \min(\mathsf{S} \cap [i\,..\,i+\tau))$. This lets us deduce the order of length-2ℓ prefixes $T[i\,..\,i+2\ell)$ based on the order of strings $T[i+\Delta\,..\,i+2\ell)$ starting at synchronizing positions. For this, from the sorted list of fragments $T[s\,..\,s+2\ell-\Delta)$ across $s \in \mathsf{S}$, we extract, using a wavelet tree, those preceded by $T[i\,..\,i+\Delta)$ (a prefix common to $T^\infty[i\,..)$ for $i \in \mathrm{SA}[y\,..\,y']$). Importantly, the synchronizing positions s sharing $T[s-\ell\,..\,s+2\ell)$ can be processed together; hence, by Theorem V.9, it suffices to use $\mathcal{O}(z)$ distinct substrings.

We formalize these ideas as follows. Let

$$\mathsf{R} = \left\{ i \in [1\,..\,n - 3\tau + 2] : \mathrm{per}\,(T[i\,..\,i+3\tau-2]) \le \tfrac{1}{3}\tau \right\}.$$

The description of the algorithm is divided into the nonperiodic case (when $\mathsf{R} = \emptyset$) and the general case.

1) The Nonperiodic Case: Let $(s'_i)_{i\in[1\,..\,|\mathsf{S}|]}$ be the sequence containing all positions in S such that $i < j$ holds if

- $T^\infty[s'_i\,..\,s'_i+7\tau) \prec T^\infty[s'_j\,..\,s'_j+7\tau)$, or
- $T^\infty[s'_i\,..\,s'_i+7\tau) = T^\infty[s'_j\,..\,s'_j+7\tau)$ and $\overline{T^\infty[s'_i-\tau\,..\,s'_i)} \prec \overline{T^\infty[s'_j-\tau\,..\,s'_j)}$.

Based on $(s'_i)_{i\in[1\,..\,|\mathsf{S}|]}$, we define three length-$|\mathsf{S}|$ sequences. For $i \in [1\,..\,|\mathsf{S}|]$, we set

$$\begin{aligned}\widetilde{W}[i] &= T^\infty[s'_i-\tau\,..\,s'_i+7\tau),\\ W[i] &= \overline{T^\infty[s'_i-\tau\,..\,s'_i)},\\ W'[i] &= T^\infty[s'_i\,..\,s'_i+7\tau).\end{aligned}$$

Recall that we can compactly represent $\widetilde{W}$ in $\mathcal{O}(1+|\mathrm{RL}(\widetilde{W})|)$ space using $\mathrm{RL}((\mathrm{lpos}(\widetilde{W}[j]))_{j\in[1\,..\,|\mathsf{S}|]})$. The sequences W and W' can be represented analogously, except that we use $\mathrm{RL}((\overline{\mathrm{lpos}}(W[j]))_{j\in[1\,..\,|\mathsf{S}|]})$ for W.

Lemma V.15. *The sequences $\widetilde{W}$, W, and W' defined above satisfy $|\mathrm{RL}(W)|$, $|\mathrm{RL}(W')| \le |\mathrm{RL}(\widetilde{W})| \le |\mathrm{comp}_7(\mathsf{S})|$.*

Proof: For the first inequality, note that $\widetilde{W}[i] = \widetilde{W}[i+1]$ implies $W[i] = W[i+1]$ and $W'[i] = W'[i+1]$.

Let $\mathrm{RL}(\widetilde{W}) = ((R_1,\lambda_1),\ldots,(R_h,\lambda_h))$. Observe that $i \ne j$ implies $R_i \ne R_j$. For $i \in [1\,..\,h]$, let $T^\infty[p-\tau\,..\,p+7\tau)$ be the occurrence of R_i in T^∞ that minimizes $p \in [1\,..\,n]$. Then, there exists $j \in [1\,..\,z]$ such that $e_j - 8\tau < p-\tau \le e_j$. By the consistency of S, we conclude that $p \in \mathsf{S} \cap (e_j-7\tau\,..\,e_j+\tau] \subseteq \mathrm{comp}_7(\mathsf{S})$. The claim follows, since this map is injective. ■

Importantly, the compact representations of $\widetilde{W}$, W, and W' can be computed efficiently.

