Nearly Tight Bounds on the Encoding Length of the Burrows-Wheeler Transform

Ankur Gupta* Roberto Grossi† Jeffrey Scott Vitter‡

Abstract

In this paper, we present a nearly tight analysis of the encoding length of the Burrows-Wheeler Transform (BWT) that is motivated by the text indexing setting. For a text T of n symbols drawn from an alphabet Σ, our encoding scheme achieves bounds in terms of the hth-order empirical entropy H_h of the text, and takes linear time for encoding and decoding. We also describe a lower bound on the encoding length of the BWT that constructs an infinite (non-trivial) class of texts that are among the hardest to compress using the BWT. We then show that our upper bound encoding length is *nearly tight* with this lower bound for the class of texts we described.

In designing our BWT encoding and its lower bound, we also address the *t-subset problem*; here, the goal is to store a subset of t items drawn from a universe $[1..n]$ using just $\lg \binom{n}{t} + O(1)$ bits of space. A number of solutions to this basic problem are known, however encoding or decoding usually requires either $O(t)$ operations on large integers [Knu05, Rus05] or $O(n)$ operations. We provide a novel approach to reduce the encoding/decoding time to just $O(t)$ operations on *small* integers (of size $O(\lg n)$ bits), without increasing the space required.

1 Introduction

The Burrows-Wheeler transform [BW94] (BWT) has had a profound impact on a myriad of algorithmic fields in Computer Science. The BWT preprocesses an input text T by a reversible transformation—the result is then easily compressible by other simpler encoding methods. The BWT highlights (among other things) that a series of encoders may be more effective than a one-pass compression algorithm. The BWT has shown remarkable evidence to that effect, especially in practice: it is at the heart of the `bzip2` algorithm, which has become a mainstream data compression tool.

The BWT has also revolutionized the field of text indexing [NM06]. Using the BWT, one can build a "compressed text index", a class of data structures that support powerful substring searches and occupy roughly the same space as that required by the best compressors. These results [GV05, FM05] have dispelled the notion that an efficient full-text index must use space superlinear in the text length. Clearly, the BWT is a powerful compression tool that manages to organize data in a searchable way—but where is the magic?

The BWT has been analyzed extensively since its original introduction in 1994, especially in the information-theory community [WMF94, EVKV02]. These results apply to a wide range of statistical models for generating a text T, including high-order Markov sources, tree sources, and finite-state machine (FSM) sources. To the best of our knowledge, the best known encoding of the BWT appears in [BB04], achieving a nearly optimal BWT encoding length that encodes in $O(n)$ time, but decoding takes longer.

In a text indexing setting, recent theoretical results [Man01, FGMS05, KLV06] have shed light on the success of the BWT and present some limits on its compressibility. For a text T of n symbols drawn from an alphabet Σ ($|\Sigma| = \sigma$), these results achieve bounds in terms of the modified hth-order empirical entropy H_h^* of the text. However, little attention is paid to reducing the cost for encoding the hth order count statistics, since $\sigma^h = o(n/\log\sigma)$ is a reasonable assumption. Even in such cases, the costs for encoding the *empirical statistical model* for T may dwarf the leading entropy term. (More precise comparisons follow in Section 3.) In this paper, we deeply consider the problem of storing the empirical statistical model, without any assumption on the size of σ and h. This investigation may lead to a clear

*Department of Computer Science and Software Engineering, Butler University, Indianapolis, IN 46208 (`agupta@butler.edu`). Some work was originally done at Purdue University. Support was provided in part by the Army Research Office (ARO) through grant DAAD20–03–1–0321 and by the National Science Foundation through research grant ISS–0415097.

†Dipartimento di Informatica, Università di Pisa, Largo Bruno Pontecorvo 3, 56127 Pisa (`grossi@di.unipi.it`). Support was provided in part by Italian Ministry of Education, University and Research (MIUR).

‡Department of Computer Science, Purdue University, West Lafayette, IN 47097–2066 (`agupta@cs.purdue.edu`, `jsv@purdue.edu`). Support was provided in part by the Army Research Office (ARO) through grant DAAD20–03–1–0321 and by the National Science Foundation through research grant ISS–0415097.

understanding of the best way to encode the BWT.

We present a nearly tight analysis on the encoding length of the BWT. Our encoding scheme can be encoded or decoded in linear time, is strongly motivated by current text indexing techniques, and we present our results to validate the current direction of text indexes. In particular, we present an encoding scheme for the BWT that can be encoded or decoded in $O(n)$ time and requires $nH_h + \min\{g'_h \lg(n/g'_h + 1), H^*_h n + \lg n + g''_h\}$ bits of space, where $g'_h = O(|\Sigma|^{h+1})$ and $g''_h = O(|\Sigma|^{h+1} \lg |\Sigma|^{h+1})$ do not depend on the text length n, while $H^*_h \geq H_h$ is the modified hth-order empirical entropy of T. Notice that we have bounded the encoding of the empirical model cost (everything but nH_h in the above bound) by the *entropy of the underlying text* T. In other words, encoding the empirical statistics does not seem to be a trivial cost!

To show that our encoding is nearly tight with the lower bound encoding length of the BWT, we describe a class of texts, called *δ-resilient texts*, for which our encoding achieves nearly tight bounds. One could view the class of δ-resilient texts as one of the hardest to compress using the BWT. A consequence of this result is the observation that one can, without fear of incurring significant overhead, separate the encoding of the underlying data from the representation of the empirical statistical model. In this paper, we explain how this separation naturally occurs in the text indexing setting.

We also present a novel subset encoding scheme that addresses some of the weaknesses of well-known results from combinatorial enumeration [Knu05, Rus05]. Our approach applies a quasi-arithmetic coder to a set of empirical probability estimates generated by a random sampling technique [Vit84] to avoid operations on large integers. In particular, our subset encoding for a set of t items out of a universe of size n can be decoded using $O(t)$ operations on small integers (of size $O(\lg n)$ bits).

2 Preliminaries

We formulate our analysis of the space complexity in terms of the high-order empirical entropy of a text T of n symbols drawn from alphabet $\Sigma = \{1, 2, \ldots, \sigma\}$. In this section, we discuss entropy from both an empirical probability model and a finite set model. We also review the BWT and its connection to suffix arrays.

2.1 Empirical Probabilistic Entropy We provide terminology for the analysis and explore empirical probability models. For each symbol $y \in \Sigma$, let n^y be the number of its occurrences in text T. With symbol y, we associate its empirical probability, $\text{Prob}[y] = n^y/n$, of occurring in T. (By definition, $n = \sum_{y\in\Sigma} n^y$, so the empirical probability is well defined.) Following Shannon's definition of entropy [Sha48], the *0th-order empirical entropy* is $H_0 = H_0(T) = \sum_{y\in\Sigma} -\text{Prob}[y] \times \lg \text{Prob}[y]$. Since $nH_0 \leq n \lg \sigma$, the previous expression simply states that an efficient variable-length coding of text T would encode each symbol y based upon its frequency in T rather than simply $\lg \sigma$ bits. The number of bits assigned for encoding an occurrence of y would be $-\lg \text{Prob}[y] = \lg(n/n^y)$.

We can generalize the definition to higher-order empirical entropy, so as to capture the dependence of symbols upon their context, made up of the h previous symbols in the text.[1] For a given h, we consider all possible h-symbol sequences x that appear in the text. (They are a subset of Σ^h, the set of all possible h-symbol sequences over the alphabet Σ. The last $h-1$ incomplete sequences are easily stored within the bounds we give in this paper; we defer these technical details until the full paper.) We denote the number of occurrences in the text of a particular context x by n^x, with $n = \sum_{x\in\Sigma^h} n^x$ as before, and we let $n^{x,y}$ denote the number of occurrences in the text of the concatenated sequence yx (meaning that y precedes x).

Manzini [Man01] gives a definition of the *hth-order empirical entropy* H_h in terms of H_0. For any given context x, let w_x be the concatenation of the symbols y that appear in the text immediately before context x. We denote its length by $|w_x|$ and its 0th-order empirical entropy by $H_0(w_x)$, thus defining H_h as

$$H_h = \frac{1}{n} \sum_{x\in\Sigma^h} |w_x| H_0(w_x). \tag{2.1}$$

An important observation to note is that $H_{h+1} \leq H_h \leq \lg \sigma$ for any integer $h \geq 0$. Hence, the previous expression states that a better variable-length coding of text T would encode each symbol y based upon the joint and conditional empirical frequency for any context x of y.

One potential difficulty with the definition of H_h is that the inner terms of the summation could equal 0 (or an arbitrarily small constant), which can be misleading when considering the encoding length of a text T. (One relatively trivial case is when the text contains n equal symbols, as no symbol needs to be "predicted".) Manzini introduced *modified high-order empirical entropy* H^*_h to address this point and capture the constraint that the encoding of the text must contain at least $\lg n$ bits for coding its length n. Using a modified $H^*_0 = H^*_0(T) = \max\{H_0, (1 +$

[1] The standard definition of conditional probability for text documents considers the symbol y immediately *after* the sequence x. It makes no difference, since we could use this definition on the reversed text as discussed in [FGMS05].

