Electronic Colloquium on Computational Complexity, Report No. 125 (2005)

Sublinear Algorithms for Approximating String Compressibility and the Distribution Support Size

Sofya Raskhodnikova* Dana Ron† Ronitt Rubinfeld‡ Amir Shpilka§
Adam Smith*

November 2, 2005

Abstract

We raise the question of approximating 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 Lempel-Ziv (LZ), and present algorithms and lower bounds for approximating compressibility with respect to both schemes. While we obtain much stronger results for RLE in terms of the efficiency of the algorithms, our investigation of LZ yields results whose interest goes beyond the initial questions we set out to study. In particular, we prove combinatorial structural lemmas that relate compressibility of a string with respect to Lempel-Ziv to the number of distinct short substrings contained in it. We also show that compressibility with respect to LZ is related to approximating the support size of a distribution. This problem has been considered under different guises in the literature. We prove a strong lower bound for it, at the heart of which is a construction of two positive integer random variables, X_1 and X_2, with very different expectations and the following condition on the moments up to k: $\mathsf{E}[X_1]/\mathsf{E}[X_2] = \mathsf{E}[X_1^2]/\mathsf{E}[X_2^2] = \ldots = \mathsf{E}[X_1^k]/\mathsf{E}[X_2^k]$.

*Weizmann Institute of Science, Rehovot, Israel. Email: <firstname>.<lastname>@weizmann.ac.il. A.S. is supported by the Louis L. and Anita M. Perlman Postdoctoral Fellowship.

†Tel Aviv University, Ramat Aviv, Israel. Email: danar@eng.tau.ac.il. Supported by the Israel Science Foundation.

‡MIT, Cambridge MA, USA. Email: ronitt@csail.mit.edu.

§Technion, Haifa, Israel. Email: shpilka@cs.technion.ac.il. Supported by the Koshland fellowship.

ISSN 1433-8092

Contents

1 Introduction

Imagine having to choose between a few compression schemes to compress a very long file. Before deciding on the scheme, you might want to obtain a rough estimate on how well each scheme performs on your file. We raise the question of approximating 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 Lempel-Ziv (LZ) [ZL77]. In the RLE scheme, each run, or a sequence of consecutive occurrences of the same character, is stored as 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, where long runs are likely. In the variant of LZ that we study, a left-to-right pass of the input string is performed and each sequence of characters that has already appeared in the previous portion of the string is replaced with the pointer to the previous location and the length of the sequence. Many popular archivers, such as zip, use variations on LZ scheme. We present sublinear algorithms and corresponding lower bounds for approximating compressibility with respect to both schemes. While we obtain much stronger results for RLE in terms of the efficiency of the algorithms, our investigation of LZ yields results whose interest goes beyond the initial questions we set out to study.

We consider three approximation notions: additive, multiplicative and the combination of additive and multiplicative. An *additive approximation* algorithm is allowed an additive error of ϵn, where n is the length of the input and $\epsilon \in (0,1)$ is a parameter. The output of a *multiplicative approximation* algorithm is within a factor $A > 1$ of the correct answer. In the combined notion we allow both types of errors: the algorithm should output an estimate $\widehat{C}$ of C such that $\frac{C}{A} - \epsilon n \leq \widehat{C} \leq A \cdot C + \epsilon n$. Our algorithms are randomized, and for all inputs the approximation guarantee holds with probability at least $\frac{2}{3}$.

Results for Run-Length Encoding

For RLE, we give sublinear algorithms for all three approximation notions defined above, providing a trade-off between the quality of approximation and the running time. The algorithms that allow an additive approximation are fast. Specifically, an ϵn-additive estimate can be obtained in time $\tilde{O}(1/\epsilon^3)$, and a combined estimate, with a multiplicative error of 3 and an additive error of ϵn, can be obtained in time $\tilde{O}(1/\epsilon)$. A 4-multiplicative approximation algorithm has running time reversely proportional to the compressibility of the input string.[1] Specifically, its running time is $\tilde{O}(n/C_{\mathrm{rle}}(w))$ where $C_{\mathrm{rle}}(w)$ denotes the compression cost of the string w. Thus, it 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 $\Omega(1/\epsilon^2)$ for ϵn-additive approximation.

Needle-in-a-haystack lower bounds. Our first lower bound mentioned above, for multiplicative approximation of RLE, is based on the difficulty of the following task: Distinguishing between

[1] The aforementioned constants, 3 and 4, can be improved to any constant greater than 1. For simplicity of the presentation, we give the full analysis for these particular constants.

the string 1^n, which consists of a single run, and strings that differ form 1^n in only few (randomly selected) positions, which have a few runs. Namely, the strings in question are very compressible, and the difficulty of the distinguishing task is solely based on finding the few "hidden" 0s. We refer to such a hardness result as a *needle-in-a-haystack* lower bound.

While such a lower bound seems discouraging, we claim it is not a significant obstacle for giving a meaningful solution to the problem we raise. The "bad" strings in the lower bound proof are highly compressible, while most strings are not. Also, the user in the scenario in the beginning of the paper probably does not care exactly how compressible his very compressible files are. Only in the case of not very compressible files he is interested in learning the compression cost more precisely. Therefore, it is worth trying to circumvent a needle-in-a-haystack lower bound when possible by giving approximation algorithms with an additive error or running time that depends on the compression cost. As noted above, for RLE we indeed obtain very efficient approximation algorithms that are allowed an additive error, and a multiplicative approximation algorithm that is very efficient when executed on strings that are not too compressible. As we explain next, for LZ such efficient algorithms do not exist.

Results for Lempel-Ziv

For the LZ compression scheme, we give an approximation algorithm with both multiplicative and additive error and a strong lower bound to justify that for this compression scheme we do not obtain very efficient algorithms even for strings that are not very compressible. The main tool in the algorithm is two combinatorial structural lemmas that relate compressibility of the string to the number of distinct short substrings contained in it. Roughly, they say that a string is well compressible with respect to LZ if and only if it has few distinct substrings of length ℓ for all small ℓ. The lemmas were inspired by a structural lemma for grammars by Lehman and Shelat [LS02]. Our lower bound implies that constant factor approximation for the LZ cost requires an almost linear number of queries, even for only slightly compressible strings. The same lower bound applies to additive approximation, and the bound generalizes to larger factors of approximation.

Colors and Distribution Support Size

Both for the algorithm and for the lower bound, we show that approximating compressibility with respect to LZ is closely related to the following problem, which we call Colors:

Definition 1.1 (Colors Problem) *Given access to a string τ over some alphabet Ψ, estimate the number of* distinct *symbols ("colors") in τ.*

We show that one may assume without loss of generality that an algorithm for Colors only takes uniform random samples with replacement from the string. This is a strengthening of a result from [Bar02] for the specific case of Colors. Hence, we view the problem as estimating the number of different colors in a set of n "colored balls", following a sequence of trials in each of which we get to see the color of a uniformly selected ball.[2] A closely related problem is that of approximating the support size of a distribution: Given access to independent samples from a distribution where each element appears with probability at least $\frac{1}{n}$, approximate the distribution support size.

[2]The balls have no identity: the algorithm cannot distinguish two different balls of the same color.

Variants of this problem have been considered under various guises in the literature, different fields competing for the best name for it: in databases it is referred to as approximating distinct values [CCMN00], in statistics as estimating the number of species in a population (see 829 references on www.stat.cornell.edu/∼bunge/bibliography.htm), in the old days as estimating the dictionary size of a long text [Shl81] and recently in theoretical computer science as approximating the frequency moment F_0 [AMS99, BKS01].

Most of these works, however, consider models different from ours. For our model, there is an A-multiplicative approximation algorithm of Charikar *et al.* [CCMN00], called the Guaranteed-Error estimator, that runs in $O\left(\frac{n}{A^2}\right)$, and a matching needle-in-a-haystack lower bound [CCMN00, BKS01]. The lower bound boils down to the observation that every algorithm requires $\Omega\left(\frac{n}{A^2}\right)$ queries to distinguish between the input with one color and the same input with A^2 unique colors inserted in random positions.

As discussed above, we are interested in bounds that apply to "slightly" compressible strings. In our reductions, such strings correspond to inputs with many colors, and so the needle-in-a-haystack lower bound of [CCMN00] leaves the following question: How fast can one distinguish an input with $\frac{n}{d_1}$ colors from an input with $\frac{n}{d_2}$ colors, where d_1 and d_2 are different constants? We prove that $n^{1-o(1)}$ queries are necessary; the same bound holds even when d_1 is constant and d_2 is $n^{o(1)}$.

To prove this, we construct two distributions on input instances, with $\frac{n}{d_1}$ and $\frac{n}{d_2}$ colors, respectively, that are hard to distinguish. Building on the ideas from [BDKR05], we prove that any algorithm for Colors and Distribution Support Size can be simulated by an algorithm that is only provided with the *histogram* of a uniformly selected sample. That is, the algorithm is only given the number of colors in the sample that appear once, twice, thrice, etc. Thus, in the two input instances we construct, we ensure that the induced distributions on these histograms are very close. At the heart of the construction are two positive integer random variables, X_1 and X_2, that correspond to the two input distributions. These random variables have very different expectations (which translate to different numbers of colors) and the following condition on the moments up to k (which translates to similar histograms):

$$\frac{\mathsf{E}[X_1]}{\mathsf{E}[X_2]} = \frac{\mathsf{E}[X_1^2]}{\mathsf{E}[X_2^2]} = \cdots = \frac{\mathsf{E}[X_1^k]}{\mathsf{E}[X_2^k]}.$$

Our lower bound technique can be extended to other problems where one needs to compute quantities invariant under the permutation of the balls and the colors. In particular, our technique gives a lower bound of $\Omega\left(n^{\frac{2}{6A^2-3+o(1)}}\right)$ on approximating the entropy of a distribution over n elements to within a multiplicative factor of A. When A is close to 1, this bound is close to $n^{2/3}$. It can be combined with the $\Omega\left(n^{\frac{1}{2A^2}}\right)$ lower bound in [BDKR05] to give $\Omega\left(n^{\max\left\{\frac{1}{2A^2}, \frac{2}{6A^2-3+o(1)}\right\}}\right)$.

Other compression schemes

It is interesting to approximate the compression cost for schemes other than RLE and LZ. One observation is that for Huffman encoding the compression cost is equal to the entropy. Approximating entropy has been studied [BDKR05, BS05], and this immediately gives algorithms and lower bounds for compressibility under Huffman encoding.

Also, there is no reason to restrict our attention only to lossless schemes, like RLE, LZ and Huffman. It would be interesting to come up with approximation algorithms for schemes like JPEG (based on the discrete cosine transform), MPEG and MP3. One lossy compression scheme to which our results extend directly is Lossy RLE, where some characters, e.g., the ones that represent similar colors, are treated as the same character.

