Published by Algorithmica February 22, 2012

Sublinear Algorithms for Approximating String Compressibility

Sofya Raskhodnikova · Dana Ron · Ronitt Rubinfeld · Adam Smith

the date of receipt and acceptance should be inserted later

Abstract We raise the question of approximating the compressibility of a string with respect to a fixed compression scheme, in sublinear time. We study this question in detail for two popular lossless compression schemes: run-length encoding (RLE) and a variant of Lempel-Ziv (LZ77), and present sublinear algorithms for approximating compressibility with respect to both schemes. We also give several lower bounds that show that our algorithms for both schemes cannot be improved significantly.

Our investigation of LZ77 yields results whose interest goes beyond the initial questions we set out to study. In particular, we prove combinatorial structural lemmas that relate the compressibility of a string with respect to LZ77 to the number of distinct short substrings contained in it (its ℓth *subword complexity*, for small ℓ). In addition, we show that approximating the compressibility with respect to LZ77 is related to approximating the support size of a distribution.

A preliminary version of this paper appeared in the proceedings of RANDOM 2007 [36].
This research was done while S.R. was at the Hebrew University of Jerusalem, Israel, supported by the Lady Davis Fellowship, and while both S.R. and A.S. were at the Weizmann Institute of Science, Israel. A.S. was supported at Weizmann by the Louis L. and Anita M. Perlman Postdoctoral Fellowship. Currently, S.R. is supported by NSF/CCF CAREER award 0845701 and A.S., by NSF/CCF CAREER award 0747294. D.R. is supported by the Israel Science Foundation (grant number 89/05).

S. Raskhodnikova
Pennsylvania State University, USA.
Tel.: +1-814-863-0608
Fax: +1-814-865-7647
E-mail: sofya@cse.psu.edu

D. Ron
Tel Aviv University, Israel.
E-mail: danar@eng.tau.ac.il

R. Rubinfeld
MIT, USA.
E-mail: ronitt@csail.mit.edu

A. Smith
Pennsylvania State University, USA.
E-mail: asmith@cse.psu.edu

Keywords Sublinear algorithms · Lossless compression · Run-length encoding · Lempel-Ziv

1 Introduction

Given an extremely long string, it is natural to wonder how compressible it is. This question is fundamental to several disciplines, including information theory, computational complexity theory, machine learning, storage systems, and communications. As massive data sets become commonplace, the ability to estimate compressibility with extremely efficient, even sublinear time, algorithms, is gaining importance. The most general measure of compressibility, Kolmogorov complexity, is not computable (see [30] for a textbook treatment), nor even approximable. Even under restrictions which make it computable (such as a bound on the running time of decompression), it is probably hard to approximate in polynomial time, since an algorithm with non-trivial approximation guarantees would allow one to distinguish random from pseudorandom strings and, hence, invert one-way functions. Nevertheless, the question of how compressible a long string is with respect to a *specific compression scheme* may be tractable, depending on the particular scheme.

We raise the question of approximating the compressibility of a string with respect to a fixed compression scheme, in sublinear time, and give algorithms and nearly matching lower bounds for several versions of the problem. We consider algorithms with *worst-case* approximation guarantees. That is, we do not assume any particular distribution over inputs; instead, our algorithms are randomized and, for every input, produce an output within a specified range with high probability over the coins of the algorithm.

Although our question is new, for one compression scheme, namely Huffman coding (applied to individual symbols as opposed to blocks of symbols), answers follow from previous work. Compressibility under Huffman encoding is determined by the entropy of the symbol frequencies. Given an arbitrary input string w, sampling symbols uniformly with replacement from w provides a sequence of independent observations from a distribution with probabilities given by the symbol frequencies. Approximating the entropy of a distribution based on i.i.d. observations is a well-studied problem[1], and the existing results immediately imply algorithms and lower bounds for sublinear-time algorithms that approximate the compressibility of a string under Huffman encoding.

In this work we study the compressibility approximation question in detail for two popular lossless compression schemes: run-length encoding (RLE) and a variant of Lempel-Ziv (LZ77) [42]. In the RLE scheme, each run, or a sequence of consecutive occurrences of the same character, is stored as a pair: the character, and the length of the run. Run-length encoding is used to compress black and white images, faxes, and other simple graphic images, such as icons and line drawings, which usually

[1] When the sample size is much larger than the alphabet size, then the frequency of each individual symbol (and hence the entropy) can be estimated accurately. When the alphabet is larger than the sample size, then the approximability of the entropy depends on several features of the distribution; see, e.g., Batu *et al.* [4], Cai *et al.* [9], Paninski [33,34], Brautbar and Samorodnitsky [6].

contain many long runs. In the LZ77 scheme, a left-to-right pass of the input string is performed and at each step, the longest sequence of characters that has started in the previous portion of the string is replaced with the pointer to the previous location and the length of the sequence (for a formal definition, see Section 4). The LZ77 scheme and other variants of Lempel-Ziv have been studied extensively in information theory, as well as in machine learning, in part because they compress strings generated by an ergodic source to the shortest possible representation (given by the entropy) in the asymptotic limit (cf. [16]). Many popular archivers, such as gzip, use variations on the Lempel-Ziv scheme. In this work we present sublinear algorithms and corresponding lower bounds for approximating compressibility with respect to both schemes, RLE and LZ77.

Motivation. Computing the compressibility of a long string with respect to specific compression schemes may be done in order to decide whether or not to compress the file, to choose which compression method is the most suitable, or check whether a small modification to the file (e.g., a rotation of an image) will make it significantly more compressible[2]. Moreover, compression schemes are used as tools for measuring properties of strings such as similarity and entropy. As such, they are applied widely in data-mining, natural language processing and genomics (the literature on this topic is too vast to survey here; see, for example, Lowenstern *et al.* [31], Witten *et al.* [41], Frank *et al.* [18], Kukushkina *et al.* [26], Benedetto *et al.* [5], Li *et al.* [29], Calibrasi and Vitányi [12,13], Keogh *et al.* [24], Sculley and Brodley [38], Ferragina *et al.* [17] and the survey of Keogh *et al.* [25]). In these applications, one usually needs only the *length* of the compressed version of a file, not the output itself; a fast and accurate approximation algorithm for compressibility could speed up the computations in these applications significantly.

Roughly, our results show that for RLE one can get a good approximation to the compressibility using very few queries to the input string. In contrast, we show that approximating LZ77 compressibility, even within a constant factor, provably requires reading a much larger fraction of the input string.

Worst-case Multiplicative and Additive Approximation. We consider three approximation notions: additive, multiplicative, and the combination of additive and multiplicative. On inputs of length n, the quantities we approximate range from 1 to n. An *additive approximation* algorithm is allowed an additive error of εn, where $\varepsilon \in (0,1)$ is a parameter. The output of a *multiplicative approximation* algorithm is within a factor $A > 1$ of the correct answer. The combined notion allows both types of error: the algorithm should output an estimate $\widehat{C}$ of the compression cost C such that $\frac{C}{A} - \varepsilon n \leq \widehat{C} \leq A \cdot C + \varepsilon n$.

Our algorithms are randomized and, for every input, the approximation guarantee holds with probability at least $\frac{2}{3}$ over the coins of the algorithm. We stress that we do not make any probabilistic assumptions concerning the way the input string w is generated. Our claims hold for every string w, and the running time of the algo-

[2] For example, a variant of the RLE scheme, typically used to compress images, runs RLE on the concatenated rows of the image and on the concatenated columns of the image, and stores the shorter of the two compressed files.

rithms may depend on the compressibility of w in addition to the given approximation parameters.

We are interested in sublinear approximation algorithms, which read few positions of the input strings. For the schemes we study, purely multiplicative approximation algorithms must (in the worst case) read almost the entire input. Nevertheless, algorithms with additive error guarantees, or a possibility of both multiplicative and additive error are often sufficient for distinguishing very compressible inputs from inputs that are not well compressible. For both the RLE and LZ77 schemes, we give algorithms with combined multiplicative and additive error that make few queries to the input. When it comes to additive approximations, however, the two schemes differ sharply: sublinear additive approximations are possible for the RLE compressibility, but not for LZ77 compressibility.

We summarize our results in Sections 1.1 and 1.2, then interpret them and discuss their implications in Section 1.3. We describe additional related work in Section 1.4 and mention potential research directions in Section 1.5.

1.1 Results for Run-Length Encoding

For RLE, we present sublinear algorithms for all three approximation notions defined above, providing a trade-off between the quality of approximation and the running time. The algorithms with an additive approximation guarantee run in time *independent* of the input size. Specifically, an εn-additive estimate can be obtained in time[3] $\tilde{O}(1/\varepsilon^3)$, and a combined estimate, with a multiplicative error of 3 and an additive error of εn, can be obtained in time $\tilde{O}(1/\varepsilon)$. As for a strict multiplicative approximation, we give a simple 4-multiplicative approximation algorithm that runs in expected time $\tilde{O}(\frac{n}{C_{\mathrm{RLE}}(w)})$, where $C_{\mathrm{RLE}}(w)$ denotes the compression cost of the string w, that is, the number of symbols w compresses to under RLE. For any $\gamma > 0$, the multiplicative error can be improved to $1+\gamma$ at the cost of multiplying the running time by $\mathrm{poly}(1/\gamma)$. Observe that the algorithm is more efficient when the string is less compressible, and less efficient when the string is more compressible. One of our lower bounds justifies such a behavior and, in particular, shows that a constant factor approximation requires linear time for strings that are very compressible. We also give a lower bound of $\Omega(1/\varepsilon^2)$ for εn-additive approximation.

1.2 Results for Lempel-Ziv

We prove that approximating compressibility with respect to LZ77 is related to its ℓth *subword complexity* (that is, the number of distinct substrings of length ℓ that it contains) for *small* ℓ. In turn, this problem reduces to the following problem, which we call DISTINCT ELEMENTS (DE):

[3] The notation $\tilde{O}(g(k))$ for a function g of a parameter k means $O(g(k) \cdot \mathrm{polylog}(g(k))$ where $\mathrm{polylog}(g(k)) = \log^c(g(k))$ for some constant c.

Definition 1 (DE **Problem**) *Given access to a string τ over alphabet Ψ, approximate the number of* distinct *elements (that is, symbols) in τ.*

This is essentially equivalent to estimating the support size of a distribution [37]. Variants of this problem have been considered under various guises in the literature: in databases it is referred to as approximating distinct values (Charikar *et al.* [10]), in statistics as estimating the number of species in a population (see the over 800 references maintained by Bunge [7]), and in streaming as approximating the frequency moment F_0 (Alon *et al.* [2], Bar-Yossef *et al.* [3]). Most of these works, however, consider models different from ours. For our model, there is an A-multiplicative approximation algorithm of [10], that runs in time $O\left(\frac{n}{A^2}\right)$, matching the lower bound in [10,3]. There is also an almost linear lower bound for approximating DE with additive error [37].