$\lfloor \lg n \rfloor)/n\}$ to make the change, Manzini defines the H_h^* as

$$H_h^* = H_h^*(T) = \frac{1}{n} \min_{P_h} \sum_{x \in P_h} |w_x| H_0^*(w_x). \tag{2.2}$$

Here, we let P_h be a *prefix cover*, namely, a set of substrings having length at most h such that every string from Σ^h has a unique prefix in P_h.

Immediate consequences of this encoding-motivated entropy measure are that $H_h^* \geq H_h$ and $nH_h^* \geq \lg n$, but nH_h can be a small constant. Let the *optimal prefix cover* P_h^* be the prefix cover that minimizes H_h^* in (2.2). Thus, (2.2) can be equivalently stated as $H_h^* = \frac{1}{n} \sum_{x \in P_h^*} |w_x| H_0^*(w_x)$.[2] The empirical probabilities that are employed in the definition of the high-order empirical entropy can be obtained from the number of occurrences $n^{x,y}$, where $\sum_{x \in P_h^*, y \in \Sigma} n^{x,y} = n$. Indeed, $n^y = \sum_{x \in P_h^*} n^{x,y}$ and $n^x = \sum_{y \in \Sigma} n^{x,y}$. This discussion motivates the following definition, which will guide us through our high-order entropy analysis.

Definition 1. The *empirical statistical model* for a text T is composed of two parts stored using $M(T, \Sigma, h)$ bits:

i. The partition of Σ^h induced by the contexts of the prefix cover P_h^*.
ii. The sequence of non-negative integers, $n^{x,1}, n^{x,2}, \ldots, n^{x,\sigma}$, where $x \in P_n^*$. (Recall that $n^{x,y}$ is the number of occurrences of yx as a substring of T.)

We denote the number of bits used to store the information in parts (i)–(ii) by $M(T, \Sigma, h)$, as n increases.

2.2 Finite Set Entropy We provide a new definition of high-order empirical entropy H_h', based on the finite set model rather than on conditional probabilities, to avoid dealing with empirical probabilities explicitly. We show that our new definition is $H_h - O(|P_h^*| \lg n) \leq H_h' \leq H_h \leq H_h^*$, so that we can provide bounds in terms of H_h'.

For ease of exposition, we "number" the lexicographically ordered contexts x as $1 \leq x \leq \sigma^h$. Let the *multinomial* coefficient $\binom{n}{m_1, m_2, \ldots, m_p} = \frac{n!}{m_1!\, m_2! \cdots m_p!}$ represent the number of partitions of n items into p subsets of size $m_1, m_2, \ldots, m_p$. In this paper, we define $0! = 1$. (Note that $n = m_1 + m_2 + \cdots + m_p$.) When $m_1 = t$ and $m_2 = n - t$, we get precisely the binomial coefficient $\binom{n}{t}$. We define

$$H_0' = H_0'(T) = \frac{1}{n} \lg \binom{n}{n^1, n^2, \ldots, n^\sigma}, \tag{2.3}$$

which counts the number of possible partitions of n items into σ unique buckets, i.e. the alphabet size. We use the optimal prefix cover P_h^* in (2.2) to define *alternative high-order empirical entropy*[3]

$$H_h' = H_h'(T) = \frac{1}{n} \sum_{x \in P_h^*} \lg \binom{n^x}{n^{x,1}, n^{x,2}, \ldots, n^{x,\sigma}}. \tag{2.4}$$

Theorem 2.1. *For any text T and context length $h \geq 0$, we have $H_h' \leq H_h$.*

Proof. It suffices to show that $nH_0' \leq nH_0$ for all alphabets Σ, since then $\lg \binom{n^x}{n^{x,1}, n^{x,2}, \ldots, n^{x,\sigma}} \leq |w_x| H_0(w_x)$.

The bound $nH_0' \leq nH_0$ trivially holds when $\sigma = 1$. We first prove this bound for an alphabet Σ of $\sigma = 2$ symbols. Let t and $n - t$ denote the number of occurrences of the two symbols in T. We want to show that $nH_0' = \lg \binom{n}{t} \leq nH_0 = t \lg(n/t) + (n-t) \lg(n/(n-t))$ by (2.3). The claim is true by inspection when $n \leq 4$ or $t = 0, 1, n-1$. Let $n > 4$ and $2 \leq t \leq n-2$. We apply Stirling's double inequality [Fel68] to obtain

$$\frac{n^n \sqrt{2\pi n}}{e^{n - \frac{1}{12n+1}}} < n! < \frac{n^n \sqrt{2\pi n}}{e^{n - \frac{1}{12n}}}. \tag{2.5}$$

Taking logarithms and applying the inequality to $\lg \binom{n}{t} = \lg(n!) - \lg(t!) - \lg((n-t)!)$, we have

$$nH_0' = \lg \binom{n}{t} < nH_0 - \frac{1}{2} \lg \frac{t(n-t)}{n}$$
$$- \lg e \left[\frac{1}{12t+1} + \frac{1}{12(n-t)+1} - \frac{1}{12n} \right] - \lg \sqrt{2\pi}.$$

Since $t(n-t) \geq n$ and $1/(12t+1) + 1/(12(n-t)+1) \geq 1/(12n)$ by our assumptions on n and t, it follows that $nH_0' \leq nH_0$, proving the result when $\sigma = 2$.

Next, we show the claimed bound for the general alphabet ($\sigma \geq 2$ and $h = 0$) and by using induction on the alphabet size (with the base case $\sigma = 2$ as detailed before). We write

$$\lg \binom{n}{n^1, n^2, \ldots, n^\sigma} = \lg \left[\binom{n - n^\sigma}{n^1, n^2, \ldots, n^{\sigma-1}} \times \binom{n}{n^\sigma} \right]. \tag{2.6}$$

We use induction for the right-hand side of (2.6) to get

$$\lg \binom{n - n^\sigma}{n^1, n^2, \ldots, n^{\sigma-1}} \leq \sum_{y=1}^{\sigma-1} n^y \lg \frac{n - n^\sigma}{n^y}, \tag{2.7}$$

$$\lg \binom{n}{n^\sigma} \leq n^\sigma \lg \frac{n}{n^\sigma} + (n - n^\sigma) \lg \frac{n}{n - n^\sigma}. \tag{2.8}$$

[2] A minor technical note: h now refers to the length of the longest substring in P_h^*, since no larger value of h can yield a more succinct entropy measure.

[3] Actually, it can be defined for any prefix cover P_h, including $P_h = \Sigma^h$.

Original	*Sorted*			*Mappings*		
Q	F		L	i	$LF(i)$	$\Phi(i)$
mississippi#	i	ppi#missis	s	1	8	7
#mississippi	i	ssippi#mis	s	2	9	10
i#mississipp	i	ssissippi#	m	3	5	11
pi#mississip	i	#mississip	p	4	6	12
ppi#mississi	m	ississippi	#	5	12	3
ippi#mississ	p	i#mississi	p	6	7	4
sippi#missis	p	pi#mississ	i	7	1	6
ssippi#missi	s	ippi#missi	s	8	10	1
issippi#miss	s	issippi#mi	s	9	11	2
sissippi#mis	s	sippi#miss	i	10	2	8
ssissippi#mi	s	sissippi#m	i	11	3	9
ississippi#m	#	mississipp	i	12	4	5

Table 1: Matrix Q for the BWT containing the cyclic shifts of text $T =$ `mississippi#` (column '*Original*'). Sorting of the rows of Q, in which the first (F) and last (L) symbols in each row are separated (column '*Sorted*'). Functions LF and Φ for each row of the sorted Q (column '*Mappings*').

Summing (2.7) and (2.8), we obtain $\sum_{y=1}^{\sigma} n^y \lg \frac{n}{n^y} = nH_0$, thus proving the claim for any alphabet size σ.

COROLLARY 2.1. *For any given text T and context length $h \geq 0$, we have $H_h - O((1/n)|P_h^*| \lg n) \leq H'_h$.*

The above discussion justifies our use of H'_h in later analysis, but we continue to state bounds in terms of H_h since it represents more standard notation. The key point is that we can derive equations in terms of multinomial coefficients without worrying about the empirical probability of symbols appearing in text T.