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 3.1 and the lower bounds, in Theorems 3.2 and 3.4. Section 4 deals with the LZ scheme: it starts with the structural lemmas, explains the approximation algorithm for compressibility with respect to LZ and finishes with the reduction from Colors to LZ compressibility. Section 5 deals with algorithms for Colors. In Section 6, we explain the lower bound for Colors, including the construction of integer distributions with our moment conditions. In Appendix A we use our lower-bound technique to obtain a lower bound for approximating the entropy of a discrete distribution. Appendix B proves a couple of technical lemmas used in Section 6.

2 Preliminaries

We use $[n]$ to denote $\{1, \ldots, n\}$.

Two random variables X and Y over a domain S have *statistical difference* δ if $\max_{S' \subset S} |\Pr[X \in S'] - \Pr[Y \in S']| = \delta$. It directly follows that for every algorithm $\mathcal{A}$, $\Big|\Pr[\mathcal{A}(X) = 1] - \Pr[\mathcal{A}(Y) = 1]\Big| \leq \delta$. We write $X \approx_\delta Y$ to indicate that X and Y have statistical difference at most δ.

The input to our algorithms is usually a string w of length n over a finite alphabet Σ. We consider estimates to various quantities, such as compression cost of w under a specific algorithm, that have both multiplicative and additive error. We call $\widehat{C}$ an (A, ϵ)-*estimate* for C if

$$\frac{C}{A} - \epsilon n \;\leq\; \widehat{C} \;\leq\; A \cdot C + \epsilon n \,,$$

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

When the error is purely additive or multiplicative, we use the following shorthand: ϵn-*additive estimate* stands for $(1, \epsilon)$-*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*.

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

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 $\tilde{O}(1/\epsilon^3)$ queries. We then reduce the query complexity to $\tilde{O}(1/\epsilon)$ at the cost of incurring a multiplicative approximation error in addition to additive: the new algorithm $(3, \epsilon)$-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 3.1. We close this section with lower bounds for approximating $C_{\mathrm{rle}}(w)$ (Theorems 3.2 and 3.4).

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

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

By definition, $\dfrac{C_{\mathrm{rle}}(w)}{n} = \mathop{\mathsf{E}}_{t \in [n]}[c(t)]$. 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 $q = \Theta\left(\frac{1}{\epsilon^2}\right)$ indices $t_1, \ldots, t_q$ uniformly and independently at random.
2. For each $i \in [q]$:
   (a) Query t_i and up to $\ell_0 = \frac{8 \log(4|\Sigma|/\epsilon)}{\epsilon}$ 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 $\widehat{C}_{\mathrm{rle}} = n \cdot \mathop{\mathsf{E}}_{i \in [q]}[\hat{c}(t_i)]$.

Correctness. 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,

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

Therefore,

$$\frac{C_{\mathrm{rle}}(w)}{n} - \frac{\epsilon}{2} \;\leq\; \frac{1}{n} \cdot \sum_{t:\, \ell(t) < \ell_0} c(t) \;\leq\; \frac{C_{\mathrm{rle}}(w)}{n}. \tag{3}$$

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

By an additive Chernoff bound, with high constant probability, the sampling error in estimating $\mathsf{E}[\hat{c}(t_i)]$ is at most $\epsilon/2$. (Recall that $|\Sigma|$ is a constant so that $c(t) = O(1)$ for every t.) Therefore, $\widehat{C}_{\mathrm{rle}}$ is an ϵn-additive estimate of $C_{\mathrm{rle}}(w)$, as desired.

Query complexity. Since the number of queries performed for each selected t_i is $O(\ell_0) = O(\log(1/\epsilon)/\epsilon)$, the total number of queries, as well as the running time is $O(\log(1/\epsilon)/\epsilon^3)$.

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

Let ℓ_0 be as defined in the previous subsection, and let $h_0 = \lceil \log \ell_0 \rceil$. Thus, $h_0 = O(\log(1/\epsilon))$. 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 $s \stackrel{\text{def}}{=} \lceil \log |\Sigma| \rceil$ and

$$C_{\text{rle}}(w,h) \stackrel{\text{def}}{=} \sum_{t \in B_h} c(t). \tag{4}$$

Then

$$|B_h| \cdot \frac{h+s+1}{2^h} \;\le\; C_{\text{rle}}(w,h) \;\le\; |B_h| \cdot \frac{h+s}{2^{h-1}}\,, \tag{5}$$

which implies that

$$C_{\text{rle}}(w,h) \;\le\; |B_h| \cdot \frac{h+s}{2^{h-1}} \;\le\; 2 \cdot C_{\text{rle}}(w,h)\,. \tag{6}$$

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

$$H_{\text{big}} = \left\{h \;:\; \frac{|B_h|}{n} \;\ge\; \frac{1}{2} \cdot \frac{\epsilon}{h_0} \cdot \frac{2^{h-1}}{h+s}\right\} \quad \text{and} \quad H_{\text{small}} = \left\{h \;:\; \frac{|B_h|}{n} \;<\; \frac{1}{2} \cdot \frac{\epsilon}{h_0} \cdot \frac{2^{h-1}}{h+s}\right\}. \tag{7}$$

Then we would like β_h to satisfy the following:

$$\begin{aligned} \frac{1}{3} \cdot \frac{|B_h|}{n} \le \quad \beta_h \le \tfrac{3}{2} \cdot \tfrac{|B_h|}{n} \quad & \text{if } h \in H_{\text{big}}; \\ 0 \le \quad \beta_h \le \tfrac{\epsilon}{h_0} \cdot \tfrac{2^{h-1}}{h+s} \quad & \text{otherwise } (h \in H_{\text{small}}). \end{aligned} \tag{8}$$

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

$$\begin{aligned} \widehat{C}_{\text{rle}} \;&=\; \sum_{h \in H_{\text{big}}} \beta_h \cdot n \cdot \frac{h+s}{2^{h-1}} + \sum_{h \in H_{\text{small}}} \beta_h \cdot n \cdot \frac{h+s}{2^{h-1}} && (9) \\ &\le\; \sum_{h \in H_{\text{big}}} \frac{3}{2} \cdot |B_h| \cdot \frac{h+s}{2^{h-1}} + h_0 \cdot \frac{\epsilon}{h_0} \cdot \frac{2^{h-1}}{h+s} \cdot n \cdot \frac{h+s}{2^{h-1}} && (10) \\ &\le\; \sum_{h \in H_{\text{big}}} 3 \cdot C_{\text{rle}}(w,h) + \epsilon n \;<\; 3 \cdot C_{\text{rle}}(w) + \epsilon n. && (11) \end{aligned}$$

The last inequality uses the upper bound from Equation (6). Similarly,

$$\begin{aligned} \widehat{C}_{\text{rle}} \;&\ge\; \sum_{h \in H_{\text{big}}} \beta_h \cdot n \cdot \frac{h+s}{2^{h-1}} && (12) \\ &\ge\; \frac{1}{3} \cdot \sum_{h \in H_{\text{big}}} C_{\text{rle}}(w) && (13) \\ &=\; \frac{1}{3} \cdot \left(C_{\text{rle}}(w) - \sum_{h \in H_{\text{small}}} C_{\text{rle}}(w,h) \right) && (14) \\ &>\; \frac{1}{3} \cdot C_{\text{rle}}(w) - \epsilon n && (15) \end{aligned}$$

Details of the algorithm and its analysis follow.

ALGORITHM II: A $(3,\epsilon)$-APPROXIMATION FOR $C_{\text{rle}}(w)$

1. Select $q = \Theta\left(\frac{\log(1/\epsilon)\cdot\log\log(1/\epsilon)}{\epsilon}\right)$ indices $t_1,\ldots,t_q$ uniformly and independently at random.
2. For $h = 1,\ldots,h_0$ 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 $\widehat{C}_{\text{rle}} = \sum_{h=1}^{h_0}\left(\frac{n}{q_h}\cdot\sum_{i=1}^{q_h} X_{h,i}\right)\cdot\frac{h+s}{2^{h-1}}$.

The query complexity. 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

$$O\left(\sum_{h=1}^{h_0} q_h\cdot 2^h\right) = O\left(q\cdot h_0^2\right) = O\left(\frac{\log^3(1/\epsilon)\cdot\log\log(1/\epsilon)}{\epsilon}\right). \tag{16}$$

Correctness. 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 Equation (8) 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,

$$\Pr\left[\left|\beta_h - \frac{|B_h|}{n}\right| \geq \frac{1}{2}\frac{|B_h|}{n}\right] < \exp\left(-c\cdot\frac{|B_h|}{n}\cdot q_h\right) \tag{17}$$

for some constant $c \in (0,1)$. Recall that $h_0 = O(\log(1/\epsilon))$ and that $q_h = \Theta(q\cdot\frac{h+s}{2^{h-1}}) = \Omega\big(\epsilon^{-1}\cdot h_0\cdot\log(h_0)\cdot\frac{h+s}{2^{h-1}}\big)$. Hence, for $h \in H_{\text{big}}$ (and for a sufficiently large constant in the $\Theta(\cdot)$ notation in the definition of q), the probability in Equation (17) is at most $\frac{1}{3}\cdot\frac{1}{h_0}$, and so Equation (8) holds with probability at least $1-\frac{1}{3}\cdot\frac{1}{h_0}$. On the other hand, for $h \in H_{\text{small}}$, the probability that $\beta_h \geq \frac{\epsilon}{h_0}\cdot\frac{2^{h-1}}{h+s}$ is bounded above by the probability of this event when $\frac{|B_h|}{n} = \frac{1}{2}\cdot\frac{\epsilon}{h_0}\cdot\frac{2^{h-1}}{h+s}$. By Equation (17) this is at most $\frac{1}{3}\cdot\frac{1}{h_0}$, and so in this case too Equation (8) holds with probability at least $1-\frac{1}{3}\cdot\frac{1}{h_0}$. By taking a union bound over all $h \in [h_0]$ the analysis is completed.

3.3 A 4-Multiplicative Estimate with $\tilde{O}(n/C_{\text{rle}}(w))$ 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 $C_{\text{rle}}(w) \geq \mu n$ for some $\mu > 0$. Then, by running Algorithm II (the $(3,\epsilon)$-approximation algorithm) with ϵ set to $\mu/2$, and outputting $\widehat{C}_{\text{rle}} + \epsilon n$, 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 $(3,\epsilon)$-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 $C_{\text{rle}}(w)$ (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 $C_{\text{rle}}(w)$

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

Correctness. For any given j, Algorithm II outputs $\widehat{C}^j_{\text{rle}} \in [\frac{1}{3}C_{\text{rle}}(w) - \epsilon_j n,\ 3C_{\text{rle}}(w) + \epsilon_j n]$, with probability at least $1 - \frac{1}{3} \cdot \delta_j$. Equivalently, $lb_j \leq C_{\text{rle}}(w) \leq ub_j$. By the Union bound, with probability at least $2/3$, $lb_j \leq C_{\text{rle}}(w) \leq ub_j$ 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 $C_{\text{rle}}(w)$. It is not hard to verify that once $\epsilon_j \leq \frac{C_{\text{rle}}(w)}{24n}$, then the algorithm indeed terminates.