We give a reduction from LZ77 compressibility to DE and vice versa. These reductions allow us to employ the known results on DE to give algorithms and lower bounds for this problem. Our approximation algorithm for LZ77 compressibility combines a multiplicative and additive error. The running time of the algorithm is $\tilde{O}\left(\frac{n}{A^3\varepsilon}\right)$ where A is the multiplicative error and εn is the additive error. In particular, this implies that for any $\alpha > 0$, we can distinguish, in sublinear time $\tilde{O}(n^{1-\alpha})$, strings compressible to $O(n^{1-\alpha})$ symbols from strings only compressible to $\Omega(n)$ symbols.[4]

The main tool in the algorithm consists of two combinatorial structural lemmas that relate compressibility of a string to its ℓth subword complexity for small ℓ, that is, the number of distinct substrings of length ℓ that it contains (when considering all $n-\ell+1$ possible overlapping substrings). Roughly, the lemmas say that a string is well compressible with respect to LZ77 if and only if its ℓth subword complexity is small for all small ℓ. The simpler of the two lemmas was inspired by a structural lemma for grammars by Lehman and Shelat [27]. The combinatorial lemmas allow us to establish a reduction from LZ77 compressibility to DE and employ a (simple) algorithm for approximating DE in our algorithm for LZ77.

Interestingly, we can show that there is also a reduction in the *opposite direction*: namely, approximating DE reduces to approximating LZ77 compressibility. The lower bound of [37], combined with the reduction from DE to LZ77, implies that our algorithm for LZ77 cannot be improved significantly. In particular, our lower bound implies that for any $B = n^{o(1)}$, distinguishing strings compressible by LZ77 to $\tilde{O}(n/B)$ symbols from strings compressible to $\tilde{\Omega}(n)$ symbols requires $n^{1-o(1)}$ queries.

1.3 Discussion of the Results

We stress again that our results are worst-case over inputs; see the discussion under "Worst-case Multiplicative and Additive Approximation", above. Nevertheless, it is natural to ask what our results imply for typical inputs. For many of the sources studied in the information theory literature, the compressibility of typical inputs of length n scales as Hn, where H is a constant depending on the source. Thus, we might

[4] To see this, set $A = o(n^{\alpha/2})$ and $\varepsilon = o(n^{-\alpha/2})$.

ask how well our algorithms approximate the constant H on such inputs. The answers are drastically different for the two compression schemes that we study.

Our positive results for RLE are stronger than our positive result for LZ77. As noted previously, if we are interested in a $(1+\gamma)$-factor approximation for RLE, for any $\gamma > 0$, then the complexity of the algorithm is $\text{poly}(1/\gamma)$ when $C_{\text{RLE}}(w) = \Omega(n)$, and in general it depends (roughly) linearly on $n/C_{\text{RLE}}(w)$. Thus, we can obtain a very precise estimate of $C_{\text{RLE}}(w)$ in sublinear time as long as $C_{\text{RLE}}(w)$ is not negligible compared to n. Stating the problem slightly differently, that is, as a decision problem (and using our first, purely additive approximation algorithm), for any δ (that may be a constant, or a function of n) and $\varepsilon < \delta$, we can distinguish between the case that $C_{\text{RLE}}(w) \geq \delta n$ and the case that $C_{\text{RLE}}(w) < (\delta - \varepsilon)n$ in time $\tilde{O}(1/\varepsilon^3)$. Alternatively (using our second result), we can distinguish between the case that $C_{\text{RLE}}(w) \geq \delta n$ and the case that $C_{\text{RLE}}(w) < (\delta/9 - (2/3)\varepsilon)n$ in time $\tilde{O}(1/\varepsilon)$.

In contrast, as stated in Section 1.2, for LZ77, such precise estimates cannot be obtained in sublinear time. One can view our results for LZ77 (unlike for RLE) as being mainly negative. Indeed, we establish the limitations of any algorithm that approximates $C_{\text{LZ77}}(w)$, and in particular we show that, for constant δ, no algorithm can distinguish between the case that $C_{\text{LZ77}}(w) \geq \delta n$ and the case that $C_{\text{LZ77}}(w) < (\delta/B)n$ in time $\tilde{O}(n/B)$. ($C_{\text{LZ77}}(w)$ denotes the number of symbols w compresses to under LZ77.) Thus, when dealing only with highly incompressible strings (e.g., typical sequences from a memoryless source), we do not, and cannot, get a sublinear algorithm. However, our algorithm can be useful in scenarios where some strings are highly compressible, and we want to detect this (with high probability) without reading the whole input and running the compression algorithm.

1.4 Additional Related Work

As noted previously, at the core of our result for LZ77 are two combinatorial structural lemmas that relate compressibility of the string to its ℓth-subword complexity (for small ℓ). Both the total subword complexity (the total number of distinct substrings in a string) and the ℓ-subword complexity (the number of substrings of length ℓ) have been studied extensively in the past.

Much of the existing work focuses on understanding how the subword complexities of a string w behave when w is chosen randomly according to various memoryless and stationary sources (Janson *et al.* [21], Gheorghiciuc and Ward [19], Kása [22] and Léve and Séébold [28]). Those results are not directly relevant to our work, given our focus on worst-case analysis.

Combinatorial results are fewer, and focus mostly on the total subword complexity. Shallit [39] and de Luca [32] study the maximum possible subword complexity of strings, and Gheorghiciuc and Ward [19] show relationships between the ℓ-subword complexity and the total subword complexity. The most similar in flavor to our work is that of Ilie *et al.* [20]. They relate the Lempel-Ziv compressibility to the total subword complexity for extremely compressible strings: specifically, they show that the total subword complexity of an infinite string's prefixes scales linearly if and only if the prefixes compress to a constant number of symbols under LZ77 (these two condi-

tions are equivalent to the string being periodic). It is not clear what their techniques imply for strings with superconstant compressibility.

Finally, we note that approximation algorithms for compressibility and "generalized" compressibility (in which one looks at how compressible a string x is given another string y as a reference) have been considered before (e.g., Cormode and Muthukrishnan [15], Keller *et al.* [23]). However, those algorithms process (or preprocess) the entire string, and thus run in at least linear time overall; in contrast, our goal is to understand when sublinear algorithms are possible.

1.5 Further Research

It would be interesting to extend our results for estimating the compressibility under LZ77 to other variants of Lempel-Ziv, such as dictionary-based LZ78 [43]. Compressibility under LZ78 can be drastically different from compressibility under LZ77: e.g., for 0^n they differ roughly by a factor of $\sqrt{n}$. (Other, less degenerate examples for which there is a gap appear in [35].) Another open question is approximating compressibility for schemes other than RLE and Lempel-Ziv, e.g., based on the Burrows-Wheeler transform (BTW) [8], prediction by partial matching (PPM) [14] and the context tree weighting method (CTW) [40]. It would also be interesting to design approximation algorithms for lossy compression schemes, e.g., schemes based on a discrete cosine transform [1], such as JPEG, MPEG and MP3, and schemes based on wavelets [11], such as JPEG-2000. One lossy scheme to which our results extend directly is a commonly used variant of RLE, where some distinct symbols, e.g., pixels of similar color, are treated as the same character.

1.6 Organization

We start with establishing common notation and defining our notions of approximation in Section 2. Section 3 presents algorithms and lower bounds for RLE. The algorithmic results are summarized in Theorem 1 and the lower bounds, in Theorem 2. Section 4 deals with the LZ77 scheme: it starts with the structural lemmas, explains the approximation algorithm for compressibility with respect to LZ77 and finishes with the reduction from DE to LZ77 compressibility. Subsection 4.3 describes a simple algorithm for DE.

2 Preliminaries

Our algorithms are given query access to a string w of length n over a finite alphabet Σ. That is, they may ask what is w_t for any $t \in [n]$ of their choice (where $[n] \stackrel{\text{def}}{=} \{1,\ldots,n\}$). Let $C(w)$ denote the length of the compressed version of w according to some fixed compression scheme. We consider estimates to $C(w)$ that have both multiplicative and additive error. We call $\widehat{C}$ an (A,ε)-*estimate* for $C(w)$ if

$$\frac{C(w)}{A} - \varepsilon n \;\le\; \widehat{C} \;\le\; A \cdot C(w) + \varepsilon n\,,$$

and say an algorithm (A,ε)-*estimates* C (or is an (A,ε)-*approximation algorithm* for C) if, *for each input* w, it produces an (A,ε)-estimate for $C(w)$ with probability at least $\frac{2}{3}$ over the coins of the algorithm.

When the error is purely additive or multiplicative, we use the following shorthand: εn-*additive estimate* stands for $(1,\varepsilon)$-*estimate* and A-*multiplicative estimate*, or A-*estimate*, stands for $(A,0)$-*estimate*. An algorithm computing an εn-additive estimate with probability at least $\frac{2}{3}$ is an εn-*additive approximation algorithm*, and if it computes an A-multiplicative estimate then it is an A-*multiplicative approximation algorithm*, or A-*approximation algorithm*.

For some settings of parameters, obtaining a valid estimate is trivial. For a quantity in $[1,n]$, for example, $\frac{n}{2}$ is an $\frac{n}{2}$-additive estimate, $\sqrt{n}$ is a $\sqrt{n}$-estimate and εn is an (A,ε)-estimate whenever $A \geq \frac{1}{2\varepsilon}$.

When measuring running time, we use a random access memory model: we charge one time unit for every symbol of the input which is read, regardless of its position in the input string.

3 Run-Length Encoding

Every n-character string w over alphabet Σ can be partitioned into maximal runs of identical characters of the form σ^ℓ, where σ is a symbol in Σ and ℓ is the length of the run, and consecutive runs are composed of different symbols. In the *Run-Length Encoding* of w, each such run is replaced by the pair (σ,ℓ). The number of bits needed to represent such a pair is $\lceil\log(\ell+1)\rceil + \lceil\log|\Sigma|\rceil$ plus the overhead which depends on how the separation between the characters and the lengths is implemented. One way to implement it is to use prefix-free encoding for lengths. For simplicity we ignore the overhead in the above expression, but our analysis can be adapted to any implementation choice. The *cost of the run-length encoding*, denoted by $C_{\mathrm{RLE}}(w)$, is the sum over all runs of $\lceil\log(\ell+1)\rceil + \lceil\log|\Sigma|\rceil$.

We assume that the alphabet Σ has constant size. This is a natural assumption when using run-length encoding, but the analysis of our algorithms can be extended in a straightforward manner to alphabets whose size is a function of n. The complexity of the algorithms will grow polylogarithmically with $|\Sigma|$.