2.3 The BWT and Suffix Arrays We now give a short description of the BWT in order to explain its salient features. Consider the text $T =$ `mississippi#` in the example shown in Table 1, where `i` $<$ `m` $<$ `p` $<$ `s` $<$ `#` and `#` is an end-of-text symbol. The BWT forms a conceptual matrix Q whose rows are the cyclic (forward) shifts of the text in sorted order and stores the last column $L =$ `ssmp#pissiii` written as a contiguous string. Note that L is an invertible permutation of the symbols in T. In particular, $LF(i) = j$ in Table 1 indicates for any symbol $L[i]$, the corresponding position j in F where $L[i]$ appears. For instance, $LF(8) = 10$ since $L[8] =$ `s` occurs in position 10 of F (as the third `s` among the four appearing consecutively in F).

Using L and LF, we can recreate the text T in reverse order by starting at the last position n (corresponding to `#mississippi`), writing its value from F, and following the LF function to the next value of F. Continuing the example from before, we follow the pointers from $LF(n)$: $LF(12) = 4$, $F[4] =$ `i`; $LF(4) = 6$, $F[6] =$ `p`; $LF(6) = 7$, $F[7] =$ `p`; and so on. In other words, the LF function gives the position in F of the preceding symbol from the original text T. Thus one could store L and recreate T, since we can obtain F by sorting L and the LF function can be derived by inspection. Note that L is compressible using 0th-order compressors, boosting them to attain high-order entropy [FGMS05]. Now, we connect the BWT with L.

We introduce the neighbor function Φ, which is used to represent the compressed suffix array (CSA) in [GV05]. The Φ function can be thought of as the FL mapping, and can represent the BWT much like the LF mapping. The neighbor function Φ is the inverse of the LF mapping, in other words, $\Phi = LF^{-1}$.

We partition the Φ (LF) mapping into Σ-*lists*. Given a symbol $y \in \Sigma$, the list y is the set of positions in L such that for any position p in list y, $L[p] = y$. The fundamental property of these Σ lists is that each list is an *increasing* series of positions. The concatenation of the lists y for $y = 1, 2, \ldots, \sigma$ yields the Φ function (as shown in Table 1). Thus, the value of $\Phi(i)$ is just the ith nonempty entry in the concatenation of the lists, and belongs to some list y.

3 A Nearly Space-Optimal Burrows-Wheeler Transform

We now show our major upper bound result; we describe a nearly optimal analysis of the compressibility of the Burrows-Wheeler transform with respect to high-order empirical entropy, exploiting the properties of the BWT illustrated in Section 2.3.

Let P_h^* be the optimal prefix cover as defined in Section 2, and let $n^{x,y}$ be the corresponding values in equation (2.4), where $x \in P_h^*$ and $y \in \Sigma$. (See also Definition 1.) We denote by $|P_h^*| \leq \sigma^h$ the number of contexts in P_h^*. We describe our main upper bound result in the following theorem.

THEOREM 3.1. (NEARLY SPACE-OPTIMAL BWT) *The Burrows-Wheeler transform for a text T of n symbols drawn from an alphabet Σ can be compressed using $nH_h + M(T, \Sigma, h)$ bits for the best choice of context length h and prefix cover P_h^*, where the number of bits required for encoding the empirical statistical model for P_h^* is $M(T, \Sigma, h) \leq \min\{g'_h \lg(1 + n/g'_h), H_h^* n + \lg n + g''_h\}$, where $g'_h = O(\sigma^{h+1})$ and $g''_h = O(\sigma^{h+1} \lg \sigma^{h+1})$ do not depend on the text length n.*

Using Theorem 3.1, we can compare our analysis with the best bounds from previous work. When compared to the additive term of $O(n \lg \lg n / \lg n)$ in the analysis of the Lempel-Ziv method in [KM99], we obtain an $O(\log n)$ additive term for $\sigma = O(1)$ and $h = O(1)$, giving strong evidence why the

BWT is better than the Lempel-Ziv method. Since $M(T, \Sigma, h) \leq g'_h \lg(n/g'_h + 1)$, our bound becomes $nH_h + O(\lg n)$ when $h = O(1)$ and $\sigma = O(1)$, thus exponentially reducing the additive term of n of the H_h-based analysis in [FGMS05].

In this case, our bound closes the gap in the analysis of the BWT, since it matches the lower bound of $nH_h + \Omega(\lg \lg n)$ up to lower-order terms. The latter comes from the lower bound of $nH_0^* + \Omega(\lg \lg n)$ bits, holding for a large family of compressors (not necessarily related to BWT), as shown in [FGMS05]; the only (reasonable) requirement is that any such compressor must produce a codeword for the text length n when it is fed with an input text consisting of the same symbol repeated n times. Since $H_h \leq H_0^*$, we easily derive the lower bound of $nH_h + \Omega(\lg \lg n)$ bits, but a lower bound of $nH_h + \Omega(\lg n)$ probably exists since $nH_0^* \geq \lg n$ while nH_h can be zero.

For the modified hth-order empirical entropy, we show that Theorem 3.1 can be upper bounded by $n(H_h + H_h^*) + \lg n + g''_h$ bits. Since $H_h \leq H_h^*$, our bound is strictly smaller than $2.5nH_h^* + \lg n + g_h$ bits in [FGMS05], apart from the lower-order terms. Actually, our bound is definitively smaller in some cases. For example, while a bound of the form $nH_h^* + \lg n + g_h$ bits is not always possible [Man01], there are an infinite number of texts for which $nH_h = 0$ while $nH_h^* \neq 0$. In these cases, our bound is $nH_h^* + \lg n + g''_h$ bits.

We devote the rest of Sections 3–5 to the proof of Theorem 3.1. We describe our analysis for an *arbitrary* prefix cover P_h, so it also holds for the optimal prefix cover P_h^* as in equation (2.4). We make use of the basic idea from [GGV03] to partition each list y further into sublists $\langle x, y\rangle$ by contexts $x \in P_h$. Intuitively, sublist $\langle x, y\rangle$ stores the positions such that the symbols that follow $F[y]$ are x. For context length $h = 1$, if we continue the example in Table 1, we break the Σ lists by context (in lexicographical order `i`, `m`, `p`, `s`, and `#`, and numbered from 1 up to $|P_h|$). The list for $y =$ `i` is $\langle 7, 10, 11, 12\rangle$, and is broken into sublist $\langle 7\rangle$ for context $x =$ `p`, sublist $\langle 10, 11\rangle$ for context $x =$ `s`, and sublist $\langle 12\rangle$ for $x =$ `#`.

For any context $x \in P_h$, if we encode its sublists using nearly $\lg \binom{n^x}{n^{x,1}, n^{x,2}, \ldots, n^{x,\sigma}}$ bits, we automatically achieve the hth-order empirical entropy as seen in equation (2.4). For example, context $x =$ `i` should be represented with nearly $\lg \binom{4}{1,1,2}$ bits, since two sublists contain one entry each and one sublist contains two entries. The empirical statistical model should record the partition induced by P_h, record which sublists are empty, and encode the lengths of all nonempty sublists using $M(T, \Sigma, h)$ bits.

Encoding: We run the boosting algorithm from [FGMS05] on the BWT to find the optimal value of context order h and the optimal prefix cover P_h^* using the cost of $nH'_h + M(T, \Sigma, h)$ according to equation (2.4). Once we know h and set $P_h = P_h^*$, we can cleanly separate the contexts and encode the Φ function. Thus, we follow the two steps below:

1. We encode the empirical statistical model given in Definition 1.
2. For each context $x \in P_h$, we separately encode the sublists $\langle x, y\rangle$ for $y \in \Sigma$ to capture high-order entropy. Each of these sublists is a subset of the integers in the range $[1, n^x]$. These sublists form a *partition* of the integers in the interval $[1, n^x]$.

The storage for step 1 is $M(T, \Sigma, h)$, the number of bits required for encoding the model. (See Definition 1.) The storage required for step 2 should use nearly $\lg \binom{n^x}{n^{x,1}, n^{x,2}, \ldots, n^{x,\sigma}}$ bits per context x, and should not exceed a total of nH_h bits plus lower-order terms, once we determine P_h^*, as stated in Theorem 3.1.

Decoding: We retrieve the empirical statistical model encoded in step 1 above, which tells us which sublists are nonempty and their lengths. (Note that the values of n and n^x can be obtained from these lengths.) Next, we retrieve the sublists encoded in step 2 since we know their lengths. At this point, we have recovered the Φ function and we can decode the BWT.

4 Encoding Sublists in High-Order Entropy

In this section, we discuss how to encode the sublists in step 2. This is the first part of the proof of Theorem 3.1.