Lemma V.16 (♠). *Given $\mathrm{comp}_7(\mathsf{S})$ and the LZ77 parsing of T, the compact representations of $\widetilde{W}$, W, and W' can be constructed in $\mathcal{O}(z\log^4 n + |\mathrm{comp}_7(\mathsf{S})|\log^3 n)$ time.*

Next, we recall the notion of *distinguishing prefixes*, originally introduced in [33], that allows mapping each suffix $T^\infty[i\,..)$ to the corresponding node of the wavelet tree of W.

Definition V.17 (Distinguishing prefix)**.** *For any position $i \in [1\,..\,\max(\mathsf{S}\cup\{0\})]$, let $i_{\mathrm{succ}} = \min\{j \in \mathsf{S} : j \ge i\}$. The* distinguishing prefix *of $T[i\,..\,n]$ is $D_i = T[i\,..\,i_{\mathrm{succ}}+2\tau)$.*

Let $\mathcal{D} = \{D_j : j \in [1\,..\,\max(\mathsf{S}\cup\{0\})]\}$. Note that if Y starts with $D \in \mathcal{D}$, then, for every occurrence $T^\infty[i\,..\,i+|Y|) = Y$ with $i \in [1\,..\,n]$, the distinguishing prefix D_i is defined and satisfies $D_i = D$.[2] Thus, for any such Y, we define $D_Y = D$. We denote $D'_Y = D_Y[1\,..\,|D_Y|-2\tau]$.

We now present the key lemma used in our algorithm. Assume that we have constructed a wavelet tree of W.

Lemma V.18. *Let Y be a string starting with an element of $\mathcal{D}$. Denote $Y = XX'$, where $X = D'_Y$, and assume that $|X| < \tau$ and $|X'| \le 7\tau$. Let $[y\,..\,y']$ be the range of all indices i such that $T^\infty[\mathrm{SA}[i]\,..]$ starts with Y for $i \in [y\,..\,y']$.*

Let $W'[f\,..\,f']$ be the range containing all elements of W' prefixed with the string X', and let $[b\,..\,b'] = \{i \in [1\,..\,|I_{\overline{X}}|] : I_{\overline{X}}[i] \in [f\,..\,f']\}$. Then

1) *$B_{\overline{X}}[b\,..\,b']$ and $\mathrm{BWT}[y\,..\,y']$ are equal as multisets.*
2) *$|\mathrm{RL}(B_{\overline{X}}[b\,..\,b'])| \le 3|\mathrm{RL}(\mathrm{BWT}[y\,..\,y'])|$.*

Proof: 1. Due to $T[n] = \$$, by the consistency of S, there is a one-to-one correspondence between the occurrences of Y in T^∞ starting in $[1\,..\,n]$, and positions $s \in \mathsf{S}$ satisfying (a) $T^\infty[s\,..\,s+|X'|) = X'$, and (b) $T^\infty[s-|X|\,..\,s) = X$. Let us interpret the process of identifying the subsequence of $(s'_i)_{i\in[1\,..\,|\mathsf{S}|]}$ containing all such s as a two-step search.

First, we note that $s'_i \in \mathsf{S}$ satisfies condition (a) if and only if $i \in [f\,..\,f']$. We refer to the process of identifying the range $[f\,..\,f']$ as the *forward search*. Then, to additionally satisfy (b), we select a subsequence of $W[f\,..\,f']$ containing only strings ending with X (*backward search*). By definition of $[b\,..\,b']$, such subsequence is given by $I_{\overline{X}}[b\,..\,b']$, and moreover, $B_{\overline{X}}[b\,..\,b']$ contains symbols preceding suffix X in all $W[f\,..\,f']$ having X as a suffix. This yields the claim.

2. Let $\widetilde{Y}$ be any substring of T^∞ such that Y is a prefix of $\widetilde{Y}$ and $|\widetilde{Y}| = |X| + 7\tau$. Let $[\widetilde{y}\,..\,\widetilde{y}']$, $[\widetilde{f}\,..\,\widetilde{f}']$, and $[\widetilde{b}\,..\,\widetilde{b}']$ be the ranges (as in the lemma statement) for $\widetilde{Y}$. Since Y is a prefix of $\widetilde{Y}$ and $D_{\widetilde{Y}} = D_Y$, we obtain $[\widetilde{y}\,..\,\widetilde{y}'] \subseteq [y\,..\,y']$, $[\widetilde{f}\,..\,\widetilde{f}'] \subseteq [f\,..\,f']$, and $[\widetilde{b}\,..\,\widetilde{b}'] \subseteq [b\,..\,b']$. Moreover, by definition of $I_{\overline{X}}$, the range $[b\,..\,b']$ is a disjoint union of ranges $[\widetilde{b}\,..\,\widetilde{b}']$ corresponding to all choices of $\widetilde{Y}$.