We first present an algorithm that, given a parameter ε, outputs an εn-additive estimate to $C_{\mathrm{RLE}}(w)$ with high probability and makes $\tilde{O}(1/\varepsilon^3)$ queries. We then reduce the query complexity to $\tilde{O}(1/\varepsilon)$ at the cost of incurring a multiplicative approximation error in addition to the additive one: the new algorithm $(3,\varepsilon)$-estimates $C_{\mathrm{RLE}}(w)$. We later discuss how to use approximation schemes with multiplicative and additive error to get a purely multiplicative approximation, at a cost on the query complexity that depends on $n/C_{\mathrm{RLE}}(w)$. That is, the more compressible the string w is, the higher the query complexity of the algorithm. These results are summarized in Theorem 1, stated next. The algorithms referred to by the theorem are presented in Subsections 3.1–3.3.

Theorem 1 *Let $w \in \Sigma^n$ be a string to which we are given query access.*

1. *Algorithm I gives an εn-additive approximation to $C_{RLE}(w)$ in time $\tilde{O}(1/\varepsilon^3)$.*

2. *Algorithm II* $(3,\varepsilon)$*-estimates* $C_{RLE}(w)$ *in time* $\tilde{O}(1/\varepsilon)$.
3. *Algorithm III* 4*-estimates* $C_{RLE}(w)$ *and runs in expected time* $\tilde{O}\left(\frac{n}{C_{RLE}(w)}\right)$. *Moreover, a slight modification of Algorithm III* $(1+\gamma)$*-estimates* $C_{RLE}(w)$ *in expected time* $\tilde{O}\left(\frac{n}{C_{RLE}(w)} \cdot \mathrm{poly}(1/\gamma)\right)$.

We note that though the (expected) running time of Algorithm III depends on $C_{\mathrm{RLE}}(w)$, the algorithm needs no prior knowledge of $C_{\mathrm{RLE}}(w)$. The same is true of the variant of the algorithm that obtains a $(1+\gamma)$-estimation.

We also give two lower bounds, for multiplicative and additive approximation, respectively, which establish that the running times in Items 1 and 3 of Theorem 1 are essentially tight.

Theorem 2

1. *For all* $A > 1$, *any* A*-approximation algorithm for* C_{RLE} *requires* $\Omega\left(\frac{n}{A^2 \log n}\right)$ *queries. Furthermore, if the input is restricted to strings with compression cost* $C_{RLE}(w) \geq C$, *then* $\Omega\left(\frac{n}{CA^2 \log(n)}\right)$ *queries are necessary.*
2. *For all* $\varepsilon \in \left(0, \frac{1}{2}\right)$, *any* εn*-additive approximation algorithm for* C_{RLE} *requires* $\Omega(1/\varepsilon^2)$ *queries.*

In the next subsections we prove Theorems 1 and 2.

3.1 An εn-Additive Estimate with $\tilde{O}(1/\varepsilon^3)$ Queries

Our first algorithm for approximating the cost of RLE is very simple: it samples a few positions in the input string uniformly at random and bounds the lengths of the runs to which they belong by looking at the positions to the left and to the right of each sample. If the corresponding run is short, its length is established exactly; if it is long, we argue that it does not contribute much to the encoding cost. For each index $t \in [n]$, let $\ell(t)$ be the length of the run to which w_t belongs. The cost contribution of index t is defined as

$$c(t) = \frac{\lceil \log(\ell(t)+1) \rceil + \lceil \log |\Sigma| \rceil}{\ell(t)} .$$

By definition, $C_{\mathrm{RLE}}(w)/n = \mathsf{E}_{t\in[n]}[c(t)]$, where $\mathsf{E}_{t\in[n]}$ denotes expectation over a uniformly random choice of t. The algorithm, presented below, estimates the encoding cost by the average of the cost contributions of the sampled short runs, multiplied by n.

ALGORITHM I: AN εn-ADDITIVE APPROXIMATION FOR $C_{\mathrm{RLE}}(w)$

1. Select $q = \Theta\left(\frac{1}{\varepsilon^2}\right)$ indices $t_1, \ldots, t_q$ uniformly and independently at random.
2. For each $i \in [q]$:
   (a) Query t_i and up to $\ell_0 = \frac{8\log(4|\Sigma|/\varepsilon)}{\varepsilon}$ positions in its vicinity to bound $\ell(t_i)$.
   (b) Set $\hat{c}(t_i) = c(t_i)$ if $\ell(t_i) < \ell_0$ and $\hat{c}(t_i) = 0$ otherwise.
3. Output $\widehat{C}_{\mathrm{RLE}} = n \cdot \mathop{\mathsf{E}}_{i\in[q]}[\hat{c}(t_i)]$.

Proof of Theorem 1, Item 1. We first prove that the algorithm is an εn-additive approximation algorithm. The error of the algorithm comes from two sources: from ignoring the contribution of long runs and from sampling. The ignored indices t, for which $\ell(t) \geq \ell_0$, do not contribute much to the cost. Since the cost assigned to the indices monotonically decreases with the length of the run to which they belong, for each such index,

$$c(t) \leq \frac{\lceil \log(\ell_0 + 1) \rceil + \lceil \log |\Sigma| \rceil}{\ell_0} \leq \frac{\varepsilon}{2}\,.$$

Therefore,

$$\frac{C_{\mathrm{RLE}}(w)}{n} - \frac{\varepsilon}{2} \;\leq\; \frac{1}{n} \cdot \sum_{t:\ell(t)<\ell_0} c(t) \;\leq\; \frac{C_{\mathrm{RLE}}(w)}{n}.$$

Equivalently, $\frac{C_{\mathrm{RLE}}(w)}{n} - \frac{\varepsilon}{2} \leq \mathsf{E}_{i\in[n]}[\hat{c}(t_i)] \leq \frac{C_{\mathrm{RLE}}(w)}{n}$.

By an additive Chernoff bound, with high constant probability, the sampling error in estimating $\mathsf{E}[\hat{c}(t_i)]$ is at most $\varepsilon/2$. Therefore, $\widehat{C}_{\mathrm{RLE}}$ is an εn-additive estimate of $C_{\mathrm{RLE}}(w)$, as desired.

We now turn to the query complexity and running time, where recall that we assume that $|\Sigma|$ is constant. Since the number of queries performed for each selected t_i is $O(\ell_0) = O(\log(1/\varepsilon)/\varepsilon)$, the total number of queries, as well as the running time, is $O(\log(1/\varepsilon)/\varepsilon^3)$. □

3.2 A $(3,\varepsilon)$-Estimate with $\tilde{O}(1/\varepsilon)$ Queries

If we are willing to allow a constant multiplicative approximation error in addition to εn-additive, we can reduce the query and time complexity to $\tilde{O}(1/\varepsilon)$. The idea is to partition the positions in the string into *buckets* according to the length of the runs they belong to. Each bucket corresponds to runs of the same length up to a small constant factor. For the sake of brevity of the analysis, we take this constant to be 2. A smaller constant results in a better multiplicative factor. Given the definition of the buckets, for every two positions t_1 and t_2 from the same bucket, $c(t_1)$ and $c(t_2)$ differ by at most a factor of 2. Hence, good estimates of the sizes of all buckets would yield a good estimate of the total cost of the run-length encoding.

The algorithm and its analysis build on two additional observations: (1) Since the cost, $c(t)$, monotonically decreases with the length of the run to which t belongs, we can allow a less precise approximation of the size of the buckets that correspond to

longer runs. (2) A bucket containing relatively few positions contributes little to the run-length encoding cost. Details follow.

ALGORITHM II: A $(3,\varepsilon)$-APPROXIMATION FOR $C_{\mathrm{RLE}}(w)$

1. Select $q = \Theta\left(\frac{\log(1/\varepsilon)\cdot\log\log(1/\varepsilon)}{\varepsilon}\right)$ indices $t_1,\ldots,t_q$ uniformly and independently at random.
2. For $h = 1,\ldots,h_0 = \lceil \log \ell_0 \rceil$ where $\ell_0 = \frac{8\log(4|\Sigma|/\varepsilon)}{\varepsilon}$ (as defined in Algorithm I), do
   (a) Consider the first $q_h = \min\left\{q, q\cdot\frac{h+s}{2^{h-1}}\right\}$ indices $t_1,\ldots,t_{q_h}$.
   (b) For each $i = 1,\ldots,q_h$, set $X_{h,i} = 1$ if $t_i \in B_h$ and set $X_{h,i} = 0$ otherwise.
3. Output $\widehat{C}_{\mathrm{RLE}} = \sum_{h=1}^{h_0}\left(\frac{n}{q_h}\cdot\sum_{i=1}^{q_h} X_{h,i}\right)\cdot\frac{h+s}{2^{h-1}}$.

Proof of Theorem 1, Item 2. Observe that by the definition of h_0 and ℓ_0, we have that $h_0 = O(\log(1/\varepsilon))$. For each $h \in [h_0]$, let $B_h = \{t : 2^{h-1} \le \ell(t) < 2^h\}$. That is, the bucket B_h contains all indices t that belong to runs of length approximately 2^h. Let $s \stackrel{\mathrm{def}}{=} \lceil \log|\Sigma| \rceil$ and

$$C_{\mathrm{RLE}}(w,h) \stackrel{\mathrm{def}}{=} \sum_{t\in B_h} c(t)\ .$$

Then

$$|B_h|\cdot\frac{h+s}{2^h} \ \le\ C_{\mathrm{RLE}}(w,h) \ \le\ |B_h|\cdot\frac{h+s}{2^{h-1}}\ ,$$

which implies that

$$C_{\mathrm{RLE}}(w,h) \ \le\ |B_h|\cdot\frac{h+s}{2^{h-1}} \ \le\ 2\cdot C_{\mathrm{RLE}}(w,h)\ . \tag{1}$$

Our goal is to obtain (with high probability), for every h, a relatively accurate estimate β_h of $\frac{|B_h|}{n}$. Specifically, let

$$H_{\mathrm{big}} = \left\{h\ :\ \frac{|B_h|}{n} \ \ge\ \frac{1}{2}\cdot\frac{\varepsilon}{h_0}\cdot\frac{2^{h-1}}{h+s}\right\} \quad\text{and}$$
$$H_{\mathrm{small}} = \left\{h\ :\ \frac{|B_h|}{n} \ <\ \frac{1}{2}\cdot\frac{\varepsilon}{h_0}\cdot\frac{2^{h-1}}{h+s}\right\}\ .$$