One simple method for encoding the sublists would be to simply encode each sublist $\langle x, y\rangle$ as a subset of $t = n^{x,y}$ items out of a universe of $n' = n^x$ items. We can use *t-subset encoding*, requiring the information-theoretic minimum of $\lceil \lg \binom{n'}{t} \rceil$ bits, with $O(t)$ operations on large integers, according to [Knu05, Rus05]. All the t-subsets are enumerated in some canonical order (e.g. lexicographic order) and the subset occupying rank r in this order is encoded by the value of r itself, which requires $\lceil \lg \binom{n'}{t} \rceil$ bits. In this way, the t-subset encoding can also be seen as the compressed representation of an *implicit bitvector* of length n': If the subset contains $1 \leq s_1 < s_2 < \cdots < s_t \leq n'$, the s_ith entry in the bitvector is $\mathbf{1}$, for $1 \leq i \leq t$; the remaining $n' - t$ bits are $\mathbf{0}$s. Thus, we can use subset rank (and unrank) primitives for encoding (and decoding) sublist $\langle x, y\rangle$ as a sequence r of $\lceil \lg \binom{n^x}{n^{x,y}} \rceil$ bits.

Instead, we will encode sublists one context at a time. In other words, we encode the sublists $\langle x, 1\rangle, \langle x, 2\rangle, \ldots, \langle x, \sigma\rangle$ at once. We encode each context x by encoding the string w_x (from Section 2.1), which consists of the symbols y that precede x, concatenated together in BWT order. One

simple method to encode w_x that we can use is a quasi-arithmetic coder from [HV94] (Theorem 1), requiring $\lg \binom{n^x}{n^{x,1}, n^{x,2}, \ldots, n^{x,\sigma}} + 2$ bits of space.

LEMMA 4.1. (QUASI-ARITHMETIC CODER [HV94]) *Suppose we know the values of n^x and $n^{x,1}, n^{x,2}, \ldots, n^{x,\sigma}$ for each context x. We can encode all contexts using one quasi-arithmetic coder for each context x taking just $nH_h + O(\sigma^h)$ bits of space. Decoding any context requires $O(n^x)$ operations on integers of size $O(\sigma)$.*

4.1 The Wavelet Tree In this section, we detail an alternate method of encoding each context x, motivated by applications to text indexing. We review the *wavelet tree* data structure, which is a binary tree structure that reduces the compression of a string from alphabet Σ to the compression of σ binary strings. We direct the reader to [GGV03] for more details and examples.

We will store one wavelet tree data structure for each context x, built on the string w_x, as used earlier. Recall that w_x is a string with alphabet Σ of length n^x. Each leaf represents one of the t^x nonempty sublists (equivalently, one of the t^x distinct symbols appearing in w_x) from context x. We implicitly consider each left branch to be associated with a **0** and each right branch to be associated with a **1**. Each internal node u is a t-subset data structure with the elements in its left subtree stored as **0**, and the elements in its right subtree stored as **1**. For instance, the root node conceptually stores a single bitvector B of length n^x represented by a t-subset data structure. If $B[p] = \mathbf{0}$, then the pth symbol in w_x is one of $1, \ldots, \sigma/2$. $B[p] = \mathbf{1}$ otherwise.

The key observation is to note that each of the $t^x - 1$ internal nodes represents elements relative to its subtrees. To decode any particular sublist in a wavelet tree, a query would only need to access $O(\lg t^x)$ internal nodes in a balanced wavelet tree. In particular, to recover the entries of any sublist $\langle x, y \rangle$, we start from the leaf corresponding to sublist $\langle x, y \rangle$ and examine the t-subsets in its ancestors.

LEMMA 4.2. (WAVELET TREE COMPRESSION) *Suppose we know the values of n^x and $n^{x,1}, n^{x,2}, \ldots, n^{x,\sigma}$ for each context x. We can encode all contexts using one wavelet tree for each context x taking just $nH_h + O(\sigma^{h+1})$ bits of space.*

One problem with our current implementation of the wavelet tree is its use of subset encoding using t-subsets, requiring $O(t)$ operations on large integers [Knu05, Rus05]. To solve this problem, we introduce our subset encoder in Section 4.2, which is a data structure of independent interest. It will replace the t-subsets in the wavelet tree, without adding any additional space.

4.2 Subset Encoding With Small Integers In this section, we describe a technique for subset encoding, storing a set S of t items out of a universe of size n such that it can be encoded or decoded using $O(t)$ operations on small integers (of size $O(\lg n)$ bits). As we described in Section 4, we can think of a t-subset as a succinct way to store an implicit bitvector. We could encode this bitvector using arithmetic (or quasi-arithmetic) coding, but encoding/decoding would require $O(n)$ operations.

Another approach is to encode the *gaps* between the items $s_1, s_2, \ldots, s_t$ that appear in the set S. (The ith gap is formally $s_i - s_{i-1}$, where $s_0 = 1$.) To encode the items, we associate a probability distribution for the different gap values, and encode each gap according to its probability using any of a number of techniques (say, for instance, the quasi-arithmetic coder from [HV94]). Using this method, the items are decoded *sequentially* using $O(t)$ operations.

We will generate the gaps sequentially. For this section, we redefine t to be the number of items left to encode out of a remaining universe of size n. In other words, the values of n and t will scale as we sequentially generate gaps. (As described in [Vit84], this won't be a problem.) We define X to be the random variable that determines the length of the next gap value to be generated. Note that the range of X is the set of integers in the interval $0 \leq x \leq n-t$. We will restrict gaps to a length of at most n/t and aggregate the probabilities of larger gaps into a single escape gap. If we need to encode an escape gap of length $g > n/t$, we reset $n = n - g - 1$ and continue processing. In other words, the range for X is $0 \leq x \leq n/t$.

One approach is to generate the gap X using the exact probabilities $f(x)$. The probability distribution function (pdf) for $f(x)$ is

$$f(x) = \begin{cases} \alpha_1 \frac{t}{n} \frac{(n-x-1)^{\underline{t-1}}}{(n-1)^{\underline{t-1}}} & \text{if } 0 \leq x < n/t; \\ \alpha_2 \frac{(n-n/t-1)^{\underline{t}}}{n^{\underline{t}}} & \text{if } x = n/t; \\ 0 & \text{otherwise,} \end{cases}$$

where we use the notation $a^{\underline{b}}$ to denote the *falling power* $a(a-1)\ldots(a-b+1) = a!/(a-b)!$. The constants α_1, α_2 are normalization factors so that $f(x)$ sums up to 1. Generating gaps according to this pdf requires large integer computations. Instead, we use a probability estimate $g(x)$ from [Vit84] that is easy to compute using the built-in logarithm functions. We define $g(x)$ as

$$g(x) = \begin{cases} \beta_1 \frac{t}{n}(1 - \frac{x}{n})^{t-1} & \text{if } 0 \leq x < n/t; \\ \beta_2 e^{-1} & \text{if } x = n/t; \\ 0 & \text{otherwise,} \end{cases}$$

where β_1 and β_2 are normalization factors so that $g(x)$ sums up to 1. Here, X can be generated

quickly with only one uniform or exponential random variable [Vit84]. We will use this probability estimate in the quasi-arithmetic coder to generate the gaps using only $O(t)$ operations on *small* integers (of size $O(\lg n)$ bits). However, we may spend additional bits to encode each gap, since our probabilities are only estimates for pdf $f(x)$. We address this point in the following lemma, showing that the worst-case difference between the encodings is quite small. The complete proof of this result is somewhat more technical and involves two probability estimates; we defer these details until the full paper.

LEMMA 4.3. *The number of extra bits needed to encode a gap $X = x$ using a probability estimate $g(x)$ instead of $f(x)$ is at most $O(1/n)$. Summed over all gaps, the total encoding cost is at most $O(1)$ bits of additional space using an arithmetic coder.*

Proof. We provide a sketch of the proof in this abstract. The details will be in the full version of the paper.

We look at the worst case where we encode a gap $X = x$ using the probability estimate $g(x)$ rather than $f(x)$. The extra bits needed to encode any gap is $\lg(1/g(x)) - \lg(1/f(x)) = \lg(f(x)/g(x))$. We write

$$\begin{aligned}\lg(f(x)/g(x)) &\leq \lg\left(\frac{n-x-1}{(n-1)(1-x/n)}\right)^{t-1} \\ &= \lg\left(1+\frac{x}{n(n-x-1)}\right)^{1-t}.\end{aligned}$$

The worst-case ratio of $f(x)/g(x)$ occurs when $x = n/t$. Substituting and using simple algebra, we arrive at the result when $t \leq \sqrt{(}n)$. In the full version of the paper, we describe a second estimate with the same result when $t > \sqrt{(}n)$, thus showing the result.

We also need several other properties so that we can use a quasi-arithmetic coder; we defer this discussion until the full version of the paper. Putting together the details, we arrive at Theorem 4.1, which describes our subset-encoding scheme.