Since $|\widetilde{Y}| - |X| = 7\tau$ implies $|\mathrm{RL}(W'[\widetilde{f}\,..\,\widetilde{f}'])| = 1$, the symbols in $B_{\overline{X}}[\widetilde{b}\,..\,\widetilde{b}']$ appear in the nondecreasing order. Consequently, $B_{\overline{X}}[b\,..\,b']$ can be obtained by partitioning $\mathrm{BWT}[y\,..\,y']$ into blocks corresponding to all $\widetilde{Y}$, and sorting the symbols in each block. If $\mathrm{BWT}[y\,..\,y']$ initially contains k runs, this adds at most $2(k-1)$ new runs. ■

Let $\mathrm{BWT}_\ell[y\,..\,y'] = c^\delta \in \mathbb{N}^+$ be a run in BWT_ℓ, and let $Y = T^\infty[c\,..\,c+\ell)$. Since Y is left-maximal, $c+\ell \le n$. Thus, by $3\tau \le \ell$ and $\mathsf{R} = \emptyset$, we obtain $[c\,..\,c+\tau) \cap \mathsf{S} \ne \emptyset$. Moreover, if $Y = XX'$ is such that $|X| = \Delta$, where

$$c + \Delta = \min(\mathsf{S} \cap [c\,..\,c+\tau)),$$

then $D'_Y = X$. Since we also have $|X'| \le 2\ell \le 7\tau$ (due to $q \ge 4$), Lemma V.18 holds for Y. Let $[b\,..\,b']$ be the range of $B_{\overline{X}}$ corresponding to Y through Lemma V.18. Then, $|\mathrm{RL}(B_{\overline{X}})| > 1$. Moreover:

- For every $j \in [0\,..\,\delta)$ such that $\mathrm{BWT}_{2\ell}[y+j] \in \Sigma$, the following equality holds: $\mathrm{BWT}_{2\ell}[y+j] = B_{\overline{X}}[b+j]$. To see this, apply Lemma V.18 to all strings $\widetilde{Y} \in \mathcal{Y} := \{T^\infty[\mathrm{SA}[j]\,..\,\mathrm{SA}[j]+2\ell) : j \in [y\,..\,y']\}$ ordered lexicographically. Since $D_{\widetilde{Y}} = D_Y$, the corresponding ranges $[\widetilde{b}\,..\,\widetilde{b}']$ form a left-to-right partition of $[b\,..\,b']$. Thus, if $\widetilde{Y}$ is not left-maximal, its BWT block is $\mathrm{BWT}[\widetilde{y}\,..\,\widetilde{y}'] = B_{\overline{X}}[\widetilde{b}\,..\,\widetilde{b}']$.

[2] Here, we utilize the assumption that $T[n] = \$$. For this reason, if Y contains \$, then $T^\infty[i\,..\,i+|Y|) = Y$ for at most one index $i \in [1\,..\,n]$.

- On the other hand, if $c' = \mathrm{BWT}_{2\ell}[y+j] \in \mathbb{N}$ holds for some $j \in [0\,..\,\delta)$, then the string $B_{\overline{X}}[\widetilde{b}\,..\,\widetilde{b'}]$ corresponding (through Lemma V.18) to $\widetilde{Y} = T^{\infty}[c'\,..\,c'+2\ell) \in \mathcal{Y}$ is not unary, i.e., there exists an index $\hat{b} \in [\widetilde{b}\,..\,\widetilde{b'})$ satisfying $B_{\overline{X}}[\hat{b}] \neq B_{\overline{X}}[\hat{b}+1]$, and $\mathrm{lcp}(W'[I_{\overline{X}}[\hat{b}]], W'[I_{\overline{X}}[\hat{b}+1]]) \geq 2\ell - |X|$. The converse is also true: if $\hat{b} \in [b\,..\,b')$ satisfies the two conditions, then $\mathrm{BWT}_{2\ell}[y+(\hat{b}-b)] \in \mathbb{N}$. Consequently, the set of left-maximal strings in $\mathcal{Y}$ is