Query complexity. The query complexity of the algorithm is dominated by the last iteration, when, as shown above, $\epsilon_j = \Theta(C_{\text{rle}}(w)/n)$. Since the query complexity of Algorithm II is $\tilde{O}(1/\epsilon)$, the query complexity of Algorithm III is $\tilde{O}(n/C_{\text{rle}}(w))$.

Improving the multiplicative approximation factor. The 4-multiplicative estimate of $C_{\text{rle}}(w)$ 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 $h = 1, \ldots, \log_{1+\frac{\gamma}{2}} \ell_0$ $(= O(\log(1/\epsilon)/\gamma))$, and setting $\epsilon = \gamma \cdot \mu/8$. The query complexity remains linear in $1/\mu = n/C_{\text{rle}}(w)$ (up to polylogarithmic factors), and is polynomial in $1/\gamma$.

The next theorem summarizes our positive results.

Theorem 3.1 *Let $w \in \Sigma^n$ be a string to which we are given query access.*

1. *Algorithm I is an ϵn-additive approximation algorithm for $C_{\text{rle}}(w)$ whose query and time complexity are $\tilde{O}(1/\epsilon^3)$.*
2. *Algorithm II $(3, \epsilon)$-estimates $C_{\text{rle}}(w)$ and has query and time complexity $\tilde{O}(1/\epsilon)$.*
3. *Algorithm III 4-estimates $C_{\text{rle}}(w)$ and has query and time complexity $\tilde{O}\left(\frac{n}{C_{\text{rle}}(w)}\right)$.*
4. *A generalization of Algorithm III $(1+\gamma)$-estimates $C_{\text{rle}}(w)$ and has query and time complexity $\tilde{O}\left(\frac{n}{C_{\text{rle}}(w)} \cdot \text{poly}(1/\gamma)\right)$.*

3.4 Lower Bounds

We start by giving a "needle-in-a-haystack" lower bound.

Theorem 3.2 *For every approximation factor $A > 1$, any A-approximation algorithm for the cost of RLE must perform $\Omega\left(\frac{n}{A^2 \log n}\right)$ queries.*

If $A \leq n^{\frac{1}{2}-\alpha}$ for some constant α, then $\Omega\left(\frac{n}{A^2}\right)$ queries are necessary. In particular, for constant A, $\Omega(n)$ queries are necessary in the worst case. More specifically, $\Omega\left(\frac{n}{C_{\mathrm{rle}}(w)}\right)$ queries are necessary for a constant factor approximation of $C_{\mathrm{rle}}(w)$.

By Theorem 3.2, the query complexity of our 4-approximation algorithm, Algorithm III, which is $\tilde{O}\left(\frac{n}{C_{\mathrm{rle}}(w)}\right)$, is tight up to logarithmic factors. Theorem 3.2 directly follows from the next lemma (by observing that $C_{\mathrm{rle}}(1^n) = \lceil \log(n+1) \rceil + 2$).

Lemma 3.3 *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) $C_{\mathrm{rle}}(w) \geq k \cdot \log\left(\frac{n}{2k} + 1\right)$ for every $w \in W_k$; (2) Distinguishing a uniformly selected string in W_k from the all-1 string 1^n requires $\Omega\left(\frac{n}{k}\right)$ queries.*

Proof: Let $\Sigma = \{0, 1\}$ and assume for simplicity that n is divisible by $2k-1$ (a slight modification deals with the case that n is not divisible by $2k-1$). Every string in W_k consists of $2k-1$ blocks, each of length $\frac{n}{2k-1}$. 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 runs of 1s, each of length at least $\frac{n}{2k-1}$ and therefore, $C_{\mathrm{rle}}(w) \geq k \cdot \log\left(\frac{n}{2k-1} + 1\right)$.

To distinguish 1^n from a string uniformly selected from W_k, any algorithm must perform $\Omega\left(\frac{n}{k}\right)$ queries. ■

We now turn to ϵn-additive estimates, which are desirable when $C_{\mathrm{rle}}(w)$ is relatively large (e.g., $\Omega(n)$) and ϵ is significantly smaller than $C_{\mathrm{rle}}(w)/n$.

Theorem 3.4 *For every $\epsilon > 0$, any ϵn-additive approximation algorithm requires $\Omega(1/\epsilon^2)$ queries for sufficiently large n.*

Proof: For any $p \in [0, 1]$ and sufficiently large n, let $\mathcal{D}_{n,p}$ be the following distribution over n-bit strings. First 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, output the unique binary string that starts with 0 and has this sequence of run lengths.[3]

Let w be a string generated according to $\mathcal{D}_{n,1/2}$ and w', a string generated according to $\mathcal{D}_{n,1/2+\epsilon}$. It is well known that $\Omega(1/\epsilon^2)$ independent coin flips are necessary to distinguish a coin with bias $\frac{1}{2}$ from a coin with bias $\frac{1}{2} + \epsilon$. Therefore, $\Omega(1/\epsilon^2)$ 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 $\Omega(\epsilon n)$. 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 got $\alpha \cdot \frac{n}{3}$

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

"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, $|\alpha - p| \leq \epsilon/4$ with probability at least $1 - 2\exp(-2(\epsilon/4)^2)$. With this probability, the encoding cost of the selected string is between $\frac{n}{3} \cdot \left(2 + p - \frac{\epsilon}{4}\right)$ and $\frac{n}{3} \cdot \left(2 + p + \frac{\epsilon}{4}\right)$. The theorem (for the case $n \mod 3 = 0$) follows, since with very high probability, $C_{\mathrm{rle}}(w') - C_{\mathrm{rle}}(w) = \Omega(\epsilon n)$.

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. ■

4 Lempel Ziv Compression

In this section we consider one of the variants of Lempel and Ziv's compression algorithm [ZL77], which we refer to as LZ77. Let $w \in \Sigma^n$ be a string over an alphabet Σ. Each symbol of the compressed representation of w, denoted $LZ(w)$, 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.

Specifically, in order to compress w, the algorithm works as follows. Starting from $t = 1$, at each step the algorithm finds the longest substring $w_t \dots w_{t+\ell-1}$ for which there exists an index $p < t$, such that $w_p, \dots, w_{p+\ell-1} = w_t, \dots, w_{t+\ell-1}$. If there is no such substring (that is, the character w_t has not appeared before) then the next symbol in $LZ(w)$ is w_t, and $t = t + 1$. Otherwise, the next symbol is (p, ℓ) and $t = t + \ell$. We refer to the substring $w_t \dots w_{t+\ell-1}$ (or w_t when w_t is a new character) as a *compressed segment*. We say that the symbol in $LZ(w)$ (i.e., w_t or (p, ℓ)) *represents* the compressed segment in w (i.e., w_t or $w_t \dots w_{t+\ell-1}$, respectively). Clearly, compression takes time $O(n^2)$, and decompression, time $O(n)$. In what follows we let $n_\ell(w)$ denote the number of compressed segments of length ℓ in w, not including the last compressed segment.

Let $C_{\mathrm{LZ77}}(w)$ denote the number of symbols in the compressed string $LZ(w)$. (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 problem reduces to estimating the number of distinct substrings in w of different lengths, which in turn directly reduces to estimating the number of distinct characters ("colors") 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.

4.1 Structural Lemmas

Here we state and prove two "structural" lemmas concerning the relation between $C_{\mathrm{LZ77}}(w)$ and the number of distinct substrings in w. We later use these lemmas to obtain an approximation algorithm for $C_{\mathrm{LZ77}}(w)$. Let $d_\ell(w)$ denote the number of distinct substrings of length ℓ in w. Unlike compressed segments in w, which are disjoint, the substrings counted above may overlap.

Lemma 4.1 *For every $\ell \in [n]$, $d_\ell(w) \leq C_{LZ77}(w) \cdot \ell$.*

Lemma 4.2 *Let $\ell_0 \in [n]$. Suppose that for some integer $m = m(\ell_0)$ 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 4.1. This proof is similar to the proof of a related lemma concerning grammars, which appears in [LS02]. 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 $LZ(w)$. We turn to the general case of $\ell > 1$.

Fix $\ell > 1$. Recall that $w_t \dots w_{t+k-1}$ of w is a *compressed segment* if it is represented by one symbol, w_t or (p, k), in $LZ(w)$. In particular, if the symbol is of the form (p, k) then $w_t, \dots, w_{t+k-1} = w_p, \dots, w_{p+k-1}$ for $p < t$. It follows that if a length-ℓ substring is contained within a compressed segment, then it has already appeared in w. Hence, the number of distinct length-ℓ substrings is bounded above by the number of length-ℓ substrings that start inside one compressed segment and end in another. 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 4.2. In what follows we use the shorthand n_ℓ for $n_\ell(w)$ and d_ℓ for $d_\ell(w)$. In order to prove the lemma we show that for every $1 \leq \ell \leq \lfloor \ell_0/2 \rfloor$,

$$\sum_{k=1}^{\ell} n_k \leq 2(m+1) \cdot \sum_{k=1}^{\ell} \frac{1}{k} \,. \tag{18}$$

Since the compressed segments in w are disjoint, we have that for every $\ell \geq 1$,

$$\sum_{k=\ell+1}^{n} n_k \leq \frac{n}{\ell+1} \,. \tag{19}$$

If we substitute $\ell = \lfloor \ell_0/2 \rfloor$ in Equations (18) and (19), and sum the two equations, we get that:

$$\sum_{k=1}^{n} n_k \leq 2(m+1) \cdot \sum_{k=1}^{\lfloor \ell_0/2 \rfloor} \frac{1}{k} + \frac{2n}{\ell_0} \leq 2(m+1)(\ln \ell_0 + 1) + \frac{2n}{\ell_0}. \tag{20}$$

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

We prove Equation (18) for every $1 \leq \ell \leq \lfloor \ell_0/2 \rfloor$ by induction on ℓ after proving the following claim.

Claim 4.3 *For every $1 \leq \ell \leq \lfloor \ell_0/2 \rfloor$,*

$$\sum_{k=1}^{\ell} k \cdot n_k \leq 2\ell(m+1) \,. \tag{21}$$

Proof: We show that each character of $w_\ell \dots w_{n-\ell}$ that appears in a compressed substring of length at most ℓ 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 characters of w contribute less than 2ℓ symbols. The claim follows.