Then we would like β_h to satisfy the following:

$$\begin{aligned}\frac{1}{3}\cdot\frac{|B_h|}{n} \le\ \beta_h \le \tfrac{3}{2}\cdot\tfrac{|B_h|}{n} \quad &\text{if } h\in H_{\mathrm{big}};\\ 0 \le \beta_h \le \tfrac{\varepsilon}{h_0}\cdot\tfrac{2^{h-1}}{h+s} \quad &\text{otherwise } (h\in H_{\mathrm{small}})\ .\end{aligned} \tag{2}$$

Given such estimates $\beta_1, \ldots, \beta_{h_0}$, approximate the encoding cost by $\widehat{C}_{\mathrm{RLE}} = \sum_{h=1}^{h_0} \beta_h \cdot n \cdot \frac{h+s}{2^{h-1}}$. Then

$$\begin{aligned}
\widehat{C}_{\mathrm{RLE}} &= \sum_{h \in H_{\mathrm{big}}} \beta_h \cdot n \cdot \frac{h+s}{2^{h-1}} + \sum_{h \in H_{\mathrm{small}}} \beta_h \cdot n \cdot \frac{h+s}{2^{h-1}} \\
&\leq \sum_{h \in H_{\mathrm{big}}} \frac{3}{2} \cdot |B_h| \cdot \frac{h+s}{2^{h-1}} + h_0 \cdot \frac{\varepsilon}{h_0} \cdot \frac{2^{h-1}}{h+s} \cdot n \cdot \frac{h+s}{2^{h-1}} \\
&\leq \sum_{h \in H_{\mathrm{big}}} 3 \cdot C_{\mathrm{RLE}}(w,h) + \varepsilon n \;\; < \;\; 3 \cdot C_{\mathrm{RLE}}(w) + \varepsilon n \, .
\end{aligned}$$

The last inequality uses the upper bound from (1). Similarly,

$$\begin{aligned}
\widehat{C}_{\mathrm{RLE}} &\geq \sum_{h \in H_{\mathrm{big}}} \beta_h \cdot n \cdot \frac{h+s}{2^{h-1}} \\
&\geq \frac{1}{3} \cdot \sum_{h \in H_{\mathrm{big}}} C_{\mathrm{RLE}}(w,h) \\
&= \frac{1}{3} \cdot \left(C_{\mathrm{RLE}}(w) - \sum_{h \in H_{\mathrm{small}}} C_{\mathrm{RLE}}(w,h) \right) \\
&> \frac{1}{3} \cdot C_{\mathrm{RLE}}(w) - \varepsilon n \, .
\end{aligned}$$

Let β_h be a random variable equal to $\frac{1}{q_h} \sum_{i=1}^{q_h} X_{h,i}$. We show that with high probability, β_h satisfies (2) for every $h \in [h_0]$. For each fixed h we have that $\Pr[X_{h,i} = 1] = \frac{|B_h|}{n}$ for every $i \in [q_h]$. Hence, by a multiplicative Chernoff bound,

$$\Pr\left[\left| \beta_h - \frac{|B_h|}{n} \right| \geq \frac{1}{2} \frac{|B_h|}{n} \right] < \exp\left(-c \cdot \frac{|B_h|}{n} \cdot q_h \right) \tag{3}$$

for some constant $c \in (0,1)$. Recall that $h_0 = O(\log(1/\varepsilon))$ and that $q_h = \Theta(q \cdot \frac{h+s}{2^{h-1}}) = \Omega\left(\varepsilon^{-1} \cdot h_0 \cdot \log(h_0) \cdot \frac{h+s}{2^{h-1}}\right)$. Hence, for $h \in H_{\mathrm{big}}$ (and for a sufficiently large constant in the $\Theta(\cdot)$ notation in the definition of q), the probability in (3) is at most $\frac{1}{3} \cdot \frac{1}{h_0}$, and so (2) holds with probability at least $1 - \frac{1}{3} \cdot \frac{1}{h_0}$. On the other hand, for $h \in H_{\mathrm{small}}$, the probability that $\beta_h \geq \frac{\varepsilon}{h_0} \cdot \frac{2^{h-1}}{h+s}$ is bounded above by the probability of this event when $\frac{|B_h|}{n} = \frac{1}{2} \cdot \frac{\varepsilon}{h_0} \cdot \frac{2^{h-1}}{h+s}$. By (3), this is at most $\frac{1}{3} \cdot \frac{1}{h_0}$, and so in this case too (2) holds with probability at least $1 - \frac{1}{3} \cdot \frac{1}{h_0}$. Taking a union bound over all $h \in [h_0]$ completes the analysis.

We now turn to the query complexity and running time. For a given index t_i, deciding whether $t_i \in B_h$ requires $O(2^h)$ queries. (More precisely, we need at most 2^{h-1} queries in addition to the queries from the previous iterations.) Hence, the total number of queries is

$$O\left(\sum_{h=1}^{h_0} q_h \cdot 2^h \right) = O\left(q \cdot h_0^2 \right) = O\left(\frac{\log^3(1/\varepsilon) \cdot \log\log(1/\varepsilon)}{\varepsilon} \right) \, .$$

□

3.3 A 4-Multiplicative Estimate with $\tilde{O}(n/C_{\mathrm{RLE}}(w))$ Queries

In this subsection we "get-rid" of the εn additive error by introducing a dependence on the run-length encoding cost (which is of course unknown to the algorithm). First, assume a lower bound $C_{\mathrm{RLE}}(w) \geq \mu n$ for some $\mu > 0$. Then, by running Algorithm II (the $(3,\varepsilon)$-approximation algorithm) with ε set to $\mu/2$, and outputting $\widehat{C}_{\mathrm{RLE}} + \varepsilon n$, we get a 4-multiplicative estimate with $\tilde{O}(1/\mu)$ queries.

We can search for such a lower bound μn, as follows. Suppose that Algorithm II receives, in addition to the additive approximation parameter ε, a confidence parameter δ, and outputs a $(3,\varepsilon)$-estimate with probability at least $1-\delta$ instead of $2/3$. This can easily be achieved by increasing the query complexity of the algorithm by a factor of $\log(1/\delta)$. By performing calls to Algorithm II with decreasing values of ε and δ, we can maintain a sequence of intervals of decreasing size, that contain $C_{\mathrm{RLE}}(w)$ (with high probability). Once the ratio between the extreme points of the interval is sufficiently small, the algorithm terminates. Details follow.

ALGORITHM III: A 4-APPROXIMATION FOR $C_{\mathrm{RLE}}(w)$

1. Set $j = 0$, $lb_0 = 0$ and $ub_0 = 1$.
2. While $\frac{ub_j}{lb_j} > 16$ do:
   (a) $j = j+1$, $\varepsilon_j = 2^{-j}$, $\delta_j = \frac{1}{3}\cdot 2^{-j}$.
   (b) Call Algorithm II with $\varepsilon = \varepsilon_j$ and $\delta = \delta_j$, and let $\widehat{C}^j_{\mathrm{RLE}}$ be its output.
   (c) Let $ub_j = 3(\widehat{C}^j_{\mathrm{RLE}} + \varepsilon_j n)$ and $lb_j = \max\left\{0, \frac{1}{3}(\widehat{C}^j_{\mathrm{RLE}} - \varepsilon_j n)\right\}$.
3. Output $\sqrt{lb_j \cdot ub_j}$.

Proof of Theorem 1, Item 3. For any given j, Algorithm II outputs $\widehat{C}^j_{\mathrm{RLE}} \in [\frac{1}{3}C_{\mathrm{RLE}}(w) - \varepsilon_j n,\ 3C_{\mathrm{RLE}}(w) + \varepsilon_j n]$, with probability at least $1 - \frac{1}{3}\cdot\delta_j$. Equivalently, $lb_j \leq C_{\mathrm{RLE}}(w) \leq ub_j$. By the union bound, with probability at least $2/3$, $lb_j \leq C_{\mathrm{RLE}}(w) \leq ub_j$ for all j. Assume this event in fact holds. Then, upon termination (when $ub_j/lb_j \leq 16$), the output is a 4-multiplicative estimate of $C_{\mathrm{RLE}}(w)$. It is not hard to verify that once $\varepsilon_j \leq \frac{C_{\mathrm{RLE}}(w)}{24n}$, then the algorithm indeed terminates with probability at least $1-\delta_j$.

The query complexity of the algorithm is dominated by its last iteration. As stated above, for each $\varepsilon_j \leq \frac{C_{\mathrm{RLE}}(w)}{24n}$, conditioned on the algorithm not terminating in iteration $j-1$, the probability that it does not terminate in iteration j is at most $\delta_j = \frac{1}{3}2^{-j}$. Since the query complexity of Algorithm II is $\tilde{O}(1/\varepsilon)$, the expected query complexity of Algorithm III is $\tilde{O}(n/C_{\mathrm{RLE}}(w))$. □

Improving the multiplicative approximation factor. The 4-multiplicative estimate of $C_{\mathrm{RLE}}(w)$ can be improved to a $(1+\gamma)$-multiplicative estimate for any $\gamma > 0$. This is done by refining the buckets defined in Subsection 3.2 so that $B_h = \{t : (1+\frac{\gamma}{2})^{h-1} \leq \ell(t) < (1+\frac{\gamma}{2})^h\}$ for $h = 1,\ldots,\log_{1+\frac{\gamma}{2}}\ell_0$ (=$O(\log(1/\varepsilon)/\gamma)$), and setting $\varepsilon = \gamma\cdot\mu/8$. The query complexity remains linear in $1/\mu = n/C_{\mathrm{RLE}}(w)$ (up to polylogarithmic factors), and is polynomial in $1/\gamma$.

3.4 A Multiplicative Lower Bound

The proof of Theorem 2, Item 1, follows from the next lemma, where we set $k = C$ and $k' = A^2 C \log n$.