$$\{X \cdot W'[I_{\overline{X}}[\hat{b}]][1\,..\,2\ell-|X|] : \hat{b} \in [b\,..\,b'),\ B_{\overline{X}}[\hat{b}] \neq B_{\overline{X}}[\hat{b}+1],\ \text{and}\ \mathrm{lcp}(W'[I_{\overline{X}}[\hat{b}]], W'[I_{\overline{X}}[\hat{b}+1]]) \geq 2\ell - |X|\}.$$

  Moreover, letting $\widetilde{Y} = X \cdot W'[I_{\overline{X}}[\hat{b}]][1\,..\,2\ell-|X|]$ for any $\hat{b}$ satisfying the above conditions, the range $[\widetilde{y}\,..\,\widetilde{y'}] = \{j \in [1\,..\,n] : T^{\infty}[\mathrm{SA}[j]\,..\,\mathrm{SA}[j]+2\ell) = \widetilde{Y}\}$ satisfies $[\widetilde{y}\,..\,\widetilde{y'}] = y - b + [\widetilde{b}\,..\,\widetilde{b'}]$, where $B_{\overline{X}}[\widetilde{b}\,..\,\widetilde{b'}]$ corresponds to $\widetilde{Y}$ through Lemma V.18.

The algorithm processing a run $\mathrm{BWT}_{\ell}[y\,..\,y'] = c^{\delta} \in \mathbb{N}^{+}$ is thus as follows. Letting $Y = T^{\infty}[c\,..\,c+\ell)$ and $X = D'_Y$, we first compute $|X|$ and the pointer to $v_{\overline{X}}$. We then perform a single forward and backward search to find the ranges $[f\,..\,f']$ and $[b\,..\,b']$ for Y. Given these, the computation of $\mathrm{BWT}_{2\ell}[y\,..\,y']$ is achieved by a series of forward and backward searches (at most one per run of $B_{\overline{X}}[b\,..\,b']$).

The length $|X|$ is computed using $\mathrm{comp}_7(\mathsf{S})$. The pointers to $v_{\overline{X}}$ are precomputed since $|\mathrm{RL}(B_{\overline{X}})| > 1$ implies $v_{\overline{X}} \in V(\mathcal{T}_c)$. To implement a forward search, we use LCE queries on T. A backward search is implemented using primary index queries on the wavelet tree of W. We thus obtain:

Proposition V.19 (♠). *Let T be a string of length n, and let $\ell = 2^q$ be such that $q \in [4\,..\,\lceil \log n \rceil)$. If $\mathsf{R} = \emptyset$, then, given $\mathrm{RL}(\mathrm{BWT}_{\ell})$ and the LZ77 parsing of T, the sequence $\mathrm{RL}(\mathrm{BWT}_{2\ell})$ can be constructed in $\mathcal{O}((r+z)\log^5 n)$ time.*

2) The General Case: Periodic fragments are handled similarly as in the BWT construction in [33], resulting in the following algorithm.

Proposition V.20 (♠). *Let T be a string of length n, and let $\ell = 2^q$ be such that $q \in [4\,..\,\lceil \log n \rceil)$. Then, given $\mathrm{RL}(\mathrm{BWT}_{\ell})$ and the LZ77 parsing of T, the sequence $\mathrm{RL}(\mathrm{BWT}_{2\ell})$ can be constructed in $\mathcal{O}((r+z)\log^5 n)$ time.*

By Proposition V.14 we can compute BWT_{ℓ} for $\ell = 2^q$ and $q < 4$ in $\mathcal{O}(z \log^4 n)$ time. For $q \geq 4$, we use Proposition V.20. Thus, by the upper bound $r = \mathcal{O}(z \log^2 n)$ from Theorem III.2, we obtain the main result of this section.

Theorem V.21. *There exists a Las-Vegas randomized algorithm that, given the LZ77 parsing of a text T of length n, computes its run-length compressed Burrows–Wheeler transform in $\mathcal{O}((r+z)\log^6 n) = \mathcal{O}(z \log^8 n)$ time.*

Acknowledgments: The authors would like to thank Barna Saha for helpful discussions. D. Kempa is supported by NSF grants no. 1652303 and 1934846, and an Alfred P. Sloan Fellowship grant. T. Kociumaka is supported by ISF grants no. 1278/16 and 1926/19, by a BSF grant no. 2018364, and by an ERC grant MPM under the EU's Horizon 2020 Research and Innovation Programme (agreement no. 683064).