THEOREM 4.1. (SMALL INTEGER SUBSETS) *Suppose each of the t items is drawn from $[1..n]$, where we already know t and n (and do not need to encode them). Then, there exists an encoding of a subset S of t items drawn from $[1..n]$ that requires $\lg\binom{n}{t} + O(1)$ bits of space and can be encoded or decoded in $O(t)$ operations on small integers, of size $O(\lg n)$ bits.*

5 The Empirical Statistical Model

In this section, we present the second part of the proof of Theorem 3.1. In Section 3, we described a method of storing the BWT using $nH_h + M(T, \Sigma, h)$ bits. Our scheme was divided into two components: the encoding of a series of small disjoint subtexts, one for each context x, and the encoding of the length of each subtext, together with the statistics of each subtext. We did not analyze the cost required to store this empirical statistical information. We briefly recap now:

- For each context x, the storage for step 2 uses fewer than $\lg\binom{n^x}{n^{x,1}, n^{x,2}, \ldots, n^{x,\sigma}} + t^x$ bits by Lemma 4.1 and 4.2. We use equation (2.4) and Theorem 2.1 to bound the above term by $nH'_h + |P^*_h|\sigma$ for all contexts $x \in P^*_h$ in the worst case. Since $|P^*_h| \leq \sigma^h$, we bound the space required to store the BWT by $nH_h + \sigma^{h+1}$ bits.
- To decode step 2, we need to know the number of symbols of each type stored in each subtext for context x. Collectively, this information maintains the empirical statistical model used to achieve hth order entropy. We call its encoding length $M(T, \Sigma, h)$ (in bits), and we are interested in discovering how succinctly this information can be stored. Thus, the storage for step 1 is $M(T, \Sigma, h)$ bits.[4]

Our storage of the BWT requires $nH_h + \sigma^{h+1} + M(T, \Sigma, h)$ bits, and bounding the quantity $M(T, \Sigma, h)$ may help in understanding the compressible nature of the BWT. We will devote the rest of this section to developing two bounds for the storage of the empirical statistical model. One benefit of pursuing bounds in this framework is that it simplifies the burden of analysis: namely, it translates the overhead costs of the BWT into the cost for encoding the integer lengths $n^{x,y}$.

5.1 Definitions and a Simple Bound In this section, we describe a simple encoding for the empirical statistical model, which takes $M(T, \Sigma, h)$ bits to encode. Recall from Definition 1 in Section 2.1 that the empirical statistical model encodes two items: the partition of Σ^h induced by the optimal prefix cover P^*_h, and the sequence of lengths $n^{x,y}$ of the sublists. The partition is easily stored using a bitvector of length σ^h (or a subset encoding of the partition using $\lceil \lg\binom{\sigma^h}{|P^*_h|} \rceil \leq \sigma^h$ bits). To store the sequence of lengths $n^{x,y}$, we simply store the concatenation of the gamma codes for each length $n^{x,y}$ and bound its length.

We briefly review Elias' gamma and delta codes [Eli75] before detailing the proof. The gamma code for a positive integer ℓ represents ℓ in two parts: the first encodes $1 + \lfloor \lg \ell \rfloor$ in unary, followed by the value of $\ell - 2^{\lfloor \lg \ell \rfloor}$ encoded in binary, for a total of $1 + 2\lfloor \lg \ell \rfloor$ bits. For example, the gamma codes for $\ell = 1, 2, 3, 4, 5, \ldots$ are

[4] In this section, we will show that $M(T, \Sigma, h) \geq \sigma^{h+1}$.

$\mathbf{1}, \mathbf{01\,0}, \mathbf{01\,1}, \mathbf{001\,00}, \mathbf{001\,01}, \ldots$, respectively. The delta code requires fewer bits asymptotically by encoding $1 + \lfloor \lg \ell \rfloor$ via the gamma code rather than in unary. For example, the delta codes for $\ell = 1, 2, 3, 4, 5, \ldots$ are $\mathbf{1}, \mathbf{010\,0}, \mathbf{010\,1}, \mathbf{011\,00}, \mathbf{011\,01}, \ldots$, and require $1 + \lfloor \lg \ell \rfloor + 2\lfloor \lg \lg 2\ell \rfloor$ bits. Now, we describe a simple upper bound on encoding the empirical statistical model.

LEMMA 5.1. *The empirical statistical model requires* $M(T, \Sigma, h) = O\left(\sigma^{h+1} \lg(1 + n/\sigma^{h+1})\right)$ *bits.*

Proof. In this encoding, we represent the lengths using the gamma code. We obtain a bitvector Z by concatenating the gamma codes for $n^{x,1}, n^{x,2}, \ldots, n^{x,\sigma}$ for $x = 1, 2, \ldots, |P_h^*|$. The bitvector Z contains $O(\sum_{x \in P_h^*, y \in \Sigma} \lg n^{x,y})$ bits; this space is maximized when all lengths $n^{x,y}$ are equal to $\Theta(n/(|P_h^*| \times \sigma) + 1)$ by Jensen's inequality [CT91]. Since $|P_h^*| \leq \sigma^h$, we bound the total space by $O\big((|P_h^*| \times \sigma) \lg\big(1 + n/(|P_h^*| \times \sigma)\big)\big) = O\big(\sigma^{h+1} \lg(1 + n/\sigma^{h+1})\big)$ bits. We do not need to encode n as it can be recovered from the sum of the sublists lengths.

The result of Lemma 5.1 is interesting, but it carries a dependance on n, unlike the bounds in related work, which are related only to σ and h [FGMS05]. In Section 5.2, we show an alternate analysis that remedies this problem and relates the encoding costs to the modified entropy nH_h^*, as defined in Section 2.1.

5.2 Nearly Tight Upper Bound on $M(T, \Sigma, h)$

In this section, we describe a nearly tight upper bound for encoding the empirical statistical model. As we described in Section 5.1, we can easily store the partition of Σ^h induced by the optimal prefix cover P_h^* using at most σ^h bits. We provide a new analysis for storing the sequence of lengths $n^{x,y}$ in Theorem 5.1.

THEOREM 5.1. *The empirical statistical model requires at most* $M(T, \Sigma, h) \leq nH_h^* + \lg n + O\left(\sigma^{h+1} \lg \sigma^{h+1}\right)$ *bits of space.*

The results of Theorem 5.1 highlight a remarkable property of the Burrows-Wheeler Transform, namely that maintaining the statistics of the text requires *more* space than the actual encoding of the information.

To prove Theorem 5.1, we have to encode the sequence of sublist lengths $n^{x,1}, n^{x,2}, \ldots, n^{x,\sigma}$, where $x = 1, 2, \ldots, |P_h^*|$ and $\sum_{x \in P_h^*, y \in \Sigma} n^{x,y} = n$. We use the following encoding scheme for each context x:

- If context x contains a single nonempty sublist y, we use σ bits to mark the yth sublist as nonempty. Then, we store the length $n^{x,y} = n^x$.
- If context x contains two or more nonempty sublists, we again use σ bits to mark the nonempty sublists. To describe the rest of the method, let $n'_1 = n^x$ and $n'_j = n^x - \sum_{i=1}^{j-1} n^{x,i}$ be a scaled universe where $j \geq 2$. We use σ bits for context x, one bit per sublist. The bit for sublist y is **1** if and only if $n^{x,y} > n'_y/2$; in this case, we set $t_y = n'_y - n^{x,y}$. Otherwise, we set the bit for sublist y to **0** and set $t_y = n^{x,y}$. Notice that $t_y \leq n'_y/2$ in both cases. Now, we encode t using its delta code. Given n'_y and t_y, we can recover the value of $n^{x,y}$ as expected.

LEMMA 5.2. *We can encode the sublist lengths* $n^{x,1}, n^{x,2}, \ldots, n^{x,\sigma}$ *for any context* x *with two or more nonempty sublists using at most* $\gamma n^x H_0^*(w_x) + O(\sigma)$ *bits, where* $0 < \gamma < 1/2$ *is a constant.*

Proof. Our scheme requires 2σ bits to store auxiliary information. Now we bound the total size of encoding the values $t_1, t_2, \ldots, t_\sigma$ using the delta code for each nonempty sublist. Our approach is to amortize the cost of writing the delta code of t_y with the encoding of its associated sublist y. We introduce some terminology to clarify the proof. For any arbitrarily fixed constant γ with $0 < \gamma < 1/2$, let $t_\gamma > 0$ be constants such that for any integer $t > t_\gamma$, $\lg t + 2 \lg \lg(2t) + 1 < \gamma(2t - \lg t - 1)$.

Then, for nonempty sublists y with $t_y \leq t_\gamma$, the delta code for t_y will take $O(\lg t_\gamma) = O(1)$ bits of space. Summing these costs for all such sublists, we would require at most $O(\sigma \lg t_\gamma) = O(\sigma)$ bits for context x.

For nonempty sublists y with $t_y > t_\gamma$, we use at most $\gamma(2t_y - \lg t_y - 1)$ bits to write the delta code of t_y using the observation above.