Lemma 1 *For every $n \geq 2$ and every integer $1 \leq k \leq n/2$, there exists a family of strings, denoted W_k, for which the following holds: (1) $C_{RLE}(w) = \Theta\left(k \log(\frac{n}{k})\right)$ for every $w \in W_k$; (2) Distinguishing a uniformly random string in W_k from one in $W_{k'}$, where $k' > k$, requires $\Omega\left(\frac{n}{k'}\right)$ queries.*

Proof: Let $\Sigma = \{0,1\}$ and assume for simplicity that n is divisible by k. Every string in W_k consists of k blocks, each of length $\frac{n}{k}$. Every odd block contains only 1s and every even block contains a single 0. The strings in W_k differ in the locations of the 0s within the even blocks. Every $w \in W_k$ contains $k/2$ isolated 0s and $k/2$ runs of 1s, each of length $\Theta(\frac{n}{k})$. Therefore, $C_{\mathrm{RLE}}(w) = \Theta\left(k \log(\frac{n}{k})\right)$. To distinguish a random string in W_k from one in $W_{k'}$ with probability 2/3, one must make $\Omega(\frac{n}{\max(k,k')})$ queries since, in both cases, with asymptotically fewer queries the algorithm sees only 1's with high probability. □

3.5 An Additive Lower Bound

Proof of Theorem 2, Item 2. For any $p \in [0,1]$ and sufficiently large n, let $\mathscr{D}_{n,p}$ be the following distribution over n-bit strings. For simplicity, consider n divisible by 3. The string is determined by $\frac{n}{3}$ independent coin flips, each with bias p. Each "heads" extends the string by three runs of length 1, and each "tails", by a run of length 3. Given the sequence of run lengths, dictated by the coin flips, we output the unique binary string that starts with 0 and has this sequence of run lengths.[5]

Let W be a random variable drawn according to $\mathscr{D}_{n,1/2}$ and W', according to $\mathscr{D}_{n,1/2+\varepsilon}$. It is well known that $\Omega(1/\varepsilon^2)$ independent coin flips are necessary to distinguish a coin with bias $\frac{1}{2}$ from a coin with bias $\frac{1}{2}+\varepsilon$. Therefore, $\Omega(1/\varepsilon^2)$ queries are necessary to distinguish w from w'.

We next show that with very high probability the encoding costs of w and w' differ by $\Omega(\varepsilon n)$. Runs of length 1 contribute 1 to the encoding cost, and runs of length 3 cost $\lceil \log(3+1) \rceil = 2$. Therefore, each "heads" contributes $3 \cdot 1$, while each "tails" contributes 2. Hence, if we get $\alpha \cdot \frac{n}{3}$ "heads", then the encoding cost of the resulting string is $\frac{n}{3} \cdot (3\alpha + 2(1-\alpha)) = \frac{n}{3} \cdot (2+\alpha)$. The expected value of α is p. By an additive Chernoff bound, $|\alpha - p| \leq \varepsilon/4$ with probability at least $1 - 2\exp(-2(\varepsilon/4)^2)$. With this probability, the encoding cost of the selected string is between $\frac{n}{3} \cdot \left(2 + p - \frac{\varepsilon}{4}\right)$ and $\frac{n}{3} \cdot \left(2 + p + \frac{\varepsilon}{4}\right)$. The theorem (for the case $n \mod 3 = 0$) follows, since with very high probability, $C_{\mathrm{RLE}}(w') - C_{\mathrm{RLE}}(w) = \Omega(\varepsilon n)$.

If $n \mod 3 = b$ for some $b > 0$ then we make the following minor changes in the construction and the analysis: (1) The first b bits in the string are always set to 0. (2) This adds b to the encoding cost. (3) Every appearance of $\frac{n}{3}$ in the proof is replaced by $\left\lfloor \frac{n}{3} \right\rfloor$. It is easy to verify that the lower bound holds for any sufficiently large n. □

[5] Let b_i be a boolean variable representing the outcome of the ith coin. Then the output is $0 b_1 01 \overline{b_2} 10 b_3 01 \overline{b_4} 1 \ldots$

4 Lempel Ziv Compression

In this section we consider a variant of Lempel and Ziv's compression algorithm [42], which we refer to as LZ77. In all that follows we use the shorthand $[n]$ for $\{1,\ldots,n\}$. Let $w \in \Sigma^n$ be a string over an alphabet Σ. Each symbol of the compressed representation of w, denoted $LZ77(w)$, is either a character $\sigma \in \Sigma$ or a pair (p,ℓ) where $p \in [n]$ is a pointer (index) to a location in the string w and ℓ is the length of the substring of w that this symbol represents. To compress w, the algorithm works as follows. Starting from $t = 1$, at each step the algorithm finds the longest substring $w_t \ldots w_{t+\ell-1}$ for which there exists an index $p < t$, such that $w_p \ldots w_{p+\ell-1} = w_t \ldots w_{t+\ell-1}$. (The substrings $w_p \ldots w_{p+\ell-1}$ and $w_t \ldots w_{t+\ell-1}$ may overlap.) If there is no such substring (that is, the character w_t has not appeared before) then the next symbol in $LZ77(w)$ is w_t, and $t = t+1$. Otherwise, the next symbol is (p,ℓ) and $t = t+\ell$. We refer to the substring $w_t \ldots w_{t+\ell-1}$ (or w_t when w_t is a new character) as a *phrase*. Clearly, compression takes time $O(n^2)$, and decompression, time $O(n)$.

Let $C_{\mathrm{LZ77}}(w)$ denote the number of symbols in the compressed string $LZ77(w)$. (We do not distinguish between symbols that are characters in Σ, and symbols that are pairs (p,ℓ).) Given query access to a string $w \in \Sigma^n$, we are interested in computing an estimate $\widehat{C}_{\mathrm{LZ77}}$ of $C_{\mathrm{LZ77}}(w)$. As we shall see, this task reduces to estimating the number of distinct substrings in w of different lengths, which in turn reduces to estimating the number of distinct symbols in a string. The actual length of the binary representation of the compressed substring is at most a factor of $2\log n$ larger than $C_{\mathrm{LZ77}}(w)$. This is relatively negligible given the quality of the estimates that we can achieve in sublinear time.

Our results on approximating LZ77 compressibility can be summarized succinctly:

Theorem 3 *For any alphabet Σ:*

1. *Algorithm IV (A,ε)-estimates $C_{LZ77}(w)$ and runs in time $\tilde{O}\left(\frac{n}{A^3\varepsilon}\right)$.*
2. *For any $B = n^{o(1)}$, distinguishing strings with LZ77 compression cost $\tilde{\Omega}(n)$ from strings with cost $\tilde{O}(n/B)$ requires $n^{1-o(1)}$ queries.*

The first bound states that non-trivial approximation guarantees are indeed possible. For example, by setting $A = o(n^{\alpha/2})$ and $\varepsilon = o(n^{-\alpha/2})$, we get an algorithm which distinguishes incompressible strings ($C_{\mathrm{LZ77}} = \Omega(n)$) from partly compressible strings ($C_{\mathrm{LZ77}} = O(n^{\alpha})$) in sublinear time $\tilde{O}(n^{1-\alpha})$. The lower bound states that in some sense this is tight: no approximation algorithm with a purely additive approximation guarantee can run in time which is significantly sublinear.

In the remainder of this section we develop the tools necessary to prove the theorem. We begin by relating LZ77 compressibility to DE (Section 4.1), then use this relation to discuss algorithms (Section 4.2) and lower bounds (Section 4.4) for compressibility.

4.1 Structural Lemmas

Our algorithm for approximating the compressibility of an input string with respect to LZ77 uses an approximation algorithm for DE (defined in the introduction) as a subroutine. The main tool in the reduction from LZ77 to DE is the relation between $C_{\mathrm{LZ77}}(w)$ and the number of distinct substrings in w, formalized in the two structural lemmas. In what follows, $d_\ell(w)$ denotes the *number of distinct substrings* of length ℓ in w. Unlike phrases in w, which are disjoint, these substrings may overlap.

Lemma 2 (Structural Lemma 1) *For every $\ell \in [n]$, $C_{LZ77}(w) \geq \frac{d_\ell(w)}{\ell}$.*

Lemma 3 (Structural Lemma 2) *Let $\ell_0 \in [n]$. Suppose that for some integer m and for every $\ell \in [\ell_0]$, $d_\ell(w) \leq m \cdot \ell$. Then*

$$C_{LZ77}(w) \leq 4(m \log \ell_0 + n/\ell_0).$$

Proof of Lemma 2. This proof is similar to the proof of a related lemma concerning grammars from [27]. First note that the lemma holds for $\ell = 1$, since each character w_t in w that has not appeared previously (that is, $w_{t'} \neq w_t$ for every $t' < t$) is copied by the compression algorithm to $LZ77(w)$.

For the general case, fix $\ell > 1$. Recall that $w_t \dots w_{t+k-1}$ of w is a *phrase* if it is represented by one symbol (p,k) in $LZ77(w)$. Any substring of length ℓ that occurs *within* a phrase must have occurred previously in the string. Such substrings can be ignored for our purposes: the number of *distinct* length-ℓ substrings is bounded above by the number of length-ℓ substrings that start inside one phrase and end in another. Each phrase (except the last) contributes $(\ell - 1)$ such substrings. Therefore, $d_\ell(w) \leq (C_{\mathrm{LZ77}}(w) - 1)(\ell - 1) < C_{\mathrm{LZ77}}(w) \cdot \ell$ for every $\ell > 1$. □

Proof of Lemma 3. Let $n_\ell(w)$ denote the number of phrases of length ℓ in w, not including the last phrase. We use the shorthand n_ℓ for $n_\ell(w)$ and d_ℓ for $d_\ell(w)$. In order to prove the lemma we shall show that for every $1 \leq \ell \leq \lfloor \ell_0/2 \rfloor$,

$$\sum_{k=1}^{\ell} n_k \leq 2(m+1) \cdot \sum_{k=1}^{\ell} \frac{1}{k} \,. \tag{4}$$

For all $\ell \geq 1$, since the phrases in w are disjoint,

$$\sum_{k=\ell+1}^{n} n_k \leq \frac{n}{\ell+1} \,. \tag{5}$$

If we substitute $\ell = \lfloor \ell_0/2 \rfloor$ in (4) and (5), and sum the two inequalities, we get:

$$\sum_{k=1}^{n} n_k \leq 2(m+1) \cdot \sum_{k=1}^{\lfloor \ell_0/2 \rfloor} \frac{1}{k} + \frac{2n}{\ell_0} \leq 2(m+1)(\ln \ell_0 + 1) + \frac{2n}{\ell_0} \,.$$

Since $C_{\mathrm{LZ77}}(w) = \sum_{k=1}^{n} n_k + 1$, the lemma follows.

It remains to prove (4). We do so below by induction on ℓ, using the following claim.

Claim 4 *For every* $1 \leq \ell \leq \lfloor \ell_0/2 \rfloor$, $\sum_{k=1}^{\ell} k \cdot n_k \leq 2\ell(m+1)$.

Proof: We show that each position $j \in \{\ell, \ldots, n-\ell\}$ that participates in a compressed substring of length at most ℓ in w can be mapped to a distinct length-2ℓ substring of w. Since $\ell \leq \ell_0/2$, by the premise of the lemma, there are at most $2\ell \cdot m$ distinct length-2ℓ substrings. In addition, the first $\ell-1$ and the last ℓ positions contribute less than 2ℓ symbols. The claim follows.