REFERENCES

[1] "The 100000 Genomes Project," https://www.genomicsengland.co.uk/about-genomics-england/the-100000-genomes-project/, accessed: 2019-04-13.

[2] D. Adjeroh, T. Bell, and A. Mukherjee, *The Burrows-Wheeler Transform: Data Compression, Suffix Arrays, and Pattern Matching*. Boston, MA, USA: Springer, 2008.

[3] M. Alzamel, M. Crochemore, C. S. Iliopoulos, T. Kociumaka, J. Radoszewski, W. Rytter, J. Straszynski, T. Waleń, and W. Zuba, "Quasi-linear-time algorithm for longest common circular factor," in *CPM*, 2019, pp. 25:1–25:14.

[4] A. Amir, C. S. Iliopoulos, and J. Radoszewski, "Two strings at Hamming distance 1 cannot be both quasiperiodic," *Inf. Process. Lett.*, vol. 128, pp. 54–57, 2017.

[5] D. Arroyuelo, G. Navarro, and K. Sadakane, "Stronger Lempel-Ziv based compressed text indexing," *Algorithmica*, vol. 62, no. 1-2, pp. 54–101, 2012.

[6] H. Bannai, T. Gagie, and T. I, "Refining the *r*-index," *Theor. Comput. Sci.*, vol. 812, pp. 96–108, 2020.

[7] D. Belazzougui, F. Cunial, T. Gagie, N. Prezza, and M. Raffinot, "Composite repetition-aware data structures," in *CPM*, 2015, pp. 26–39.

[8] ——, "Flexible indexing of repetitive collections," in *CiE*, 2017, pp. 162–174.

[9] D. Belazzougui, T. Gagie, P. Gawrychowski, J. Kärkkäinen, A. O. Pereira, S. J. Puglisi, and Y. Tabei, "Queries on LZ-bounded encodings," in *DCC*, 2015, pp. 83–92.

[10] P. Bille, M. B. Ettienne, I. L. Gørtz, and H. W. Vildhøj, "Time-space trade-offs for Lempel-Ziv compressed indexing," *Theor. Comput. Sci.*, vol. 713, pp. 66–77, 2018.

[11] P. Bille, G. M. Landau, R. Raman, K. Sadakane, S. R. Satti, and O. Weimann, "Random access to grammar-compressed strings and trees," *SIAM J. Comput.*, vol. 44, no. 3, pp. 513–539, 2015.

[12] A. Blumer, J. A. Blumer, D. Haussler, R. M. McConnell, and A. Ehrenfeucht, "Complete inverted files for efficient text retrieval and analysis," *J. ACM*, vol. 34, no. 3, pp. 578–595, 1987.

[13] M. Burrows and D. J. Wheeler, "A block-sorting lossless data compression algorithm," Digital Equipment Corporation, Palo Alto, California, Tech. Rep. 124, 1994.

[14] M. Charikar, E. Lehman, D. Liu, R. Panigrahy, M. Prabhakaran, A. Sahai, and A. Shelat, "The smallest grammar problem," *Trans. Inf. Theory*, vol. 51, no. 7, pp. 2554–2576, 2005.

[15] A. R. Christiansen, M. B. Ettienne, T. Kociumaka, G. Navarro, and N. Prezza, "Optimal-time dictionary-compressed indexes," 2019, arXiv 1811.12779.

[16] J. G. Cleary and I. H. Witten, "Data compression using adaptive coding and partial string matching," *IEEE Trans. Commun.*, vol. 32, no. 4, pp. 396–402, 1984.

[17] G. V. Cormack and R. N. Horspool, "Data compression using dynamic Markov modelling," *Comput. J.*, vol. 30, no. 6, pp. 541–550, 1987.

[18] T. M. Cover and J. A. Thomas, *Elements of Information Theory 2nd Edition*. Wiley, 2006.

[19] N. J. Fine and H. S. Wilf, "Uniqueness theorems for periodic functions," *Proc. Am. Math. Soc.*, vol. 16, no. 1, pp. 109–114, 1965.