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

Given the above, for each character w_j in the compressed substrings $w_t \dots w_{t+k-1}$ such that $\ell \leq j \leq n - \ell$, we map w_j to the length-2ℓ substring that ends at $w_{j+\ell}$. Therefore, each character in $w_\ell \dots w_{n-\ell}$ that appears in a compressed substring of length at most ℓ is mapped to a distinct length-2ℓ substring, as desired. ■ (Claim 4.3)

It remains to prove Equation (18) by induction on ℓ. Equation (21) 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 holds for every $j \in [\ell - 1]$. To prove it for ℓ, add Equation (21) to the sum of the induction hypothesis inequalities (Equation (18)) for every $j \in [\ell - 1]$. The left hand side of the resulting inequality is

$$\begin{aligned}
\sum_{k=1}^{\ell} k \cdot n_k + \sum_{j=1}^{\ell-1} \sum_{k=1}^{j} n_k &= \sum_{k=1}^{\ell} k \cdot n_k + \sum_{k=1}^{\ell-1} \sum_{j=1}^{\ell-k} n_k && (22)\\
&= \sum_{k=1}^{\ell} k \cdot n_k + \sum_{k=1}^{\ell-1} (\ell - k) \cdot n_k && (23)\\
&= \ell \cdot \sum_{k=1}^{\ell} n_k \,. && (24)
\end{aligned}$$

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

$$\begin{aligned}
\ell + \sum_{j=1}^{\ell-1} \sum_{k=1}^{j} \frac{1}{k} &= \ell + \sum_{k=1}^{\ell-1} \sum_{j=1}^{\ell-k} \frac{1}{k} && (25)\\
&= \ell + \sum_{k=1}^{\ell-1} \frac{\ell - k}{k} && (26)\\
&= \ell + \ell \cdot \sum_{k=1}^{\ell-1} \frac{1}{k} - (\ell - 1) && (27)\\
&= \ell \cdot \sum_{k=1}^{\ell} \frac{1}{k} \,. && (28)
\end{aligned}$$

Dividing both sides by ℓ gives the inequality in Equation (18). ■ (Lemma 4.2)

4.1.1 Tightness of Lemma 4.2

The following lemma shows that Lemma 4.2 is asymptotically tight.

Lemma 4.4 *For all positive integers ℓ_0 and 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 '$1 \dots m$'. 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 '$11\ 22\ 33 \dots mm$', phase 3 the string '$122\ 344\ 566 \dots m$', phase 4 the string '$1233\ 4566\ 7899 \dots m$', 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 $m(1+1/2+1/3+...+1/\ell_0) = \Omega(m \log \ell_0) = \Omega(m \log \ell_0 + n/\ell_0)$. 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 4.4 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 $\Theta(m^2 \log m)$ 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 Colors (Definition 1.1) as a subroutine. The main tool in the reduction from Colors to LZ77 is the structural lemmas 4.1 and 4.2, summarized in the following corollary.

Corollary 4.5 *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 Colors from Lemma 5.6, described later in the paper, as a subroutine. Recall that an algorithm for Colors is required to approximate the number of distinct colors in an input string, where the ith character represents the ith color. We denote the number of colors in an input string τ by $C_{\mathrm{COL}}(\tau)$. To approximate d_ℓ, the number of distinct length-ℓ substrings in w, using an algorithm for Colors, view each length-ℓ substring as a separate color. Each query of the algorithm for Colors can be implemented by ℓ queries to w.

Let $\textsc{Estimate}(\ell, B, \delta)$ 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 Colors, 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 5.6, 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 $C_{\text{LZ77}}(w)$

1. Set $\ell_0 = \left\lceil \frac{2}{A\epsilon} \right\rceil$ and $B = \frac{A}{2\sqrt{\log(2/(A\epsilon))}}$.
2. For all ℓ in $[\ell_0]$, let $\hat{d}_\ell = \text{ESTIMATE}(\ell, B, \frac{1}{3\ell_0})$.
3. Combine the estimates to get an approximation of m from Corollary 4.5: set $\hat{m} = \max_\ell \frac{\hat{d}_\ell}{\ell}$.
4. Output $\widehat{C}_{\text{LZ77}} = \hat{m} \cdot \frac{A}{B} + \epsilon n$.

Theorem 4.6 *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 $\tilde{O}\left(\frac{n}{A^3\epsilon}\right)$.*

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 4.5, which implies that

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

Equivalently, $\dfrac{C_{\text{LZ77}} - 4(n/\ell_0)}{4B \log \ell_0} \leq \hat{m} \leq B \cdot C_{\text{LZ77}}$. 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 $\widehat{C}_{\text{LZ77}}$ is indeed an (A,ϵ)-estimate for C_{LZ77}.

As we explained before presenting the algorithm, 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 requested to query only the length-ℓ prefixes of the length-ℓ_0 substrings queried in the first call. With this implementation, the query complexity is $O\left(\frac{n}{B^2}\ell_0 \log \ell_0\right) = O\left(\frac{n}{A^3\epsilon^2} \log^2 \frac{1}{A\epsilon}\right)$. 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 Reducing Colors to LZ77

The previous subsection demonstrates that estimating the LZ77 compressibility of a string reduces to Colors. As we show later, the Colors problem is quite hard, and it is not possible to improve much

on the simple approximation algorithm in Section 5, 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 Colors? As we shall see, it is not much easier in general.

Lemma 4.7 (Reduction from Colors to LZ77) *Suppose there is an algorithm $\mathcal{A}_{LZ}$ 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 Colors taking inputs of length $n' = \Theta(\alpha n)$ that performs q queries and, with probability at least $2/3$, distinguishes inputs with at most $\alpha' n'$ colors from those with at least $\beta' n'$ colors, for $\alpha' = \alpha/2$ 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 Colors algorithm obtained in Lemma 4.7, is larger than the gap between the parameters α and β for which the LZ-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 LZ-compressibility problem and, in particular, of the length of the input strings, n. Hence, when writing q as a function of the parameters of the Colors problem 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.

Before giving the proof of the lemma, we discuss its implications. Theorem 6.1 gives a strong lower bound on the sample complexity of approximation algorithms for Colors. An interesting special case is that a subpolynomial-factor approximation for Colors requires many queries even with a promise that the strings are only slightly compressible: for any $B = n^{o(1)}$, there exists a constant $\beta \in (0, 1)$ such that distinguishing inputs with βn colors from those with $\beta n/B$ colors requires $n^{1-o(1)}$ queries. The lemma above extends that bound to estimating LZ compressibility: The lemma above extends that bound to estimating compressibility via Lempel-Ziv:

> *For any $B = n^{o(1)}$, and any alphabet Σ, distinguishing strings with LZ compression cost $\tilde{\Omega}(n)$ from strings with cost $\tilde{O}(n/B)$ requires $n^{1-o(1)}$ queries.*

Theorem 6.1 applies to a broad range of parameters, and yields the following general statement when combined with Lemma 4.7:

Corollary 4.8 (LZ is Hard to Approximate with Few Samples, General Statement) *For sufficiently large n, for all alphabets Σ and for every $B \in (1, n^{1/4}/(4 \log n^{3/2}))$, 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 Reduction from Colors (Lemma 4.7). Suppose we have an algorithm $\mathcal{A}_{\mathrm{LZ}}$ for LZ-compressibility as specified in the premise of Lemma 4.7. In what follows we show how to transform a Colors instance τ into an input for $\mathcal{A}_{\mathrm{LZ}}$, and use the output of $\mathcal{A}_{\mathrm{LZ}}$ in order to distinguish between the case that τ contains at most $\alpha' n'$ colors, and the case that τ contains more than $\beta' n'$ colors, where β'/α' 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 Colors, we transformed substrings into colors. Here we perform the reverse operation.

Given a Colors 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 $\mathcal{A}_{\mathrm{LZ}}$ on w to obtain information about τ. We begin by replacing each color 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 $\alpha' n'$ colors, then $C_{\mathrm{LZ77}}(w) \leq 2\alpha' n$;
2. If τ has $\beta' n'$ or more colors, then $C_{\mathrm{LZ77}}(w) \geq \frac{1}{2} \cdot \min\left\{1, \frac{\log|\Sigma|}{4 \log n'}\right\} \cdot \beta' n$ with probability at least $\frac{7}{8}$ over the choice of w.

That is, in the first case we get an input w for Colors such that $C_{\mathrm{LZ77}}(w) \leq \alpha n$ for $\alpha = 2\alpha'$, and in the second case, with probability at least 7/8, $C_{\mathrm{LZ77}}(w) \geq \beta n$ 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 between the case that $C_{\mathrm{COL}}(\tau) \leq \alpha' n'$ and the case that $C_{\mathrm{COL}}(\tau) > \beta' n'$, we can run $\mathcal{A}_{\mathrm{LZ}}$ on w and output its answer. Taking into account the failure probability of $\mathcal{A}_{\mathrm{LZ}}$ and the failure probability in Item 2 above, the Lemma follows.

We prove these two claims momentarily, but first observe that in order to run the algorithm $\mathcal{A}_{\mathrm{LZ}}$, there is no need to generate the whole string w. Rather, upon each query of $\mathcal{A}_{\mathrm{LZ}}$ to w, if the index of the query belongs to a block that has already been generated, the answer to $\mathcal{A}_{\mathrm{LZ}}$ is determined. Otherwise, we query the element (color) in τ that corresponds to the block. If this color was not yet observed, then we set the block to a uniformly selected substring in Σ^k. If this color was already observed in τ, then we set the block according to the substring that was already selected for the color. 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 $\alpha' n'$ colors then w contains at most $\alpha' n'$ distinct blocks. Since each block is of length k, at most k compressed segments start in each new block. By definition of LZ77, at most one compressed segment 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 colors, then 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

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

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

5 The Colors and the Distribution Support Size Problems

This section deals with algorithms for Colors. Recall that in that problem, given random access to n colored balls, we would like to approximate the number of distinct colors. In Subsection 5.1 we explain why it is enough to consider algorithms for Colors that sample uniformly at random with replacement, define the Distribution Support Size Problem and point out how it is related to Colors. Subsection 5.2 describes a simple algorithm for Colors and Distribution Support Size. The Guaranteed-Error estimator of Charikar *et al.* has the same guarantees as our approximation algorithm. But since our algorithm is so simple, we present it here for completeness. Also for completeness, in Subsection 5.3 we give the needle-in-a-haystack lower bound for Colors due to Charikar *et al.* and Bar-Yossef *et al.*

5.1 Algorithms with Uniform Samples, and the Distribution Support Problem

In general, an algorithm for Colors is allowed to make arbitrary (adaptive) queries to the input. In this subsection, we show that restricted algorithms that take samples uniformly at random with replacement are essentially as good for Colors as general algorithms. Then we define the Distribution Support Problem, which is a slight generalization of Colors when algorithms are restricted to uniform samples with replacement.

First, consider algorithms that take their samples uniformly at random *without replacement* from $[n]$. The following lemma, appearing in Bar-Yossef's thesis [Bar02, Page 88], shows that such algorithms are essentially as good for solving Colors as general algorithms.