Now, we will use the fact that $t_y \leq n'_y/2$ for each sublist y in our scheme to bound the encoding length of sublist y, and then amortize accordingly. In general, for any $1 < t < n/2$, $\lg \binom{n}{t} \geq \lg \binom{2t}{t} \geq \lg(2^{2t}/2t) = 2t - \lg t - 1$. Since each sublist y with $t_y > t_\gamma$ satisfies this condition by the construction of our scheme, we can bound the delta code of t_y by $\gamma \lg \binom{n'_y}{t_y}$ bits. Summing over all such sublists for context x, we would require at most $\gamma \lg \binom{n^x}{n^{x,1}, n^{x,2}, \ldots, n^{x,\sigma}} + \sigma = \gamma n^x H_0^*(w_x) + \sigma$ bits using the analysis from Section 4.1, thus proving the lemma.

The above scheme requires us to store the length n^x of each context x, since the sum of the t_y values we store may be less than n^x. (For the case with a single nonempty context, n^x is the size of the only sublist.) For example, suppose for some context x, $n^x = 20$, $n^{x,1} = 11$, $n^{x,2} = 3$, $n^{x,3} = 5$, $n^{x,4} = 1$. According to our scheme, we would store the t_y values 9, 3, 1, and 1, which sum up to $14 < 20$. To determine the value of $n^{x,1}$, we must therefore compute $n^x - t$; thus, we must know the value of n^x.

The storage of n^x is a subtle but important point, and it is a key component in understanding the *lower bound* on encoding length for the BWT, which we discuss more in Section 6. In Lemma 5.3, we describe a technique to store the sequence of lengths n^x for $x = 1, 2, \ldots, |P_h^*|$ using $\lg n^x$ bits, plus some small additional costs. We use this result to give a simple bound on the space of our encoding scheme in Lemma 5.4.

LEMMA 5.3. *The sequence of lengths n^x for $x = 1, 2, \ldots, |P_h^*|$ can be stored using $\sum_{x \in P_h^*} \lg n^x + \lg n + O(\sigma^{h+1} \lg \sigma^{h+1})$ bits of space.*

Proof. For each context x with n^x entries, we encode its length n^x in binary using $b(x) = \lfloor \lg n^x \rfloor + 1$ bits. These $b(x)$ bits do not permit a decoding of n^x by themselves, since they are not prefix codes. We describe how to fix this problem. We permute the contexts x so that they are sorted by their $b(x)$ values. Now, contexts requiring the same number of bits $b(x)$ to store their lengths are contiguous. In other words, we know that for any two consecutive contexts x and x' in the sorted order, either $b(x) = b(x')$ or $b(x) < b(x')$. What remains is the storage of the positions where $b(x) < b(x')$. We store this information using $|P_h^*|$ bits.

To remember which lengths $b(x)$ actually occur, we observe that the number of distinct lengths is at most $\lg n + 1$, since $1 \leq b(x) \leq \lg n + 1$. We store a bitvector of length $\lg n + 1$ bits to keep track of this information. Finally, we store the permutation to restore the original order of the contexts using $O(\lg |P_h^*|!) = O(\sigma^{h+1} \lg \sigma^{h+1})$ bits, thus proving the lemma.

LEMMA 5.4. *The empirical statistical model requires at most $(1+\gamma)nH_h^* + \lg n + O(\sigma^{h+1} \lg \sigma^{h+1})$ for all contexts x, where $0 < \gamma < 1/2$ is a constant.*

Proof. Using the definition of modified hth-order empirical entropy in equation (2.2), we bound the first term in Lemma 5.3 by $\sum_{x \in P_h^*} \lg n^x \leq nH_h^*$. According to our scheme, storing the length n^x along with σ bits is sufficient to encode any context x with a single nonempty sublist. For the remaining contexts, we apply Lemma 5.2 to achieve the desired result.

We can further improve our bound by amortizing the cost of storing the length n^x for context x with the encoding of its sublists. The technique is reminiscent of the one we used in Lemma 5.2. We change our encoding scheme as follows.

- If context x contains a single nonempty sublist y, we use σ bits to mark the yth sublist as nonempty. Then, we store the length $n^{x,y} = n^x$.
- If context x contains two or more nonempty sublists, we use the scheme below.
- Let t_γ be defined as in Lemma 5.2. For any arbitrarily fixed constant γ with $0 < \gamma < 1/2$, let $n_\gamma > 0$ be a constant such that for any integer $n > n_\gamma$ and $t > t_\gamma$ with $t \leq n/2$, $\binom{n}{t} \geq \frac{n(n-1)\ldots(n-\lceil 1/\gamma \rceil)}{(\lceil 1/\gamma \rceil + 1)!} > n^{\lceil 1/\gamma \rceil}$.
- Instead of encoding $n^{x,1}$ as the first sublist length for context x, we use σ bits to indicate that we encode n^{x,y_b} first, where $t_b = \min\{n^{x,y_b}, n^x - n^{x,y_b}\}$ satisfies the condition $\binom{n^x}{t_b} > (n^x)^{\gamma^{-1}}$. If no such y_b exists, we encode $n^{x,1}$ as before.

LEMMA 5.5. *We can encode the sublist lengths $n^{x,1}, n^{x,2}, \ldots, n^{x,\sigma}$ along the context length n^x for any context x with two or more nonempty sublists using at most $n^x H_0^*(w_x) + O(\sigma)$ bits.*

Proof. The cost for encoding the sublist lengths is analyzed using Lemma 5.2. We focus on bounding the cost for $\lg n^x$. If any sublist y_b satisfies the constraint in our scheme, we know that $\lg n^x < \gamma \lg \binom{n^x}{t_b}$, which is the same upper bound on the number of bits required to encode t_b. Thus, encoding both n^{x,y_b} and n^x will take $2\gamma \lg \binom{n^x}{t_b} + O(\sigma)$ bits of space. The encoding size for the rest of the new sequence remains the same as we observed in Lemma 5.2, thus we require at most $2\gamma n^x H_0^*(w_x) + O(\sigma)$ bits. Since $\gamma < 1/2$, this shows the bound for contexts x that satisfy the constraint.

If no sublist satisfied the constraint, then we know that each $t_i \leq t_\gamma (1 \leq i \leq \sigma)$ so the delta code for each t_i takes $O(\lg t_\gamma) = O(1)$ bits, which take at most $O(\sigma)$ bits overall. Then, the $\lg n^x$ bits for encoding n^x can be bounded by $n^x H_0^*(w_x)$ as in Lemma 5.4, since $n^x H_0^*(w_x) \geq \lg n^x$. This case will contribute at most $n^x H_0^*(w_x) + O(\sigma)$ bits to the bound, thus proving the bound.

Combining Lemma 5.5 with our encoding for the singleton context, we prove Theorem 5.1 and Theorem 3.1.

6 Nearly Tight Bounds for the BWT

Manzini conjectures that the BWT cannot be compressed to just $nH_h^* + \lg n + g_h$ bits of space, where $g_h = O(\sigma^{h+1} \lg \sigma)$. However, in Section 5.2, we provide an analysis that gives an upper bound of $n(H_h + H_h^*) + \lg n + g_h''$ bits, where $g_h'' = O(\sigma^{h+1} \lg \sigma^{h+1})$. Since there are an infinite number of strings where $nH_h = 0$ but $nH_h^* \neq 0$, our bound is $nH_h + M(T, \Sigma, h) \leq nH_h^* + \lg n + g_h''$ in these cases, matching Manzini's conjectured lower bound (but not for all strings).

In this section, we will explore other classes of strings that help establish a non-trivial lower bound on the compressibility of the BWT. Surprisingly, the encoding of the BWT requires an amount of space very close to our encoding length for the upper bound. In

particular, we will prove the following theorem, which shows that our upper bound analysis is nearly tight.

THEOREM 6.1. *For any chosen positive constants $\delta \leq 1$ and $k > \lceil 1/\delta \rceil = d$, there exists an infinite family of strings such that for any string of length n in the family, its encoding length $nH_h + M(T, \Sigma, h)$ satisfies the following two relations:*

(6.9)
$$nH_h^* + \frac{k-1}{k}\delta n H_h - O(\text{poly}(kd)) \leq nH_h + M(T, \Sigma, h)$$

$$nH_h + M(T, \Sigma, h) \leq nH_h^* + \delta n H_h + \lg n + g_h''. \quad (6.10)$$

When $\delta > 1$, we use Theorem 3.1 as the upper bound for (6.10). To prove inequality (6.10), we give a tighter analysis of the space-intensive part of the encoding scheme from Section 5. To capture the primary challenge from Section 5, we define a *δ-resilient* text. Let BWT(T) denote the result of applying the BWT to the text T. For any given constant δ such that $0 < \delta \leq 1$, the text T is *δ-resilient* if the optimal partition induced by P_h^* for BWT(T) satisfies $\max_{y \in \Sigma}\{n^{x,y}\} \leq n^x - \lceil 1/\delta \rceil$ for every context $x \in P_h^*$. In other words, no partition x of BWT(T) induced by P_h^* contains more than $n^x - \lceil 1/\delta \rceil$ identical symbols. We define $d = \lceil 1/\delta \rceil$. Now, we apply Theorem 5.1 to δ-resilient texts and achieve the following lemma.