[20] T. Gagie, "Large alphabets and incompressibility," *Inf. Process. Lett.*, vol. 99, no. 6, pp. 246–251, 2006.

[21] T. Gagie, P. Gawrychowski, J. Kärkkäinen, Y. Nekrich, and S. J. Puglisi, "A faster grammar-based self-index," in *LATA*, 2012, pp. 240–251.

[22] T. Gagie, G. Manzini, G. Navarro, and J. Stoye, "25 Years of the Burrows-Wheeler Transform (Dagstuhl Seminar 19241)," *Dagstuhl Reports*, vol. 9, no. 6, pp. 55–68, 2019.

[23] T. Gagie, G. Navarro, and N. Prezza, "Fully-functional suffix trees and optimal text searching in BWT-runs bounded space," 2018, arXiv 1809.02792.

[24] ——, "On the approximation ratio of Lempel-Ziv parsing," in *LATIN*, 2018, pp. 490–503.

[25] ——, "Fully functional suffix trees and optimal text searching in BWT-runs bounded space," *J. ACM*, vol. 67, no. 1, pp. 1–54, apr 2020.

[26] J. Gailly and M. Adler, "gzip Homepage," https://www.gzip.org/, accessed: 2019-10-19.

[27] J. K. Gallant, "String compression algorithms," Ph.D. dissertation, Princeton University, 1982.

[28] R. Grossi, A. Gupta, and J. S. Vitter, "High-order entropy-compressed text indexes," in *SODA*, 2003, pp. 841–850.

[29] J. Kärkkäinen, D. Kempa, and M. Piątkowski, "Tighter bounds for the sum of irreducible LCP values," *Theor. Comput. Sci.*, vol. 656, pp. 265–278, 2016.

[30] J. Kärkkäinen, D. Kempa, and S. J. Puglisi, "Lazy Lempel-Ziv factorization algorithms," *ACM J. Exp. Algor.*, vol. 21, no. 1, pp. 2.4:1–2.4:19, 2016.

[31] T. Kasai, G. Lee, H. Arimura, S. Arikawa, and K. Park, "Linear-time longest-common-prefix computation in suffix arrays and its applications," in *CPM*, 2001, pp. 181–192.

[32] D. Kempa, "Optimal construction of compressed indexes for highly repetitive texts," in *SODA*, 2019, pp. 1344–1357.

[33] D. Kempa and T. Kociumaka, "String synchronizing sets: Sublinear-time BWT construction and optimal LCE data structure," in *STOC*, 2019, pp. 756–767.

[34] D. Kempa and N. Prezza, "At the roots of dictionary compression: String attractors," in *STOC*, 2018, pp. 827–840.

[35] T. Kida, T. Matsumoto, Y. Shibata, M. Takeda, A. Shinohara, and S. Arikawa, "Collage system: A unifying framework for compressed pattern matching," *Theor. Comput. Sci.*, vol. 298, no. 1, pp. 253–272, 2003.

[36] T. Kociumaka, "Efficient data structures for internal queries in texts," Ph.D. dissertation, University of Warsaw, 2018.

[37] T. Kociumaka, G. Navarro, and N. Prezza, "Towards a definitive measure of repetitiveness," 2019, arXiv 1910.02151.

[38] S. Kreft and G. Navarro, "On compressing and indexing repetitive sequences," *Theor. Comput. Sci.*, vol. 483, pp. 115–133, 2013.

[39] B. Langmead, C. Trapnell, M. Pop, and S. L. Salzberg, "Ultrafast and memory-efficient alignment of short DNA sequences to the human genome," *Genome Biol.*, vol. 10, no. 3, p. R25, 2009.

[40] H. Li and R. Durbin, "Fast and accurate short read alignment with Burrows-Wheeler transform," *Bioinform.*, vol. 25, no. 14, pp. 1754–1760, 2009.

[41] R. Li, C. Yu, Y. Li, T. W. Lam, S. Yiu, K. Kristiansen, and J. Wang, "SOAP2: an improved ultrafast tool for short read alignment," *Bioinform.*, vol. 25, no. 15, pp. 1966–1967, 2009.

[42] M. Mahoney, "Large Text Compression Benchmark," http://mattmahoney.net/dc/text.html, accessed: 2020-09-07.