We call a substring *new* if no instance of it started in the previous portion of w. Namely, $w_t \ldots w_{t+\ell-1}$ is *new* if there is no $p < t$ such that $w_t \ldots w_{t+\ell-1} = w_p \ldots w_{p+\ell-1}$. Consider a compressed substring $w_t \ldots w_{t+k-1}$ of length $k \leq \ell$. The substrings of length greater than k that start at w_t must be *new*, since LZ77 finds the longest substring that appeared before. Furthermore, every substring that contains such a *new* substring is also *new*. That is, every substring $w_{t'} \ldots w_{t+k'}$ where $t' \leq t$ and $k' \geq k + (t'-t)$, is *new*.

Map each position $j \in \{\ell, \ldots, n-\ell\}$ in the compressed substring $w_t \ldots w_{t+k-1}$ to the length-2ℓ substring that ends at $w_{j+\ell}$. Then each position in $\{\ell, \ldots, n-\ell\}$ that appears in a compressed substring of length at most ℓ is mapped to a distinct length-2ℓ substring, as desired. □ (Claim 4)

Establishing Equation (4). We prove (4) by induction on ℓ. Claim 4 with ℓ set to 1 gives the base case, i.e., $n_1 \leq 2(m+1)$. For the induction step, assume the induction hypothesis for every $j \in [\ell-1]$. To prove it for ℓ, add the equation in Claim 4 to the sum of the induction hypothesis inequalities (in (4)) for every $j \in [\ell-1]$. The left hand side of the resulting inequality is

$$\begin{aligned}
\sum_{k=1}^{\ell} k \cdot n_k + \sum_{j=1}^{\ell-1} \sum_{k=1}^{j} n_k &= \sum_{k=1}^{\ell} k \cdot n_k + \sum_{k=1}^{\ell-1} \sum_{j=1}^{\ell-k} n_k \\
&= \sum_{k=1}^{\ell} k \cdot n_k + \sum_{k=1}^{\ell-1} (\ell-k) \cdot n_k \\
&= \ell \cdot \sum_{k=1}^{\ell} n_k \,.
\end{aligned}$$

The right hand side, divided by the factor $2(m+1)$, which is common to all inequalities, is

$$\begin{aligned}
\ell + \sum_{j=1}^{\ell-1} \sum_{k=1}^{j} \frac{1}{k} &= \ell + \sum_{k=1}^{\ell-1} \sum_{j=1}^{\ell-k} \frac{1}{k} \\
&= \ell + \sum_{k=1}^{\ell-1} \frac{\ell-k}{k} \\
&= \ell + \ell \cdot \sum_{k=1}^{\ell-1} \frac{1}{k} - (\ell-1) \\
&= \ell \cdot \sum_{k=1}^{\ell} \frac{1}{k} \,.
\end{aligned}$$

Dividing both sides by ℓ gives the inequality in (4). □

Tightness of Lemma 3. The following lemma shows that Lemma 3 is asymptotically tight.

Lemma 5 *For all positive integers m and $\ell_0 \leq m$, there is a string w of length n ($n \approx m(\ell_0 + \ln \ell_0)$) with $O(\ell m)$ distinct substrings of length ℓ for each $\ell \in [\ell_0]$, such that $C_{LZ77}(w) = \Omega(m \log \ell_0 + n/\ell_0)$.*

Proof: We construct such *bad* strings over the alphabet $[m]$. A *bad* string is constructed in ℓ_0 phases, where in each new phase, ℓ, we add a substring of length between m and $2m$ that might repeat substrings of length up to ℓ that appeared in the previous phases, but does not repeat longer substrings. Phase 1 contributes the string '$1 \ldots m$'. In phase $\ell > 1$, we list characters 1 to m in the increasing order, repeating all characters divisible by $\ell - 1$ twice. For example, phase 2 contributes the string '11 22 33$\ldots mm$', phase 3 the string '122 344 566$\ldots m$', phase 4 the string '1233 4566 7899$\ldots m$', etc. The spaces in the strings are introduced for clarity.

First observe that the length of the string, n, is at most $2m\ell_0$. Next, let us calculate the number of distinct substrings of various sizes. Since the alphabet size is m, there are m length-1 substrings. There are at most $2m$ length-2 substrings: '$i\ i$' and '$i\ (i+1)$' for every i in $[m-1]$, as well as '$m\ m$' and 'm 1'. We claim that for $1 < \ell \leq \ell_0$, there are at most $3\ell m$ length-ℓ substrings. Specifically, for every i in $[m]$, there are at most 3ℓ length-ℓ substrings that start with i. This is because each of the first ℓ phases contributes at most 2 such substrings: one that starts with '$i\ (i+1)$', and one that starts with '$i\ i$'. In the remaining phases a length-ℓ substring can have at most one repeated character, and so there are ℓ such substrings that start with i. Thus, there are at most $\ell \cdot 3m$ distinct length-ℓ substrings in the constructed string.

Finally, let us look at the cost of LZ77 compression. It is not hard to see that ℓth phase substring compresses by at most a factor of ℓ. Since each phase introduces a substring of length at least m, the total compressed length is at least $m(1 + 1/2 + 1/3 + \ldots + 1/\ell_0) = \Omega(m \log \ell_0) = \Omega(m \log \ell_0 + n/\ell_0)$. The last equality holds because $n \leq 2m\ell_0$ and, consequently, $\frac{n}{\ell_0} = o(m \log \ell_0)$. □

In the proof of Lemma 5 the alphabet size is large. It can be verified that by replacing each symbol from the large alphabet $[m]$ with its binary representation, we obtain a binary string of length $\Theta(m \log m\ell_0)$ with the properties stated in the lemma.

4.2 An Algorithm for LZ77

This subsection describes an algorithm for approximating the compressibility of an input string with respect to LZ77, which uses an approximation algorithm for DE (Definition 1) as a subroutine. The main tool in the reduction from LZ77 to DE consists of structural lemmas 2 and 3, summarized in the following corollary.

Corollary 4 *For any $\ell_0 \geq 1$, let $m = m(\ell_0) = \max_{\ell=1}^{\ell_0} \frac{d_\ell(w)}{\ell}$. Then*

$$m \ \leq \ C_{LZ77}(w) \ \leq \ 4 \cdot \left(m \log \ell_0 + \frac{n}{\ell_0} \right) .$$

The corollary allows us to approximate C_{LZ77} from estimates for d_ℓ for all $\ell \in [\ell_0]$. To obtain these estimates, we use the algorithm for DE, described in Subsection 4.3, as a subroutine. Recall that an algorithm for DE approximates the number of distinct symbols in an input string. We denote the number of distinct symbols in an input string τ by $C_{\mathrm{DSS}}(\tau)$. To approximate d_ℓ, the number of distinct length-ℓ substrings in w, using an algorithm for DE, we view each length-ℓ substring as a separate symbol. Each query of the algorithm for DE can be implemented by ℓ queries to w.

Let ESTIMATE(ℓ, B, δ) be a procedure that, given access to w, an index $\ell \in [n]$, an approximation parameter $B = B(n,\ell) > 1$ and a confidence parameter $\delta \in [0,1]$, computes a B-estimate for d_ℓ with probability at least $1-\delta$. It can be implemented using an algorithm for DE, as described above, and employing standard amplification techniques to boost success probability from $\frac{2}{3}$ to $1-\delta$: running the basic algorithm $\Theta(\log \delta^{-1})$ times and outputting the median. By Lemma 7, the query complexity of ESTIMATE(ℓ, B, δ) is $O\left(\frac{n}{B^2}\,\ell \log \delta^{-1}\right)$. Using ESTIMATE$(\ell, B, \delta)$ as a subroutine, we get the following approximation algorithm for the cost of LZ77.

ALGORITHM IV: AN (A,ε)-APPROXIMATION FOR $C_{\mathrm{LZ77}}(w)$

1. Set $\ell_0 = \left\lceil \frac{2}{A\varepsilon} \right\rceil$ and $B = \frac{A}{2\sqrt{\log(2/(A\varepsilon))}}$.
2. For all ℓ in $[\ell_0]$, let $\hat{d}_\ell = \text{ESTIMATE}(\ell, B, \frac{1}{3\ell_0})$.
   (*N.B.*: One can use the same queries for all ℓ_0 executions of ESTIMATE. See proof of Lemma 6.)
3. Combine the estimates to get an approximation of m from Corollary 4: set $\hat{m} = \max_\ell \frac{\hat{d}_\ell}{\ell}$.
4. Output $\widehat{C}_{\mathrm{LZ77}} = \hat{m} \cdot \frac{A}{B} + \varepsilon n$.

Lemma 6 (Theorem 3, part 1, restated) *Algorithm IV (A,ε)-estimates $C_{LZ77}(w)$. With a proper implementation that reuses queries and an appropriate data structure, its query and time complexity are $\tilde{O}\left(\frac{n}{A^3\varepsilon}\right)$.*

Proof: By the union bound, with probability $\geq \frac{2}{3}$, all values $\hat{d}_\ell$ computed by the algorithm are B-estimates for the corresponding d_ℓ. When this holds, $\hat{m}$ is a B-estimate for m from Corollary 4, which implies that

$$\frac{\hat{m}}{B} \;\leq\; C_{\mathrm{LZ77}}(w) \;\leq\; 4 \cdot \left(\hat{m} B \log \ell_0 + \frac{n}{\ell_0}\right) .$$

Equivalently, $\frac{C_{\mathrm{LZ77}} - 4(n/\ell_0)}{4B\log\ell_0} \leq \hat{m} \leq B \cdot C_{\mathrm{LZ77}}$. Multiplying all three terms by $\frac{A}{B}$ and adding εn to them, and then substituting parameter settings for ℓ_0 and B, specified in the algorithm, shows that $\widehat{C}_{\mathrm{LZ77}}$ is indeed an (A,ε)-estimate for C_{LZ77}.

As explained before the algorithm statement, each call to ESTIMATE$(\ell, B, \frac{1}{3\ell_0})$ costs $O\left(\frac{n}{B^2}\,\ell \log \ell_0\right)$ queries. Since the subroutine is called for all $\ell \in [\ell_0]$, the straightforward implementation of the algorithm would result in $O\left(\frac{n}{B^2}\ell_0^2 \log \ell_0\right)$

queries. Our analysis of the algorithm, however, does not rely on independence of queries used in different calls to the subroutine, since we employ the union bound to calculate the error probability. It will still apply if we first run ESTIMATE to approximate d_{ℓ_0} and then reuse its queries for the remaining calls to the subroutine, as though it queried only the length-ℓ prefixes of the length-ℓ_0 substrings queried in the first call. With this implementation, the query complexity is $O\left(\frac{n}{B^2}\ell_0 \log \ell_0\right) = O\left(\frac{n}{A^3\varepsilon}\log^2 \frac{1}{A\varepsilon}\right)$. To get the same running time, one can maintain counters for all $\ell \in [\ell_0]$ for the number of distinct length-ℓ substrings seen so far and use a trie to keep the information about the queried substrings. Every time a new node at some depth ℓ is added to the trie, the ℓth counter is incremented. □

4.3 A Simple Algorithm for DE

Here we describe a simple approximation algorithm for DE. The Guaranteed-Error estimator of Charikar *et al.* has the same guarantees as our approximation algorithm. Our algorithm is (even) simpler, and we present it here for completeness.