LEMMA 6.1. *For any constant δ with $0 < \delta \leq 1$ and any δ-resilient text T of n symbols over Σ, we have $nH_h + M(T, \Sigma, h) \leq n(\delta H_h + H_h^*) + \lg n + g_h''$.*

To prove our lower bound inequality (6.9) from Theorem 6.1, we describe a construction scheme that takes user-defined parameters and creates a δ-resilient text T of length n. We will then count the total number of δ-resilient texts that our construction scheme generates, and use a combinatorial argument to bound the space required to distinguish between these texts.

6.1 Constructing δ-resilient Texts In this section, we describe how to construct δ-resilient texts using a generalized construction scheme; then, we will use the resulting class of texts to prove inequality (6.9) of Theorem 6.1. First, we define some terminology that will help clarify the discussion. Let $d = \lceil 1/\delta \rceil$, where $0 < \delta \leq 1$ is a constant. Let T_s be a support text of length n_s composed of an alphabet $\Sigma = \{\mathtt{a}_1, \mathtt{a}_2, \ldots, \mathtt{a}_k, \mathtt{b}, \mathtt{c}_1, \mathtt{c}_2, \ldots, \mathtt{c}_k, \#\}$, where $k = O(\text{polylg}(n)) > d$ is a fixed positive integer. We assume that $\mathtt{a}_i < \mathtt{a}_{i+1} < \mathtt{b} < \mathtt{c}_j < \mathtt{c}_{j+1} < \#$ for all i and j. We define T_s as

$$T_s = \underbrace{(\mathtt{a}_1\mathtt{c}_1)^{\ell_1}}_{r_1}\ \underbrace{(\mathtt{a}_2\mathtt{c}_2)^{\ell_2}}_{r_2}\ \ldots\ \underbrace{(\mathtt{a}_k\mathtt{c}_k)^{\ell_k}}_{r_k},$$

where each length parameter $\ell_i \geq d$. We define a run r_i as the sequence of ℓ_i substrings of the form $\mathtt{a}_i\mathtt{b}^*\mathtt{c}_i$. In T_s, $\mathtt{b}$ never appears. The length of the support string T_s is $n_s = 2\sum_{i=1}^{k} \ell_i$. We now prove the following lemma.

LEMMA 6.2. *The* BWT(T_s) *is*

$$\text{BWT}(T_s) = \overbrace{\underbrace{\mathtt{c}_k(\mathtt{c}_1)^{\ell_1 - 1}}_{P_1}\ \underbrace{\mathtt{c}_1(\mathtt{c}_2)^{\ell_2 - 1}}_{P_2}\ \ldots\ \underbrace{\mathtt{c}_{k-1}(\mathtt{c}_k)^{\ell_k - 1}}_{P_k}}^{B_1}$$
$$\overbrace{\underbrace{(\mathtt{a}_1)^{\ell_1}}_{Q_1}\ \underbrace{(\mathtt{a}_2)^{\ell_2}}_{Q_2}\ \ldots\ \underbrace{(\mathtt{a}_k)^{\ell_k}}_{Q_k}}^{B_2},$$

where $B_1 = P_1P_2 \ldots P_k$ as the first block of the BWT *transform, and $B_2 = Q_1Q_2 \ldots Q_k$ as the second block. Here, P_i refers to the positions of the* BWT *corresponding to strings that start with symbol $\mathtt{a}_i$, and Q_i refers to positions of the* BWT *corresponding to strings that start with symbol $\mathtt{c}_i$.*

Proof. Consider the strings in the BWT matrix M, sorted in lexicographical order. According to the rank of symbols in alphabet Σ, all strings beginning with $\mathtt{a}_i$ will precede strings before $\mathtt{a}_{i+1}$. Similarly, strings beginning with $\mathtt{c}_i$ will precede strings beginning with $\mathtt{c}_{i+1}$. Finally, all strings beginning with $\mathtt{a}_i$ will precede strings beginning with $\mathtt{c}_1$. Also, there are exactly ℓ_i strings that begin with $\mathtt{a}_i$ and $\mathtt{c}_i$. We now focus on the strings that begin with $\mathtt{c}_i$.

Each string beginning with $\mathtt{c}_i$ has the symbol $\mathtt{a}_i$ preceding it (or equivalently, at the end of the string) in all cases. Thus, the part of the BWT corresponding to strings beginning with $\mathtt{c}_i$ is $(\mathtt{a}_i)^{\ell_i}$. Collectively, we call this block B_2.

Each string beginning with $\mathtt{a}_i$ has the symbol $\mathtt{c}_i$ preceding it (or at the end of the string, since it's cyclic), except the string corresponding to the first $\mathtt{a}_i$ in run r_i. This string is lexicographically the first string among all of the strings beginning with $\mathtt{a}_i$ and is preceded by $\mathtt{c}_{i-1}$ or $\mathtt{c}_k$ if $i = 1$. Thus, the part of the BWT corresponding to strings beginning with $\mathtt{a}_i$ is $\mathtt{c}_{i-1}(\mathtt{c}_i)^{\ell_i - 1}$. If $i = 1$, $\mathtt{c}_{i-1}$ is replaced with $\mathtt{c}_k$. Collectively, we call this block B_1.

Thus, the lemma is proved.

Now, we introduce $d = \lceil 1/\delta \rceil$ partition vectors $v_i = \langle v_i[1], v_i[2], \ldots v_i[k] \rangle$ which will generate a δ-resilient property for B_2; B_1 remains unchanged, but will implicitly encode the length of the corresponding portions of B_2. We augment T_s as follows: for each entry of v_i for all i, we replace the $v_i[j]$th occurrence of the string $\mathtt{a}_j\mathtt{c}_j$ with $\mathtt{a}_j\mathtt{b}\mathtt{c}_j$. We will make d such replacements in each of the k partitions. We call this augmented text T_s', of length $n_s' = n_s + dk$.

LEMMA 6.3. *The* BWT(T'_s) *is*

$$\text{BWT}(T'_s) = \overbrace{P'_1P'_2\dots P'_k}^{B_1}\ \overbrace{(\mathtt{a}_1)^d(\mathtt{a}_2)^d\dots(\mathtt{a}_k)^d}^{A}\ \overbrace{Q'_1Q'_2\dots Q'_k}^{B_2},$$

where P'_i is composed of symbols preceding strings that start with $\mathtt{a}_i$, A is composed of symbols preceding strings that start with b*, and Q'_i is composed of symbols preceding strings that start with $\mathtt{c}_i$.*

Proof. This proof is similar to Lemma 6.2, where each string in P_i precedes strings in P_{i+1}. Here, all strings in P'_i precede strings in P'_{i+1}, strings in Q'_i precede strings in Q'_{i+1}, and strings in P'_i precede strings beginning with b (called A) and strings in A precede strings in Q'_1.

Then, P'_i is a string of length ℓ_i similar to P_i, but the single occurrence of $\mathtt{c}_{i-1}$ (or $\mathtt{c}_k$ if $i = 1$) could be in any of the ℓ_i positions. Also, Q'_i is a string of length ℓ_i where d positions contain the symbol b, and all others are $\mathtt{a}_i$. Block A consists of exactly d occurrences of each $\mathtt{a}_i$ sorted in lexicographical order, since all d strings beginning with $\mathtt{bc}_i$ precede all strings beginning with $\mathtt{bc}_{i+1}$, thus finishing the proof.

A simple verification will show that blocks A and B_2 are δ-resilient portions of T'_s. Furthermore, block A is deterministic once the parameters d and k have been chosen; block B_1 encodes the length of each Q'_i. To have a fully δ-resilient text, we want B_1 to have the same property, so we generate the string $T = T'_s(T_s)^{d-1}\#$. This will include $d-1$ occurrences of a different symbol inside each P'_1. Note that $|T| = n = dn_s + dk + 1$.

LEMMA 6.4. *The* BWT(T) *is*

$$\text{BWT}(T) = \overbrace{P''_1P''_2\dots P''_k}^{B_1}\ A\ \overbrace{Q''_1Q''_2\dots Q''_k}^{B_2}\ \mathtt{c}_k,$$

where P''_i is composed of symbols preceding strings that start with $\mathtt{a}_i$, A is composed of symbols preceding strings that start with b*, and Q''_i is composed of symbols preceding strings that start with $\mathtt{c}_i$.*

Proof. The strings P''_i and Q''_i are of length $d\ell_i$. Similar to the arguments in Lemma 6.3, P''_1 consists of the symbol $\mathtt{c}_1$ in all but $d\ell_1 - d$ positions; one position contains # and the other $d-1$ positions contain $\mathtt{c}_k$. P''_i consists of the symbol $\mathtt{c}_i$ in all but $d\ell_i - d$ positions; the other d positions contain $\mathtt{c}_{i-1}$. Each Q''_i is similar to the previous case, except its length is now $d\ell_i$. Q''_i still contains only d occurrences of b.