Lemma 5.1 ([Bar02]) *For any function invariant under permutations of input elements (ball positions), any algorithm that makes s queries can be simulated by an algorithm that takes s samples uniformly at random* without replacement *and has the same guarantees on the output as the original algorithm.*

The main idea in the proof of the lemma is that the new algorithm, given input w, can simulate the old algorithm on $\pi(w)$, where π is a random permutation of the input, dictated by the random samples chosen by the new algorithm. Since the value of the function (in our case, the number of colors) is the same for w and $\pi(w)$, the guarantees on the old algorithm hold for the new one.

Next, we would like to go from the algorithms that sample uniformly *without replacement* to the ones that sample uniformly *with replacement* and find out the corresponding color, but not the input position that was queried. Bar-Yossef proved that for all functions invariant under permutations, algorithms that take $O(\sqrt{n})$ uniform samples *without replacement* can be simulated by algorithms that take the same number of samples *with replacement*. The idea is that with so few samples, an

algorithm sampling *with replacement* is likely to never look at the same input position twice. To prove a statement along the same lines for algorithms that take more samples, Bar-Yossef allows them to see not only the color of each sample, but also which input position was queried. It is possible to avoid giving this extra information to an algorithm for Colors, with a slight loss in the approximation factor.

Definition 5.2 *We call an algorithm* uniform *if it takes independent samples* with replacement *and only gets to see the colors of the samples, but not the input positions corresponding to them.*

Lemma 5.3 *Let $B = B(n)$, such that $\sqrt{0.1} \cdot B \geq 1$. For every $(\sqrt{0.1} \cdot B)$-approximation algorithm $\mathcal{A}$ for Colors that makes s queries and works with probability $\geq \frac{11}{12}$, there is a B-approximation* uniform *algorithm $\mathcal{A}'$ that takes s samples and works with probability $\geq \frac{2}{3}$.*

Proof: Conduct the following mental experiment: let algorithm $\mathcal{A}'$ generate an instance of Colors by taking n uniform samples from its input and recording their colors. If there are $C = C(n)$ colors in the input of $\mathcal{A}'$, the generated instance will have at most C colors. However, some of the colors might be missing. We will show later (see Claim 5.4 with s set to n) that with probability $\geq \frac{3}{4}$ at least $0.1 \cdot C$ colors appear in the instance. That is, with probability $\geq \frac{3}{4}$, the instance generated in our mental experiment will have between $0.1 \cdot C$ and C colors. When $\mathcal{A}$ is run on that instance, with probability $\geq \frac{11}{12}$, it will output an answer between $\frac{0.1 \cdot C}{\sqrt{0.1} \cdot B} = \sqrt{0.1} \cdot \frac{C}{B}$ and $\sqrt{0.1} \cdot B \cdot C$. Thus, if $\mathcal{A}'$ runs $\mathcal{A}$ on this instance and multiplies its answer by $\sqrt{10}$, it will get a B-multiplicative approximation to C with probability $\geq 1 - \frac{1}{4} - \frac{1}{12} \geq \frac{2}{3}$, as promised. The final observation is that since each color in the instance is generated independently, $\mathcal{A}'$ can run $\mathcal{A}$ on that instance, generating colors on demand, resulting in s samples instead of n. ■

The next claim is more general than required for the proof of the preceding lemma because it will be also used in the analysis of our algorithm for Colors. The constants in the lemma can be improved with a more elaborate argument.

Claim 5.4 *Let $s = s(n) \leq n$. Then s independent samples from a distribution with $C = C(n)$ elements, where each element has probability $\geq \frac{1}{n}$, yield at least $\frac{Cs}{10n}$ distinct elements, with probability $\geq \frac{3}{4}$.*

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,

$$\mathsf{E}[X] = \sum_{i=1}^{C} \mathsf{E}[X_i] \geq C\left(1 - \left(1 - \frac{1}{n}\right)^s\right) \geq C\left(1 - e^{-(s/n)}\right) \geq (1 - e^{-1})\frac{Cs}{n}. \tag{29}$$

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. Since X is the sum of Bernoulli variables, $\mathsf{Var}[X] \leq \mathsf{E}[X]$. For any fixed δ,

$$\Pr\left[X \leq \delta\,\mathsf{E}[X]\right] \leq \Pr\left[|X - \mathsf{E}[X]| \geq (1 - \delta)\,\mathsf{E}[X]\right] \leq \frac{\mathsf{Var}[X]}{((1 - \delta)\,\mathsf{E}[X])^2} \leq \frac{1}{(1 - \delta)^2\,\mathsf{E}[X]}. \tag{30}$$

Set $\delta = 3-\sqrt{8}$. If $\mathsf{E}[X] \geq \frac{4}{(1-\delta)^2}$, then by Equations (30) and (29), 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 (29) 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. ■

Now we define the Distribution Support Problem, which is a slight generalization of Colors with uniform algorithms.

Definition 5.5 (Distribution Support Problem) *Given access to independent samples from a distribution where each element appears with probability at least $\frac{1}{n}$, determine (or approximate) the size of the distribution support.*

Clearly, Colors Problem with uniform algorithms (with replacement) is a special case of the Distribution Support: each color can be viewed as an element in the support of the distribution. A color that appears a times in the input will be generated with probability $\frac{a}{n}$ under uniform sampling with replacement. Since a has to be integer in the Colors Problem, while it can be any number greater or equal to 1 in the Distribution Support Problem, the latter problem is more general. We state our algorithms for Distribution Support and our lower bound for Colors, so that they apply to both problems.

5.2 A Simple Algorithm for Distribution Support and Colors

In this subsection, we give a simple algorithm for Distribution Support and Colors. Recall that we proved that without loss of generality we can consider only uniform algorithms for Colors, and that Colors with uniform algorithms is a special case of Distribution Support.

ALGORITHM V: AN A-APPROXIMATION FOR DISTRIBUTION SUPPORT

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

Lemma 5.6 *Let $A = A(n)$. Algorithm V is an A-approximation algorithm for Distribution Support whose query complexity and running time are $O\left(\frac{n}{A^2}\right)$.*

Proof: Let C be the number of elements in the support of the distribution. We need to show that $\frac{C}{A} \leq \widehat{C} \cdot A \leq C \cdot A$, or equivalently, $\frac{C}{A^2} \leq \widehat{C} \leq C$, with probability at least $\frac{2}{3}$. The sample always contains at most as many elements as are in the support: $\widehat{C} \leq C$. And Claim 5.4, applied with $s = \frac{10n}{A^2}$, shows that $\widehat{C} \geq \frac{C}{A^2}$ with probability $\geq \frac{2}{3}$. To get the running time $O\left(\frac{n}{A^2}\right)$ one can use a random 2-universal hash function. ■

5.3 A "Needle in a Haystack" Lower Bound for Colors

Lemma 5.7 ([BKS01, CCMN00]) *Any A-approximation algorithm for Colors has to make $\Omega\left(\frac{n}{A^2}\right)$ queries.*

Proof: Any deterministic algorithm needs $\Omega\left(\frac{n}{A^2}\right)$ queries to distinguish between a string with one color and the same string with A^2 unique colors inserted in random positions. The lemma follows by Yao's Principle [Yao77]. ■

6 The Main Lower Bound for Colors

This section is devoted to the lower bound for Colors, that applies to inputs with many colors, and is formalized in the following theorem:

Theorem 6.1 *For every sufficiently large n and for every $B \in (1, n^{1/4}/\sqrt{\log n})$, there exist $d_1 = \Theta(1)$ and $d_2 \geq B \cdot d_1$ such that the following holds for $k = k(n, B) = \left\lfloor \sqrt{\frac{\log n}{\log B + \frac{1}{2}\log\log n}} \right\rfloor$. Every algorithm for Colors needs to perform $\Omega\left(n^{1-\frac{2}{k}}\right)$ queries to distinguish between inputs with at least $\frac{n}{d_1}$ colors and inputs with at most $\frac{n}{d_2}$ colors.*

To illustrate the lower bound in Theorem 6.1 as a function of the multiplicative factor B, we look at two cases. If $B = n^{o(1)}$ then the lower bound is $n^{1-\eta(n)}$ for $\eta(n) = o(1)$. In particular, for constant B we get $\eta(n) = \Theta\left(\sqrt{\frac{\log\log n}{\log n}}\right)$. If $B = n^{\beta}$ for constant β, then the lower bound is $n^{1-2\sqrt{\beta}(1+o(1))}$. Unlike in the "needle-in-a-haystack" lower bound in Lemma 5.7 which is based on the difficulty of distinguishing strings with $B = A^2$ colors from strings with a single color, this bound is based on the difficulty of distinguishing strings with at least $\frac{n}{d}$ colors, for constant d, from strings with less than $\frac{n}{B \cdot d}$ colors.

6.1 Main Building Blocks for the Proof of Theorem 6.1

We prove Theorem 6.1 by constructing, for each B, two inputs as specified by the theorem (or, more precisely, two distributions on inputs), such that any algorithm making $o\left(n^{1-\frac{2}{k}}\right)$ queries gets statistically indistinguishable answers. Recall that we may assume without loss of generality that the algorithm is uniform. The next definition is central to our analysis.

Definition 6.2 (Collisions and Histograms) *Consider s samples taken by a uniform algorithm. An ℓ-way collision occurs if a color appears exactly ℓ times in the sample. We denote by F_ℓ, for $\ell = 0, 1, \cdots, s$, the number of ℓ-way collisions in the sample. The histogram F of the sample is the vector $(F_1, \cdots, F_s)$, indicating for each non-zero ℓ how many colors appear exactly ℓ times in the sample.*

Intuitively, the histogram is the only piece of information that an algorithm can use when trying to estimate the number of colors. We will formalize this in Subsection 6.4. Our goal is to define two instances of the Colors problem (or, more precisely, two distributions on instances) that contain a significantly different number of colors, but for which the corresponding distributions on histograms that a uniform algorithm sees are close. For this we would like to ensure that, for all ℓ, the expected number of ℓ-way collisions is very similar under both distributions on samples. A priori it is not clear why making all expectations very similar will ensure that the distributions on histograms are close in statistical difference. We establish this later.

Instead of working directly with ℓ-way collisions, we consider the following related notion. A *monochromatic ℓ-tuple* is a set of ℓ samples that have the same color. Observe that the number of ℓ-way collisions can be calculated by the Inclusion-Exclusion Principle from the number of monochromatic ℓ'-tuples for $\ell' \geq \ell$. To ensure that the expected number of ℓ-way collisions is the same when sampling from two different Colors instances, it is enough to make the expected number of monochromatic ℓ-tuples the same, for all ℓ. Further below, we translate this requirement into conditions on the moments of a pair of integer distributions.

The next definition relates distributions on integers and distributions on strings (instances of Colors). We later show the connection between the moments of the former and the expected number of monochromatic tuples in a sample from the latter.