ALGORITHM V: AN A-APPROXIMATION FOR DE

1. Take $\frac{10n}{A^2}$ samples from the string τ.
2. Let $\widehat{C}$ be the number of distinct symbols in the sample; output $\widehat{C}\cdot A$.

Lemma 7 *Let $A = A(n)$. Algorithm V is an A-approximation algorithm for* DE *whose query complexity and running time are $O\left(\frac{n}{A^2}\right)$.*

Proof: Let C be the number of distinct symbols in the string τ. We need to show that $\frac{C}{A} \le \widehat{C}\cdot A \le C\cdot A$, or equivalently, $\frac{C}{A^2} \le \widehat{C} \le C$, with probability at least $\frac{2}{3}$. The sample always contains at most as many distinct symbols as there are in τ: $\widehat{C} \le C$. Claim 8, stated below and applied with $s = \frac{10n}{A^2}$, shows that $\widehat{C} \ge \frac{C}{A^2}$ with probability $\ge \frac{2}{3}$. To get the running time $O\left(\frac{n}{A^2}\right)$ one can use a random 2-universal hash function. □

Claim 8 *Let $s = s(n) \le n$. Then s independent samples from a distribution with $C = C(n)$ elements, where each element has probability $\ge \frac{1}{n}$, yield at least $\frac{Cs}{10n}$ distinct elements, with probability $\ge \frac{3}{4}$.*

Proof: For $i \in [C]$, let X_i be the indicator variable for the event that color i is selected in s samples. Then $X = \sum_{i=1}^{C} X_i$ is a random variable for the number of distinct colors. Since each color is selected with probability at least $\frac{1}{n}$ for each sample,

$$\begin{aligned} \mathsf{E}[X] &= \sum_{i=1}^{C} \mathsf{E}[X_i] \ge C\left(1 - \left(1 - \frac{1}{n}\right)^s\right) \\ &\ge C\left(1 - e^{-(s/n)}\right) \ge (1 - e^{-1})\frac{Cs}{n}. \end{aligned} \tag{6}$$

The last inequality holds because $1-e^{-x} \geq (1-e^{-1}) \cdot x$ for all $x \in [0,1]$.

We now use Chebyshev's inequality to bound the probability that X is far from its expectation. For any distinct pair of colors i, j, the covariance $\mathsf{E}(X_i X_j) - \mathsf{E}(X_i)\,\mathsf{E}(X_j)$ is negative (knowing that one color was not selected makes it more likely for any other color to be selected). Since X is a sum of Bernoulli variables, $\mathsf{Var}[X] \leq \mathsf{E}[X]$. For any $\delta > 0$,

$$\begin{aligned} \Pr[X \leq \delta\,\mathsf{E}[X]] &\leq \Pr[|X - \mathsf{E}[X]| \geq (1-\delta)\,\mathsf{E}[X]] \\ &\leq \frac{\mathsf{Var}[X]}{((1-\delta)\,\mathsf{E}[X])^2} \\ &\leq \frac{1}{(1-\delta)^2\,\mathsf{E}[X]}. \end{aligned} \tag{7}$$

Set $\delta = 3-\sqrt{8}$. If $\mathsf{E}[X] \geq \frac{4}{(1-\delta)^2}$, then by (7) and (6), with probability $\geq \frac{3}{4}$, variable $X \geq \delta\,\mathsf{E}[X] \geq \delta(1-e^{-1})\frac{Cs}{n} > \frac{Cs}{10n}$, as stated in the claim. Otherwise, that is, if $\mathsf{E}[X] < \frac{4}{(1-\delta)^2}$, Equation (6) implies that $\frac{4\delta}{(1-\delta)^2} > \delta(1-e^{-1})\frac{Cs}{n}$. Substituting $3-\sqrt{8}$ for δ gives $1 > \frac{Cs}{10n}$. In other words, the claim for this case is that at least one color appears among the samples, which, clearly, always holds. □

4.4 Lower Bounds: Reducing DE to LZ77

We have demonstrated that estimating the LZ77 compressibility of a string reduces to DE. As shown in [37], DE is quite hard, and it is not possible to improve much on the simple approximation algorithm in Subsection 4.3, on which we base the LZ77 approximation algorithm in the previous subsection. A natural question is whether there is a better algorithm for the LZ77 estimation problem. That is, is the LZ77 estimation strictly easier than DE? As we shall see, it is not much easier in general.

Lemma 9 (Reduction from DE **to LZ77)** *Suppose there exists an algorithm $\mathscr{A}_{LZ77}$ that, given access to a string w of length n over an alphabet Σ, performs $q = q(n, |\Sigma|, \alpha, \beta)$ queries and with probability at least $5/6$ distinguishes between the case that $C_{LZ77}(w) \leq \alpha n$ and the case that $C_{LZ77}(w) > \beta n$, for some $\alpha < \beta$.*

Then there is an algorithm for DE *taking inputs of length $n' = \Theta(\alpha n)$ that performs q queries and, with probability at least $2/3$, distinguishes inputs with at most $\alpha' n'$ distinct symbols from those with at least $\beta' n'$ distinct symbols, $\alpha' = \alpha/2$ and $\beta' = \beta \cdot 2 \cdot \max\left\{1, \frac{4\log n'}{\log|\Sigma|}\right\}$.*

Two notes are in place regarding the reduction. The first is that the gap between the parameters α' and β' that is required by the DE algorithm obtained in Lemma 9, is larger than the gap between the parameters α and β for which the LZ77-compressibility algorithm works, by a factor of $4 \cdot \max\left\{1, \frac{4\log n'}{\log|\Sigma|}\right\}$. In particular, for binary strings $\frac{\beta'}{\alpha'} = O\left(\log n' \cdot \frac{\beta}{\alpha}\right)$, while if the alphabet is large, say, of size at least n', then $\frac{\beta'}{\alpha'} = O\left(\frac{\beta}{\alpha}\right)$. In general, the gap increases by at most $O(\log n')$. The

second note is that the number of queries, q, is a function of the parameters of the LZ77-compressibility problem and, in particular, of the length of the input strings, n. Hence, when writing q as a function of the parameters of DE and, in particular, as a function of $n' = \Theta(\alpha n)$, the complexity may be somewhat larger. It is an open question whether a reduction without such increase is possible.

Prior to proving the lemma, we discuss its implications. [37] give a strong lower bound on the sample complexity of approximation algorithms for DE. An interesting special case is that a subpolynomial-factor approximation for DE requires many queries even with a promise that the strings are only slightly compressible: for any $B = n^{o(1)}$, distinguishing inputs with $n/11$ distinct symbols from those with n/B distinct symbols requires $n^{1-o(1)}$ queries. Lemma 9 extends that bound to estimating LZ77 compressibility, as stated in Theorem 3. In fact, the lower bound for DE in [37] applies to a broad range of parameters, and yields the following general statement when combined with Lemma 9:

Corollary 5 (LZ77 is Hard to Approximate with Few Samples) *For sufficiently large n, all alphabets Σ and all $B \leq n^{1/4}/(4\log n^{3/2})$, there exist $\alpha, \beta \in (0,1)$ where $\beta = \Omega\left(\min\left\{1, \frac{\log|\Sigma|}{4\log n}\right\}\right)$ and $\alpha = O\left(\frac{\beta}{B}\right)$, such that every algorithm that distinguishes between the case that $C_{LZ77}(w) \leq \alpha n$ and the case that $C_{LZ77}(w) > \beta n$ for $w \in \Sigma^n$, must perform $\Omega\left(\left(\frac{n}{B'}\right)^{1-\frac{2}{k}}\right)$ queries for $B' = \Theta\left(B \cdot \max\left\{1, \frac{4\log n}{\log|\Sigma|}\right\}\right)$ and $k = \Theta\left(\sqrt{\frac{\log n}{\log B' + \frac{1}{2}\log\log n}}\right)$.*

Proof of Lemma 9. Suppose we have an algorithm $\mathscr{A}_{\text{LZ77}}$ for LZ77-compressibility as specified in the premise of Lemma 9. Here we show how to transform a DE instance τ into an input for $\mathscr{A}_{\text{LZ77}}$, and use the output of $\mathscr{A}_{\text{LZ77}}$ to distinguish τ with at most $\alpha' n'$ distinct symbols from τ with at least $\beta' n'$ distinct symbols, where α' and β' are as specified in the lemma. We shall assume that $\beta' n'$ is bounded below by some sufficiently large constant. Recall that in the reduction from LZ77 to DE, we transformed substrings into single symbols. Here we perform the reverse operation.

Given a DE instance τ of length n', we transform it into a string of length $n = n' \cdot k$ over Σ, where $k = \lceil \frac{1}{\alpha} \rceil$. We then run $\mathscr{A}_{\text{LZ77}}$ on w to obtain information about τ. We begin by replacing each distinct symbol in τ with a uniformly selected substring in Σ^k. The string w is the concatenation of the corresponding substrings (which we call *blocks*). We show that:

1. If τ has at most $\alpha' n'$ distinct symbols, then $C_{\text{LZ77}}(w) \leq 2\alpha' n$;
2. If τ has at least $\beta' n'$ distinct symbols, then

$$\Pr_w[C_{\text{LZ77}}(w) \geq \frac{1}{2} \cdot \min\left\{1, \frac{\log|\Sigma|}{4\log n'}\right\} \cdot \beta' n] \geq \frac{7}{8}.$$

That is, in the first case we get an input w such that $C_{\text{LZ77}}(w) \leq \alpha n$ for $\alpha = 2\alpha'$, and in the second case, with probability at least $7/8$, $C_{\text{LZ77}}(w) \geq \beta n$ for $\beta = \frac{1}{2} \cdot \min\left\{1, \frac{\log|\Sigma|}{4\log n'}\right\} \cdot \beta'$. Recall that the gap between α' and β' is assumed to be sufficiently large so that $\alpha < \beta$. To distinguish the case that $C_{\text{DSS}}(\tau) \leq \alpha' n'$ from the

case that $C_{\mathrm{DSS}}(\tau) > \beta' n'$, we can run $\mathscr{A}_{\mathrm{LZ77}}$ on w and output its answer. Taking into account the failure probability of $\mathscr{A}_{\mathrm{LZ77}}$ and the failure probability in Item 2 above, the Lemma follows.