[43] ——, "Adaptive weighing of context models for lossless data compression," Florida Institute of Technology, Melbourne, Florida, Tech. Rep. CS-2005-16, 2005.

[44] V. Mäkinen, D. Belazzougui, F. Cunial, and A. I. Tomescu, *Genome-scale algorithm design: Biological sequence analysis in the era of high-throughput sequencing*. Cambridge, UK: Cambridge University Press, 2015.

[45] U. Manber and E. W. Myers, "Suffix arrays: A new method for on-line string searches," *SIAM J. Comput.*, vol. 22, no. 5, pp. 935–948, 1993.

[46] S. Mantaci, A. Restivo, G. Romana, G. Rosone, and M. Sciortino, "String attractors and combinatorics on words," in *ICTCS*, 2019, pp. 57–71.

[47] G. Manzini, "An analysis of the Burrows-Wheeler transform," *J. ACM*, vol. 48, no. 3, pp. 407–430, 2001.

[48] G. Navarro, *Compact data structures: A practical approach*. Cambridge, UK: Cambridge University Press, 2016.

[49] ——, "Indexing highly repetitive string collections," 2020, arXiv 2004.02781.

[50] G. Navarro and N. Prezza, "Universal compressed text indexing," *Theor. Comput. Sci.*, vol. 762, pp. 41–50, 2019.

[51] E. Ohlebusch, *Bioinformatics algorithms: Sequence analysis, genome rearrangements, and phylogenetic reconstruction*. Ulm, Germany: Oldenbusch Verlag, 2013.

[52] E. Ohlebusch, T. Beller, and M. I. Abouelhoda, "Computing the Burrows-Wheeler transform of a string and its reverse in parallel," *J. Discrete Alg.*, vol. 25, pp. 21–33, 2014.

[53] T. Ohno, K. Sakai, Y. Takabatake, T. I, and H. Sakamoto, "A faster implementation of online RLBWT and its application to LZ77 parsing," *J. Discrete Alg.*, vol. 52-53, pp. 18–28, 2018.

[54] I. Pavlov, "7-zip Homepage," https://www.7-zip.org/, accessed: 2019-10-19.

[55] A. Policriti and N. Prezza, "From LZ77 to the run-length encoded Burrows-Wheeler transform, and back," in *CPM*, 2017, pp. 17:1–17:10.

[56] ——, "LZ77 computation based on the run-length encoded BWT," *Algorithmica*, vol. 80, no. 7, pp. 1986–2011, 2018.

[57] N. Prezza, "Can Lempel-Ziv and Burrows-Wheeler compression be asymptotically compared?" https://nms.kcl.ac.uk/iwoca/problems/Prezza2016_updated2019.pdf, 2016, IWOCA 2016 Open Problems.

[58] ——, "Optimal rank and select queries on dictionary-compressed text," in *CPM*, 2019, pp. 4:1–4:12.

[59] M. Przeworski, R. R. Hudson, and A. D. Rienzo, "Adjusting the focus on human variation," *Trends Genet.*, vol. 16, no. 7, pp. 296–302, 2000.

[60] J. Seward, "bzip2 Homepage," https://www.sourceware.org/bzip2/, accessed: 2019-10-19.

[61] S. Sinha and O. Weinstein, "Local decodability of the Burrows-Wheeler transform," in *STOC*, 2019, pp. 744–755.

[62] J. Sirén, N. Välimäki, V. Mäkinen, and G. Navarro, "Run-length compressed indexes are superior for highly repetitive sequence collections," in *SPIRE*, 2008, pp. 164–175.

[63] J. A. Storer and T. G. Szymanski, "The macro model for data compression," in *STOC*, 1978, pp. 30–39.

[64] A. Thue, "Über unendliche zeichenreihen," *Norsk. Vid. Selsk. Skr.I, Mat.-Nat.Kl. Nr.7*, pp. 1–22, 1906.

[65] P. Weiner, "Linear pattern matching algorithms," in *SWAT/FOCS*, 1973, pp. 1–11.

[66] J. Ziv and A. Lempel, "A universal algorithm for sequential data compression," *Trans. Inf. Theory*, vol. 23, no. 3, pp. 337–343, 1977.

[67] ——, "Compression of individual sequences via variable-rate coding," *Trans. Inf. Theory*, vol. 24, no. 5, pp. 530–536, 1978.