Given a distribution X on integers, we will produce distribution D_{X} over strings with roughly $\frac{n}{\mathsf{E}[\mathsf{X}]}$ distinct colors. Suppose X takes on k distinct integer values $a_0, ..., a_{k-1}$ with probabilities $p_0, ..., p_{k-1}$, respectively. The colors in each string in D_{X} are partitioned into *types* 0 through $k-1$. Each color of type i appears a_i times and, for each i, there are roughly $\frac{np_i}{\mathsf{E}[\mathsf{X}]}$ colors of type i. If $\frac{np_i}{\mathsf{E}[\mathsf{X}]}$ is always an integer, then this accounts for all n symbols in the string. In general, we need to account for rounding errors. This leads to the following, more exact, definition:

Definition 6.3 (The Distribution D_{X}) *Let $a_0 < a_1 < \ldots < a_{k-1}$ be $k > 1$ integers, and let X be a random variable defined over these integers where $\Pr[\mathsf{X} = a_i] = p_i$, so that in particular $\mathsf{E}[\mathsf{X}] = \sum_{i=0}^{k-1} p_i \cdot a_i$. Based on X, we define a distribution D_{X} over strings in $[n]^n$ that contain M_{X} colors, where $M_{\mathsf{X}} = \sum_{i=0}^{k-1} \left\lfloor \frac{np_i}{\mathsf{E}[\mathsf{X}]} \right\rfloor + n - \sum_{i=0}^{k-1} \left\lfloor \frac{np_i}{\mathsf{E}[\mathsf{X}]} \right\rfloor \cdot a_i$. (Note that if $\frac{np_i}{\mathsf{E}[\mathsf{X}]}$ is an integer for every i then $M_{\mathsf{X}} = \frac{n}{\mathsf{E}[\mathsf{X}]}$.) For $i = 0, \ldots, k-1$, every string in the support of D_{X} contains $\left\lfloor \frac{np_i}{\mathsf{E}[\mathsf{X}]} \right\rfloor$ colors of* type *i, where each color of type i appears a_i times. In addition, there are $n - \sum_{i=0}^{k-1} \left\lfloor \frac{np_i}{\mathsf{E}[\mathsf{X}]} \right\rfloor \cdot a_i$ colors that appear once each.*

To select a string according to D_{X}, uniformly select M_{X} colors in $[n]$, partition the colors into types as defined above (which determines how many times each color appears), and then randomly permute all colors (symbols) in the string.

Suppose that an algorithm takes s uniform samples with replacement from an instance in the support of D_{X}. Assume for simplicity that $\frac{np_i}{\mathsf{E}[\mathsf{X}]}$ is an integer for every i (so $M_{\mathsf{X}} = \frac{n}{\mathsf{E}[\mathsf{X}]}$). Since there are $p_i \frac{n}{\mathsf{E}[\mathsf{X}]}$ colors of type i and each has probability mass $\frac{a_i}{n}$, the probability that a particular ℓ-tuple is monochromatic is $\sum_i p_i \frac{n}{\mathsf{E}[\mathsf{X}]} \left(\frac{a_i}{n}\right)^\ell$. The expected number of monochromatic ℓ-tuples in s samples is

$$\binom{s}{\ell} \cdot \sum_i p_i \frac{n}{\mathsf{E}[\mathsf{X}]} \cdot \left(\frac{a_i}{n}\right)^\ell = \binom{s}{\ell} \cdot \frac{1}{n^{\ell-1}} \cdot \frac{1}{\mathsf{E}[\mathsf{X}]} \cdot \sum_i p_i a_i^\ell = \binom{s}{\ell} \cdot \frac{1}{n^{\ell-1}} \cdot \frac{\mathsf{E}[\mathsf{X}^\ell]}{\mathsf{E}[\mathsf{X}]} . \tag{31}$$

The last equality follows from the definition of X. We would like to make this expression the same for two different distributions that contain a different number of colors. Consider a (uniform) algorithm that takes $s = o\left(n^{1-\frac{1}{k}}/a_{k-1}\right)$ samples, where a_{k-1} comes from the definition of D_{X}. For such an algorithm, whenever $\ell \geq k$, the expected number of monochromatic ℓ-tuples is $o(1)$. For the remaining $\ell < k$, we would like to make the expected number of monochromatic ℓ-tuples the same for two distributions on instances, $D_{\hat{\mathsf{X}}}$ and $D_{\tilde{\mathsf{X}}}$, that differ by a factor of at least B in the

number of colors they contain. This leads to the following conditions on pairs of distributions over integers, which are the core of our lower bound.

Theorem 6.4 (Moments conditions) *For all integers $k > 1$ and $B > 1$, there exist two random variables $\hat{\mathsf{X}}$ and $\tilde{\mathsf{X}}$ over positive integers $a_0 < a_1 < \cdots < a_{k-1}$ satisfying*

$$\frac{\mathsf{E}[\tilde{\mathsf{X}}]}{\mathsf{E}[\hat{\mathsf{X}}]} \geq B \quad \textit{and} \quad \frac{\mathsf{E}[\tilde{\mathsf{X}}^\ell]}{\mathsf{E}[\hat{\mathsf{X}}^\ell]} = \frac{\mathsf{E}[\tilde{\mathsf{X}}]}{\mathsf{E}[\hat{\mathsf{X}}]} \ \textit{ for } \ \ell = 2, \ldots, k-1 \ . \tag{32}$$

In particular, such random variables exist for $a_0, \ldots, a_{k-1}$ that satisfy $a_i = (Bk)^i$.

In order to reduce notation, all variables pertaining to the first distribution are marked by a hat ($\hat{}$) and those pertaining to the second, by a tilde ($\tilde{}$). Whenever we make definitions or prove statements relevant to both distributions, the corresponding variables without hat or tilde are used. We prove Theorem 6.4 in Subsection 6.3. Our next main building block in the proof of Theorem 6.1, is the lemma stated below. It shows (roughly) that if two distributions $\hat{\mathsf{X}}, \tilde{\mathsf{X}}$ over integers obey the conditions from Theorem 6.4 then the corresponding distributions on strings, $D_{\hat{\mathsf{X}}}$ and $D_{\tilde{\mathsf{X}}}$, cannot be distinguished by a uniform algorithm using, roughly, $o(n^{1-\frac{1}{k}}/a_{k-1})$ samples. In fact, the bound is a bit more complicated, since it depends on how the maximum value, a_{k-1}, in the support of $\hat{\mathsf{X}}$ and $\tilde{\mathsf{X}}$ varies as n increases.

Lemma 6.5 (Distinguishability by Uniform Algorithms) *For any* uniform *algorithm $\mathcal{A}$ that takes $s \leq \frac{n}{4 \cdot a_{k-1}}$ samples,*

$$\left|\Pr[\mathcal{A}(D_{\hat{\mathsf{X}}}) = 1] - \Pr[\mathcal{A}(D_{\tilde{\mathsf{X}}}) = 1]\right| \ = \ O\left(k^2 \left(\frac{a_{k-1}}{n} \cdot s\right)^{2/3} + \frac{k}{\lfloor \frac{k}{2} \rfloor ! \cdot \lceil \frac{k}{2} \rceil !} \cdot \left(\frac{a_{k-1}}{n}\right)^{k-1} \cdot (2s)^k\right) \ .$$

The proof of Lemma 6.5 appears in Subsection 6.4. We are now ready to prove the main lower bound (Theorem 6.1).

6.2 Proof of Theorem 6.1

In this section we prove the main lower bound (Theorem 6.1) by combining the construction of distributions satisfying the moments condition (Theorem 6.4) with the bound on distinguishability by uniform algorithms (Lemma 6.5).

Let $\hat{\mathsf{X}}$ and $\tilde{\mathsf{X}}$ obey the conditions in Theorem 6.4, and let $D_{\hat{\mathsf{X}}}$ and $D_{\tilde{\mathsf{X}}}$ be as determined in Definition 6.3 based on $\hat{\mathsf{X}}$ and $\tilde{\mathsf{X}}$, respectively. Consider any uniform algorithm $\mathcal{A}$ that takes $\frac{s}{2}$ samples (where the choice of $\frac{s}{2}$ rather than s samples is made for the convenience of the analysis). According to Lemma 6.5,

$$\left|\Pr[\mathcal{A}(D_{\hat{\mathsf{X}}}) = 1] - \Pr[\mathcal{A}(D_{\tilde{\mathsf{X}}}) = 1]\right| = O\left(k^2 \left(\frac{a_{k-1}}{n} \cdot s\right)^{2/3} + \frac{k}{\lfloor \frac{k}{2} \rfloor ! \cdot \lceil \frac{k}{2} \rceil !} \cdot \left(\frac{a_{k-1}}{n}\right)^{k-1} \cdot s^k\right). \tag{33}$$

Recall that Theorem 6.4 stated that there exist $\hat{\mathsf{X}}$ and $\tilde{\mathsf{X}}$ such that $a_{k-1} = (Bk)^{k-1} < (BK)^k$, so we may assume that this is in fact the case. Therefore,

$$\left|\Pr[\mathcal{A}(D_{\hat{\mathbf{X}}})=1]-\Pr[\mathcal{A}(D_{\tilde{\mathbf{X}}})=1]\right| = O\left(k^2\left(\frac{(Bk)^k\cdot s}{n}\right)^{2/3}+\frac{k}{\lfloor\frac{k}{2}\rfloor!\cdot\lceil\frac{k}{2}\rceil!}\cdot\frac{(Bk)^{k(k-1)}\cdot s^k}{n^{k-1}}\right). \quad (34)$$

We set k and s as functions of B so that the error term in Equation (34) is bounded from above by $o(1)$. Given B, we define $q=q(n,B)$ by the equality $B=\log(n)^q$. Set $k=\left\lfloor\sqrt{\frac{\log(n)}{(q+\frac{1}{2})\log\log(n)}}\right\rfloor$, and $s=\left\lfloor n^{1-\frac{2}{k}}\right\rfloor$. To ensure $s\geq 1$, we need $k>2$, so we restrict q to be $0<q<\frac{\log n}{4\log\log n}-\frac{1}{2}$. In particular, B is bounded from above by $n^{\frac{1}{4}}/\sqrt{\log n}$. To make the calculations easier assume that $n>16$, so that $k<\sqrt{\log n}$. We handle the two summands in Equation(34) separately. We begin with the first summand.

$$\begin{aligned}
k^2\left(\frac{(Bk)^k s}{n}\right)^{2/3} &< \log(n)\left(\frac{\left(\log(n)^{q+\frac{1}{2}}\right)^k n^{1-\frac{2}{k}}}{n}\right)^{2/3} && (35)\\
&= \log(n)\left(\frac{\left(\log(n)^{q+\frac{1}{2}}\right)^k}{n^{\frac{2}{k}}}\right)^{2/3} && (36)\\
&\leq \log(n)\left(\frac{2^{\sqrt{(q+\frac{1}{2})\log\log(n)\log(n)}}}{2^{2\sqrt{(q+\frac{1}{2})\log\log(n)\log(n)}}}\right)^{2/3} && (37)\\
&< 2^{-\frac{1}{3}\sqrt{\log\log(n)\log(n)}}. && (38)
\end{aligned}$$

Equation (38) follows since $q>0$. We estimate the second summand in a similar fashion.

$$\begin{aligned}
\frac{k}{\lfloor\frac{k}{2}\rfloor!\cdot\lceil\frac{k}{2}\rceil!}\cdot\frac{(Bk)^{k(k-1)}s^k}{n^{k-1}} &= \frac{k}{(Bk)^k\,\lfloor\frac{k}{2}\rfloor!\cdot\lceil\frac{k}{2}\rceil!}\cdot\frac{(Bk)^{k^2}s^k}{n^{k-1}} && (39)\\
&< \frac{2}{(Bk)^k}\cdot\frac{(\log(n)^{q+\frac{1}{2}})^{k^2}n^{k-2}}{n^{k-1}} && (40)\\
&= \frac{2}{(Bk)^k}\cdot\frac{\left(\log(n)^{q+\frac{1}{2}}\right)^{\frac{\log n}{(q+\frac{1}{2})\log\log n}}}{n} && (41)\\
&= \frac{2}{(Bk)^k} && (42)\\
&< 2^{-\frac{1}{2}\sqrt{\log\log(n)\log(n)}}. && (43)
\end{aligned}$$

The last inequality follows by a careful analysis that is omitted here. Combining Equations (34), (38) and (43) we get that

$$\left|\Pr[\mathcal{A}(D_{\hat{\mathbf{X}}})=1]-\Pr[\mathcal{A}(D_{\tilde{\mathbf{X}}})=1]\right| = O\left(2^{-\frac{1}{3}\sqrt{\log\log(n)\log(n)}}\right). \quad (44)$$

This completes the proof of Theorem 6.1.