Before we prove these two claims, we observe that in order to run the algorithm $\mathscr{A}_{\mathrm{LZ77}}$, there is no need to generate the whole string w. Rather, upon each query of $\mathscr{A}_{\mathrm{LZ77}}$ to w, if the index of the query belongs to a block that has already been generated, the answer to $\mathscr{A}_{\mathrm{LZ77}}$ is determined. Otherwise, we query the symbol in τ that corresponds to the block. If this symbol was not yet observed, then we set the block to a uniformly selected substring in Σ^k. If this symbol was already observed in τ, then we set the block according to the substring that was already selected for the symbol. In either case, the query to w can now be answered. Thus, each query to w is answered by performing at most one query to τ.

It remains to prove the two items concerning the relation between the number of colors in τ and $C_{\mathrm{LZ77}}(w)$. If τ has at most $\alpha' n'$ distinct symbols then w contains at most $\alpha' n'$ distinct blocks. Since each block is of length k, at most k phrases start in each new block. By definition of LZ77, at most one phrases starts in each repeated block. Hence,

$$C_{\mathrm{LZ77}}(w) \leq \alpha' n' \cdot k + (1-\alpha')n' \leq \alpha' n + n' \leq 2\alpha' n.$$

If τ contains $\beta' n'$ or more distinct symbols, w is generated using at least $\beta' n' \cdot \log(|\Sigma|^k) = \beta' n \log|\Sigma|$ random bits. Hence, with high probability (e.g., at least $7/8$) over the choice of these random bits, any lossless compression algorithm (and in particular LZ77) must use at least $\beta' n \log|\Sigma| - 3$ bits to compress w. Each symbol of the compressed version of w can be represented by $\max\{\lceil \log|\Sigma| \rceil, 2\lceil \log n \rceil\} + 1$ bits, since it is either an alphabet symbol or a pointer-length pair. Since $n = n' \lceil 1/\alpha' \rceil$, and $\alpha' > 1/n'$, each symbol takes at most $\max\{4\log n', \log|\Sigma|\} + 2$ bits to represent. This means the number of symbols in the compressed version of w is

$$C_{\mathrm{LZ77}}(w) \geq \frac{\beta' n \log|\Sigma| - 3}{\max\{4\log n', \log|\Sigma|\}) + 2} \geq \frac{1}{2} \cdot \beta' n \cdot \min\left\{1, \tfrac{\log|\Sigma|}{4\log n'}\right\}$$

where we have used the fact that $\beta' n'$, and hence $\beta' n$, is at least some sufficiently large constant. □

Acknowledgements We would like to thank Amir Shpilka, who was involved in a related paper on distribution support testing [37] and whose comments greatly improved drafts of this article. We would also like to thank Eric Lehman for discussing his thesis material with us and Oded Goldreich and Omer Reingold for helpful comments. Finally, we thank several anonymous reviewers for helpful comments, especially regarding previous work.

References

1. Ahmed, N., Natarajan, T., Rao, K.R.: Discrete cosine transform. IEEE Transactions on Computers **23**(1), 90–93 (1974)
2. Alon, N., Matias, Y., Szegedy, M.: The space complexity of approximating the frequency moments. Journal of Computer and System Sciences **58**(1), 137–147 (1999)

3. Bar-Yossef, Z., Kumar, R., Sivakumar, D.: Sampling algorithms: lower bounds and applications. In: Proceedings of the Thirty-Third Annual ACM Symposium on the Theory of Computing (STOC), pp. 266–275 (2001)
4. Batu, T., Dasgupta, S., Kumar, R., Rubinfeld, R.: The complexity of approximating the entropy. SIAM Journal on Computing **35**(1), 132–150 (2005)
5. Benedetto, D., Caglioti, E., Loreto, V.: Language trees and zipping. Physical Review Letters **88**(4), 048,702 (2002). See comment by Khmelev DV, Teahan WJ, in Physical Review Letters, **90**(8), 089803 (2003) and the reply in Physical Review Letters, **90**(8), 089804 (2003).
6. Brautbar, M., Samorodnitsky, A.: Approximating entropy from sublinear samples. In: Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 366–375 (2007)
7. Bunge, J.: Bibliography on estimating the number of classes in a population. `www.stat.cornell.edu/~bunge/bibliography.htm`
8. Burrows, M., Wheeler, D.: A block sorting lossless data compression algorithm. Tech. Rep. 124, Digital Equipment Corporation (1994)
9. Cai, H., Kulkarni, S.R., Verdú, S.: Universal entropy estimation via block sorting. IEEE Transactions on Information Theory **50**(7), 1551–1561 (2004)
10. Charikar, M., Chaudhuri, S., Motwani, R., Narasayya, V.R.: Towards estimation error guarantees for distinct values. In: Proceedings of the Nineteenth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS), pp. 268–279. ACM (2000)
11. Chui, C.K.: An Introduction to Wavelets. San Diego: Academic Press (1992)
12. Cilibrasi, R., Vitányi, P.M.B.: Clustering by compression. IEEE Transactions on Information Theory **51**(4), 1523–1545 (2005)
13. Cilibrasi, R., Vitányi, P.M.B.: Similarity of objects and the meaning of words. In: J. Cai, S.B. Cooper, A. Li (eds.) Proceedings of the Third International Conference on Theory and Applications of Models of Computation (TAMC), *Lecture Notes in Computer Science*, vol. 3959, pp. 21–45. Springer (2006)
14. Cleary, J., Witten, I.: Data compression using adaptive coding and partial string matching. IEEE Transactions on Communications **32**(4), 396–402 (1984)
15. Cormode, G., Muthukrishnan, S.: Substring compression problems. In: Proceedings of the Thirty-Third Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 321–330 (2005)
16. Cover, T., Thomas, J.: Elements of Information Theory. Wiley & Sons (1991)
17. Ferragina, P., Giancarlo, R., Greco, V., Manzini, G., G.Valiente: Compression-based classification of biological sequences and structures via the universal similarity metric: Experimental assessment. BMC Bioinformatics **8**(252) (2007)
18. Frank, E., Chui, C., Witten, I.H.: Text categorization using compression models. In: Proceedings of the Data Compression Conference (DCC), p. 555 (2000)
19. Gheorghiciuc, I., Ward, M.: On correlation polynomials and subword complexity. In: Discrete Math and Theoretical Computer Science (DMTCS) Proceedings of the Conference on Analysis of Algorithms (AofA), pp. 1–18 (2007)
20. Ilie, L., Yu, S., Zhang, K.: Repetition complexity of words. In: O.H. Ibarra, L. Zhang (eds.) Proceedings of the 8th Annual International Conference on Computing and Combinatorics (COCOON), *Lecture Notes in Computer Science*, vol. 2387, pp. 320–329. Springer (2002)
21. Janson, S., Lonardi, S., Szpankowski, W.: On average sequence complexity. Theoretical Computer Science **326**(1–3), 213–227 (2004)
22. Kása, Z.: On the d-complexity of strings. Pure Mathematics and Application **9**(1–2), 119–128 (1998)
23. Keller, O., Kopelowitz, T., Landau, S., Lewenstein, M.: Generalized substring compression. In: Proceedings of the 20th Annual Symposium on Combinatorial Pattern Matching (CPM), pp. 26–38 (2009)
24. Keogh, E., Lonardi, S., Ratanamahatana, C.: Towards parameter-free data mining. In: Proceedings of ACM Conference on Knowledge Discovery and Data Mining (KDD), pp. 206–215 (2004)
25. Keogh, E.J., Keogh, L., Handley, J.: Compression-based data mining. In: J. Wang (ed.) Encyclopedia of Data Warehousing and Mining, pp. 278–285. IGI Global (2009)
26. Kukushkina, O.V., Polikarpov, A.A., Khmelev, D.V.: Using literal and grammatical statistics for authorship attribution. Problemy Peredachi Informatsii **37**(2), 96–98 (2000). [Problems of Information Transmission (Engl. Transl.) 37, 172–184 (2001)]
27. Lehman, E., Shelat, A.: Approximation algorithms for grammar-based compression. In: Proceedings of the Thirteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 205–212 (2002)

28. Levé, F., Séébold, P.: Proof of a conjecture on word complexity. Bull. Belg. Math. Soc. **8**(2), 277–291 (2001)
29. Li, M., Chen, X., Li, X., Ma, B., Vitányi, P.M.B.: The similarity metric. IEEE Transactions on Information Theory **50**(12), 3250–3264 (2004)
30. Li, M., Vitányi, P.: An Introduction to Kolmogorov Complexity and Its Applications. Springer (1997)
31. Loewenstern, D., Hirsh, H., Noordewier, M., Yianilos, P.: DNA sequence classification using compression-based induction. Tech. Rep. 95-04, Rutgers University, DIMACS (1995)
32. de Luca, A.: On the combinatorics of finite words. Theoretical Computer Science **218**(1), 13–39 (1999)
33. Paninski, L.: Estimation of entropy and mutual information. Neural Computation **15**(6), 1191–1253 (2003)
34. Paninski, L.: Estimating entropy on m bins given fewer than m samples. IEEE Transactions on Information Theory **50**(9), 2200–2203 (2004)
35. Pierce II, L., Shields, P.C.: Sequences incompressible by SLZ (LZW), yet fully compressible by ULZ. In: Numbers, Information and Complexity, I, pp. 385–390. Norwell, MA: Kluwer (2000)
36. Raskhodnikova, S., Ron, D., Rubinfeld, R., Smith, A.: Sublinear algorithms for approximating string compressibility. In: Proceedings of the Eleventh International Workshop on Randomization and Computation (RANDOM), pp. 609–623 (2007)
37. Raskhodnikova, S., Ron, D., Shpilka, A., Smith, A.: Strong lower bounds for approximating distribution support size and the distinct elements problem. SIAM Journal on Computing **39**(3), 813–842 (2009)
38. Sculley, D., Brodley, C.E.: Compression and machine learning: A new perspective on feature space vectors. In: Proceedings of the Data Compression Conference (DCC), pp. 332–341 (2006)
39. Shallit, J.: On the maximum number of distinct factors of a binary string. Graphs and Combinatorics **9**(2), 197–200 (1993)
40. Willems, F.M.J., Shtarkov, Y.M., Tjalkens, T.J.: The context-tree weighting method: basic properties. IEEE Transactions on Information Theory **41**(3), 653–664 (1995)
41. Witten, I.H., Bray, Z., Mahoui, M., Teahan, W.J.: Text mining: A new frontier for lossless compression. In: Proceedings of the Data Compression Conference (DCC), pp. 198–207 (1999)
42. Ziv, J., Lempel, A.: A universal algorithm for sequential data compression. IEEE Transactions on Information Theory **23**, 337–343 (1977)
43. Ziv, J., Lempel, A.: Compression of individual sequences via variable-rate coding. IEEE Transactions on Information Theory **24**, 530–536 (1978)