Finally, the last $\mathtt{c}_k$ is the symbol preceding # in the text, which is lexicographically the largest symbol, and therefore the last string represented in the BWT, thus finishing the proof.

6.2 Encoding a δ-resilient Text In this section, we analyze the space required to store a δ-resilient text T. Since B_1 and A are deterministic once d and k are chosen, we focus only on the encoding cost of B_2. First, we prove the following lemma.

LEMMA 6.5. *For any set of p objects, at least half of them will take at least $\lg p - 1$ bits to encode so that the objects can be distinguished from one another.*

Proof. Since one can distinguish at most 2^j objects from one another using j bits, the most succinct encoding would greedily store two objects using one bit each, four objects using two bits each, and so on. Thus, we need to make sure that $\sum_i^j 2^i \geq p$. Thus, $j + 1 \geq \lg p$, and the lemma follows.

Let Λ be the set of all possible choices λ of length parameters $\ell_1, \ell_2, \dots, \ell_k$ used to generate δ-resilient texts in Section 6.1. By construction, $|\Lambda| = \binom{n_s/2 - dk + k - 1}{k-1}$. For a given choice λ of parameters, we choose d positions in each partition Q''_i that will contain a b. However, we are only choosing from the first ℓ_i positions for each run r_i (i.e. the positions that correspond to the entries in T'_s). Once these positions are chosen, we perform the steps described in our construction scheme. Since the BWT is a reversible transform, we have $\binom{\ell_i}{d}$ possible partitions Q''_i and our construction scheme generates one of

$$X = \sum_{\lambda\in\Lambda} \binom{\ell_1}{d}\binom{\ell_2}{d}\dots\binom{\ell_k}{d}$$

different texts. We let an adversary encode the X texts in any way he wishes. Then, we use Lemma 6.5 to consider only half of these texts, namely the ones that take at least $\lg X - 1$ bits to encode. Now we analyze the quantity $\lg X - 1$.

To help analyze $\lg X - 1$, we divide Λ into two sets Y and Z of equal cardinality, such that for any texts $y \in Y$ and $z \in Z$, the product $p(y) \leq p(z)$, where $p(T) = \prod_1^k \binom{\ell_i}{d}$. In words, Y contains the texts T where $p(T)$ is smaller, and Z contains the ones where $p(T)$ is larger. We take a single arbitrary text S from set Y and determine which choice λ^* of length parameters ℓ_i was used. We separate the k terms corresponding to λ^* from $\lg X - 1$ and analyze their cost separately. The terms are $\sum_1^k \lg \binom{\ell_i}{d} = nH'_1(S)$, by our definition of finite set empirical entropy. Since $nH'_h(S) \leq nH'_1(S)$, the contribution of this part of $\lg X - 1$ is at least $nH'_h(S)$ bits. We translate this into a bound in terms of nH^*_h using the following lemma.

LEMMA 6.6. *For a δ-resilient text, $nH_h^* - \Theta(k \lg d) \leq nH_h'$.*

Proof. It suffices to show that $nH_0^* - \Theta(\lg d) \leq nH_0'$ for each partition Q_i'' in a δ-resilient text, since there are at most $2k+1$ partitions. We apply Stirling's double inequality to the expression $\lg \binom{\ell_i}{d}$ and find that

$$\begin{aligned} \lg \binom{l_i}{d} &> \ell_i H_0 + \frac{1}{2} \lg \frac{\ell_i}{d(\ell_i - d)} - O(1) \\ &> \ell_i H_0 + \frac{1}{2} \lg \frac{1}{d} - O(1), \end{aligned}$$

thus proving the lemma.

Thus, the total contribution of the part of $\lg X - 1$ corresponding to the text S is at least $nH_h^*(S) - \Theta(k \lg d)$ bits. Now we bound the term X to figure out the entire cost of encoding the string S. We will lower bound X by the sum for just the set Z and obtain

$$\begin{aligned} X &\geq \sum_{z \in Z} \prod_{1}^{k} \binom{\ell_i}{d} \\ &\geq \sum_{z \in Z} p(S) \\ &= \frac{1}{2} \binom{n_s/2 - dk + k - 1}{k - 1} p(S). \end{aligned}$$

Taking logs, we require $nH_h^*(S) + \lg \binom{n_s/2 - dk + k - 1}{k-1} - 1$ bits of space.

To finish the proof, we analyze the contribution of the term $\lg \binom{n_s/2 - dk + k - 1}{k-1}$. For ease of notation, let $g = n_s/2 - dk + k - 1$. We want to show that $(k-1) \lg g/(k-1) \leq \lg \binom{g}{k-1}$. The claim is true by inspection when $g \leq 4$ or $k - 1$ is $0, 1$, or $g - 1$. For the remainder of the cases, we apply Stirling's inequality as in Theorem 2.1 to verify the claim. Now, $(k-1) \lg g/(k-1) \geq (k-1) \lg n_s/2 - (k-1) \lg(dk) - (k-1) \lg k$. Thus, the contribution of this part of $\lg X - 1$ is at least $(k-1) \lg n - \Theta(k \lg(dk))$ bits, proving inequality (6.9) and Theorem 6.1 for any arbitrary δ-resilient text S.

References

[BB04] D. Baron and Y. Bresler. An $o(n)$ semipredictive universal encoder via the bwt. *IEEE Transactions on Information Theory*, 50:928–937, 2004.

[BW94] M. Burrows and D.J. Wheeler. A block sorting data compression algorithm. Technical report, Digital Systems Research Center, 1994.

[CT91] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. Wiley-Interscience, New York, 1991.

[Eli75] Peter Elias. Universal codeword sets and representations of the integers. *IEEE Transactions on Information Theory*, IT-21:194–203, 1975.

[EVKV02] Michelle Effros, Karthik Visweswariah, Sanjeev R. Kulkarni, and Sergio Verdu. Universal lossless source coding with the burrows-wheeler transform. *IEEE Transactions on Information Theory*, 48(5):1061–1081, 2002.

[Fel68] William Feller. *An Introduction to Probability Theory and its Applications*, volume 1. John Wiley & Sons, New York, 3rd edition, 1968.

[FGMS05] Paolo Ferragina, Raffaele Giancarlo, Giovanni Manzini, and Gabriella Sciortino. Boosting textual compression in optimal linear time. *Journal of the ACM*, 52(4):688–713, 2005. (Also in CPM 2003, ACM-SIAM SODA 2004.).

[FM05] Paolo Ferragina and Giovanni Manzini. On compressing and indexing data. *Journal of the ACM*, 52(4):552–581, 2005. (Also in IEEE FOCS 2000.).

[GGV03] Roberto Grossi, Ankur Gupta, and Jeffrey Scott Vitter. High-order entropy-compressed text indexes. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms*, January 2003.

[GV05] Roberto Grossi and Jeffrey Scott Vitter. Compressed suffix arrays and suffix trees with applications to text indexing and string matching. *SIAM Journal on Computing*, 35(2):378–407, 2005.

[HV94] Paul G. Howard and Jeffrey Scott Vitter. Arithmetic coding for data compression. *Proceedings of the IEEE*, 82(6), June 1994.

[KLV06] Haim Kaplan, Shir Landau, and Elad Verbin. A simpler analysis of burrows-wheeler based compression. pages 282–293, 2006.

[KM99] S. Rao Kosaraju and Giovanni Manzini. Compression of low entropy strings with lempel-ziv algorithms. *SIAM J. Comput.*, 29(3):893–911, 1999.

[Knu05] Donald E. Knuth. *Combinatorial Algorithms*, volume 4 of *The Art of Computer Programming*. Addison-Wesley, Reading, MA, USA, 2005. In preparation.

[Man01] Giovanni Manzini. An analysis of the Burrows — Wheeler transform. *Journal of the ACM*, 48(3):407–430, May 2001.

[NM06] Gonzalo Navarro and Veli Mäkinen. Compressed full-text indexes. 2006. To appear.

[Rus05] Frank Ruskey. *Combinatorial Generation*. 2005. In preparation.

[Sha48] Claude E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423, July 1948.

[Vit84] Jeffrey Scott Vitter. Faster methods for random sampling. *Communications of the ACM*, 27(7):703–718, July 1984.

[WMF94] Marcelo J. Weinberger, Neri Merhav, and Meir Feder. Optimal sequential probability assignment for individual sequences. *IEEE Transactions on Information Theory*, 40:384–396, 1994.