6.3 Constructing the Distributions: Proof of Theorem 6.4

We first give the intuition behind our construction. Recall that the support of $\hat{\mathsf{X}}$ and $\tilde{\mathsf{X}}$ is contained in the set $\{a_0, \ldots, a_{k-1}\}$. Let $\hat{\mathsf{p}}_i = \Pr[\hat{\mathsf{X}} = a_i]$ and $\tilde{\mathsf{p}}_i = \Pr[\tilde{\mathsf{X}} = a_i]$. Denote with V the $(k-1) \times k$ Vandermonde matrix satisfying $V_{i,j} = (a_j)^i$. Then the theorem can be restated in the following way: For every k and B there exists such a matrix V, two probability distributions $\vec{\hat{p}}, \vec{\tilde{p}}$ on $\{a_0, \ldots, a_{k-1}\}$ and some $C \geq B$ such that $V(C \cdot \vec{\hat{p}} - \vec{\tilde{p}}) = \vec{0}$. Thus, to prove the theorem, we find a Vandermonde matrix V and a vector $\vec{u}$ such that $V\vec{u} = \vec{0}$, and from u we extract two vectors of distributions $\vec{\hat{p}}$ and $\vec{\tilde{p}}$.

In our construction we take V to be the Vandermonde matrix corresponding to the points $a_j = a^j$ for $j = 0, \ldots, k-1$ and some a that we later define. Our vector $\vec{u}$ corresponds to the coefficients of the (unique) non-zero monic polynomial f of degree $k-1$ that vanishes on $a, a^2, \ldots, a^{k-1}$. Notice that $V\vec{u} = \vec{0}$. The vector $\vec{\hat{p}}$ is constructed from the positive coefficients of f (after normalization), and $\vec{\tilde{p}}$ comes from the negative coefficients of f (after normalization). Because the set of zeros of f is a geometric sequence, it turns out that the coefficients of f also grow rapidly, and this property enables us to bound the ratio between $\mathsf{E}[\tilde{\mathsf{X}}]$ and $\mathsf{E}[\hat{\mathsf{X}}]$ from below. We now go to the actual construction.

Proof of Theorem 6.4. Following the intuition above, the distributions will be based on the evaluations of certain multivariate polynomials at a particular point. Specifically, for every $0 \leq i \leq k-1$ let $s_i(y_1, \ldots, y_{k-1})$ be the ith symmetric function

$$s_i(y_1, \ldots, y_{k-1}) = \sum_{\substack{T \subset [k-1] \\ |T|=i}} \prod_{j \in T} y_j \,. \tag{45}$$

For example, if $k = 4$ and $i = 2$ then $s_2(y_1, \ldots, y_{k-1}) = y_1y_2 + y_1y_3 + y_2y_3$. In general, $s_0 = 1$ and $s_{k-1}(y_1, ..., y_{k-1}) = y_1 \cdot \ldots \cdot y_{k-1}$.

Let a be some integer that we fix later, and define

$$s_i(a) \stackrel{\text{def}}{=} s_i(a, a^2, \ldots, a^{k-1}) \,. \tag{46}$$

Following our previous example, $s_2(a) = a^3 + a^4 + a^5$, while $s_3(a) = a^6$. In general, as we later prove in detail, $s_i(a)$ is larger than $s_{i-1}(a)$ for sufficiently large a.

Consider the polynomial $f(t) = \prod_{i=1}^{k-1}(t - a^i)$. It is easy to see that

$$f(t) = (-1)^{k-1} \cdot \sum_{i=0}^{k-1} (-1)^i \cdot s_{k-1-i}(a) \cdot t^i. \tag{47}$$

As in the overview above, we construct two distributions out of the coefficients of f. The supports of the distributions are contained in the set $\{1, a, a^2, \ldots, a^{k-1}\}$. We define

$$\forall i, \;\; 0 \leq i \leq k-1 \quad \Pr[\hat{\mathsf{X}} = a^i] = \begin{cases} \frac{s_{k-1-i}(a)}{\hat{\mathsf{N}}(a)} & \text{for even } i \\ 0 & \text{for odd } i \end{cases} \tag{48}$$

$$\forall i, \;\; 0 \leq i \leq k-1 \quad \Pr[\tilde{\mathsf{X}} = a^i] = \begin{cases} 0 & \text{for even } i \\ \frac{s_{k-1-i}(a)}{\hat{\mathsf{N}}(a)} & \text{for odd } i \end{cases} \tag{49}$$

where

$$\hat{\mathsf{N}}(a) \stackrel{\text{def}}{=} \sum_{j=0}^{\lfloor (k-1)/2 \rfloor} s_{k-1-2j}(a) \;\; \textit{and} \;\; \tilde{\mathsf{N}}(a) \stackrel{\text{def}}{=} \sum_{j=0}^{\lfloor (k-2)/2 \rfloor} s_{k-2-2j}(a) \tag{50}$$

are normalizing factors.

We now show that for an appropriate choice of the parameter a, the distributions $\hat{\mathsf{X}}$ and $\tilde{\mathsf{X}}$ satisfy the requirements of Theorem 6.4. Let $C \stackrel{\text{def}}{=} \frac{\hat{\mathsf{N}}(a)}{\tilde{\mathsf{N}}(a)}$.

Lemma 6.6

$$C \cdot \mathsf{E}[\hat{\mathsf{X}}^\ell] = \mathsf{E}[\tilde{\mathsf{X}}^\ell] \quad \textit{for } \ell = 1, \ldots, k-1.$$

Proof: We have

$$\begin{aligned} C \cdot \mathsf{E}[\hat{\mathsf{X}}^\ell] - \mathsf{E}[\tilde{\mathsf{X}}^\ell] &= C \cdot \sum_{\substack{0 \le i \le k-1 \\ \text{i even}}} (a^i)^\ell \cdot \frac{s_{k-1-i}(a)}{\hat{\mathsf{N}}(a)} - \sum_{\substack{0 \le i \le k-1 \\ \text{i odd}}} (a^i)^\ell \cdot \frac{s_{k-1-i}(a)}{\tilde{\mathsf{N}}(a)} \\ &= \frac{1}{\tilde{\mathsf{N}}(a)} \cdot \sum_{i=0}^{k-1} (-1)^i \cdot s_{k-1-i}(a) \cdot (a^\ell)^i = \frac{(-1)^{k-1}}{\tilde{\mathsf{N}}(a)} \cdot f(a^\ell) = 0. \end{aligned} \tag{51}$$

■

Notice that for the case $\ell = 1$, Lemma 6.6 implies that $C = \mathsf{E}[\tilde{\mathsf{X}}] / \mathsf{E}[\hat{\mathsf{X}}]$. Next we show that for any $B \ge 1$ we can find a such that $\mathsf{E}[\tilde{\mathsf{X}}] / \mathsf{E}[\hat{\mathsf{X}}] \ge B$. For this the following upper bound on the value of $s_i(a)$ is needed.

Lemma 6.7 *For every $1 \le i \le k-1$ it holds that*

$$\frac{s_{i-1}(a)}{s_i(a)} \le \frac{i}{a^{k-i}} \, .$$

Proof: Recall that

$$s_i(a) = s_i(a, a^2, \ldots, a^{k-1}) = \sum_{\substack{T \subset [k-1] \\ |T| = i}} \prod_{j \in T} a^j,$$

and

$$s_{i-1}(a) = s_{i-1}(a, a^2, \ldots, a^{k-1}) = \sum_{\substack{T \subset [k-1] \\ |T| = i-1}} \prod_{j \in T} a^j \, .$$

We prove the lemma by mapping each term in s_{i-1} to a term in s_i. Given a product of $i-1$ elements, $a^{e_1} \cdot a^{e_2} \cdot \ldots \cdot a^{e_{i-1}}$, map it to a product of i elements in the following way: let $e \le k-1$ be the largest integer not in the set $\{e_1, \ldots, e_{i-1}\}$. Note that $e \ge k - i$. Then the product $a^{e_1} \cdot a^{e_2} \cdot \ldots \cdot a^{e_{i-1}}$ is mapped to the product $a^{e_1} \cdot a^{e_2} \cdot \ldots \cdot a^{e_{i-1}} \cdot a^e$. By definition, we map at most i products of $i-1$ elements to each product of i elements. Thus, $a^{k-i} \cdot s_{i-1}(a) \le i \cdot s_i(a)$, and the lemma follows. ■

As an immediate corollary we get

Corollary 6.8 *For every* $1 \leq i \leq k-1$ *it holds that*

$$\frac{s_{k-i}(a)}{s_{k-1}(a)} \leq \frac{(k-1)^{i-1}}{a^{\binom{i}{2}}} \,.$$

Proof: By Lemma 6.7 we get

$$\frac{s_{k-i}(a)}{s_{k-1}(a)} = \prod_{j=k-i+1}^{k-1} \frac{s_{j-1}(a)}{s_j(a)} \leq \prod_{j=k-i+1}^{k-1} \frac{j}{a^{k-j}} \leq \frac{(k-1)^{i-1}}{a^{\binom{i}{2}}} \,. \tag{52}$$

■

We also need the following estimate on $\tilde{\mathsf{N}}(a)$.

Lemma 6.9 *For* $a \geq \max(k,2)$*, we have that* $\tilde{\mathsf{N}}(a) < s_{k-1}(a) \cdot \frac{k}{a}$.

Proof: By Corollary 6.8 and the assumption that $a \geq \max(k,2)$,

$$\begin{aligned}
\tilde{\mathsf{N}}(a) &= \sum_{j=0}^{\lfloor (k-2)/2 \rfloor} s_{k-2-2j}(a) \\
&= s_{k-1}(a) \cdot \sum_{j=0}^{\lfloor (k-2)/2 \rfloor} \frac{s_{k-2-2j}(a)}{s_{k-1}(a)} && (53) \\
&\leq s_{k-1}(a) \cdot \sum_{j=0}^{\lfloor (k-2)/2 \rfloor} \frac{(k-1)^{2j+1}}{a^{\binom{2j+2}{2}}} && (54) \\
&\leq s_{k-1}(a) \cdot \left(\frac{k-1}{a} + \sum_{j=1}^{\lfloor (k-2)/2 \rfloor} \frac{1}{a^{\binom{2j+1}{2}}} \right) && (55) \\
&< s_{k-1}(a) \cdot \left(\frac{k-1}{a} + \frac{1}{a^3 - a} \right) && (56) \\
&< s_{k-1}(a) \cdot \frac{k}{a} \,. && (57)
\end{aligned}$$

■

It remains to find, for every $B > 1$, an a such that $\mathsf{E}[\tilde{\mathsf{X}}]/\,\mathsf{E}[\hat{\mathsf{X}}] \geq B$. By Lemmas 6.6 and 6.9,

$$\frac{\mathsf{E}[\tilde{\mathsf{X}}]}{\mathsf{E}[\hat{\mathsf{X}}]} = C = \frac{\hat{\mathsf{N}}(a)}{\tilde{\mathsf{N}}(a)} > \frac{\hat{\mathsf{N}}(a)}{s_{k-1}(a) \cdot \frac{k}{a}} \geq \frac{s_{k-1}(a)}{s_{k-1}(a) \cdot \frac{k}{a}} = \frac{a}{k}. \tag{58}$$

Thus, if we take $a = B \cdot k$ then $\mathsf{E}[\tilde{\mathsf{X}}]/\,\mathsf{E}[\hat{\mathsf{X}}] > B$. This completes the construction and the proof of Theorem 6.4. ■

6.4 Indistinguishability by Uniform Algorithms: Proof of Lemma 6.5

Lemma 6.5 gives a bound on how well a uniform algorithm can distinguish the pairs of distributions constructed in the previous section. Recall that we showed (see Lemma 5.3) that we can restrict our attention to such algorithms without loss of generality. Even though uniform algorithms are much simpler than general algorithms, they still might be tricky to analyze because of dependencies between the numbers of balls of various colors that appear in the sample. Batu *et al.* [BDKR05] showed that such dependencies are avoided when an algorithm takes a random number of samples according to a *Poisson* distribution. The Poisson distribution $\mathrm{Po}(\lambda)$ takes on the value $k \in \mathbb{N}$ with probability $e^{\lambda}\lambda^k/k!$. In Appendix B we state some useful properties of the Poisson distribution, which we use throughout the rest of the section.

Definition 6.10 *We call a uniform algorithm* Poisson-s *if the number of samples it takes is a random variable, distributed as* $\mathrm{Po}(s)$.

Batu *et al.* [BDKR05] proved a variant of the following lemma in the context of entropy estimation of distributions. However, the statements and the proofs also apply to estimating symmetric functions over strings and, in particular, to Colors.

Lemma 6.11 ([BDKR05])

(a) Poisson algorithms can simulate uniform algorithms. Specifically, for every uniform *algorithm $\mathcal{A}$ that uses at most $\frac{s}{2}$ samples, there is a Poisson-s algorithm $\mathcal{A}'$ such that for every input w, the statistical difference between the distributions $\mathcal{A}(w)$ and $\mathcal{A}'(w)$ is $o(1/s)$.*

(b) If the number of balls of a particular color in the input to Colors is b, then the number of balls of that color seen by a Poisson-s algorithm is distributed as $\mathrm{Po}(\frac{b \cdot s}{n})$. Moreover, it is independent of the number of balls of all other colors in the sample.

(c) For any function invariant under permutations of the alphabet symbols (color names), any Poisson algorithm can be simulated by an algorithm that gets only the histogram of the sample as its input. The simulation has the same approximation guarantees as the original algorithm.

The independence of the number of occurrences of different colors in the sample (Property (b) above) will be very useful in analyzing the distributions seen by the algorithm.

6.4.1 Analyzing the Distributions on Histograms

Recall that $X \approx_\delta Y$ denotes that the statistical difference between random variables X and Y is at most δ.

Consider a particular Poisson-s algorithm. For $\ell = 0, 1, \ldots, s$, let F_ℓ be a random variable representing the number of ℓ-way collisions the Poisson algorithm sees, and $\mathsf{F} = (\mathsf{F}_1, \mathsf{F}_2, \mathsf{F}_3 \ldots)$ be the corresponding histogram. We can restate Lemma 6.5 in terms of histograms:

Lemma 6.12 (Main Lemma, Restated) *For $s \leq \frac{n}{2 \cdot a_{k-1}}$, the statistical difference between the histograms $(\hat{\mathsf{F}}_1, \hat{\mathsf{F}}_2, \hat{\mathsf{F}}_3, \ldots)$ and $(\tilde{\mathsf{F}}_1, \tilde{\mathsf{F}}_2, \tilde{\mathsf{F}}_3, \ldots)$ is at most*

$$O\left(k^2 \left(\frac{a_{k-1}}{n} \cdot s\right)^{2/3} + \frac{k}{\lfloor \frac{k}{2} \rfloor ! \cdot \lceil \frac{k}{2} \rceil !} \cdot \left(\frac{a_{k-1}}{n}\right)^{k-1} \cdot s^k\right) .$$

For the remainder of this section, we assume that $s \leq \frac{n}{2 \cdot a_{k-1}}$. The next lemma states that ℓ-way collisions are very unlikely for $\ell \geq k$, when s is sufficiently small.

Lemma 6.13 *The probability that there is a collision involving $k > 1$ or more balls is at most*

$$\delta_1 = O\left(\frac{1}{k!} \cdot \left(\frac{a_{k-1}}{n}\right)^{k-1} \cdot s^k\right).$$

The proof of Lemma 6.13 is given later. To complete the proof of Lemma 6.12, we need two additional lemmas that show that certain pairs of distributions are close when s is sufficiently small.

Lemma 6.14 $\hat{\mathsf{F}}_\ell \approx_{\delta_2} \tilde{\mathsf{F}}_\ell$ *for $\ell \in [k-1]$, where* $\delta_2 = O\left(\frac{k \cdot a_{k-1} \cdot s}{n} + \frac{1}{\lfloor \frac{k}{2} \rfloor ! \cdot \lceil \frac{k}{2} \rceil !} \cdot \left(\frac{a_{k-1}}{n}\right)^{k-1} \cdot s^k\right)$.

Lemma 6.15 *For both distributions, $\mathsf{F}_1, \ldots, \mathsf{F}_{k-1}$ are close to independent, that is, $(\mathsf{F}_1, \ldots, \mathsf{F}_{k-1}) \approx_{\delta_3} (\mathsf{F}'_1, \ldots, \mathsf{F}'_{k-1})$, where the variables F'_ℓ are independent, for each ℓ the distributions of F_ℓ and F'_ℓ are identical, and $\delta_3 = O\left(k^2 \cdot (\frac{a_{k-1} \cdot s}{n})^{2/3}\right)$.*

The main claim (Lemma 6.12) follows by a standard hybrid argument. Consider a chain of distributions "between" the two histograms of Lemma 6.12. Starting from the "hat" histogram, we first replace all counts of collisions greater than k by 0, and then replace each count $\hat{\mathsf{F}}_\ell$ with an independent copy $\hat{\mathsf{F}}'_\ell$ for $\ell \in [k-1]$, as in Lemma 6.15. Next, change each $\hat{\mathsf{F}}'_\ell$ with a corresponding $\tilde{\mathsf{F}}'_\ell$. Finally, replace these independent $\tilde{\mathsf{F}}'_\ell$s with the real, dependent variables $\tilde{\mathsf{F}}_\ell$ and add back in the counts of the collisions involving more than k variables to obtain the "tilde" histogram. The resulting chain has $k+3$ steps. By the triangle inequality, we can sum these differences to obtain a bound on the difference between the two histograms. In symbols, the chain of distribution looks as follows (where δ_1, δ_2 and δ_3 are as defined in Lemmas 6.13, 6.14 and 6.15, respectively:

$$\begin{array}{rllll}
 & (\hat{\mathsf{F}}_1, & \ldots, & \hat{\mathsf{F}}_{k-1}, & \hat{\mathsf{F}}_k, \hat{\mathsf{F}}_{k+1}, \ldots) \\
\approx_{\delta_1} & (\hat{\mathsf{F}}_1, & \ldots, & \hat{\mathsf{F}}_{k-1}, & 0, 0, \ldots) \\
\approx_{\delta_3} & (\hat{\mathsf{F}}'_1, & \ldots, & \hat{\mathsf{F}}'_{k-1}, & 0, 0, \ldots) \\
\approx_{\delta_2} & (\tilde{\mathsf{F}}'_1, & \ldots, & \hat{\mathsf{F}}'_{k-1}, & 0, 0, \ldots) \\
 & & & \vdots & \\
\approx_{\delta_2} & (\tilde{\mathsf{F}}'_1, & \ldots, & \tilde{\mathsf{F}}'_{k-1}, & 0, 0, \ldots) \\
\approx_{\delta_3} & (\tilde{\mathsf{F}}_1, & \ldots, & \tilde{\mathsf{F}}_{k-1}, & 0, 0, \ldots) \\
\approx_{\delta_1} & (\tilde{\mathsf{F}}_1, & \ldots, & \tilde{\mathsf{F}}_{k-1}, & \tilde{\mathsf{F}}_k, \tilde{\mathsf{F}}_{k+1}, \ldots)
\end{array}$$

The total statistical difference is at most

$$\begin{aligned}
& 2 \cdot \delta_1 + 2 \cdot \delta_3 + (k-1) \cdot \delta_2 \qquad (59) \\
&= O\left(\frac{1}{k!} \cdot \left(\frac{a_{k-1}}{n}\right)^{k-1} \cdot s^k + k^2 \cdot \left(\frac{a_{k-1} \cdot s}{n}\right)^{2/3} + k \cdot \frac{k \cdot a_{k-1} \cdot s}{n} + \frac{k}{\lfloor \frac{k}{2} \rfloor ! \cdot \lceil \frac{k}{2} \rceil !} \cdot \left(\frac{a_{k-1}}{n}\right)^{k-1} \cdot s^k\right).
\end{aligned}$$

The first and third terms are negligible given the others. Removing them yields the claimed bound. This completes the proof of Lemma 6.12. We now prove the three intermediate lemmas used in the proof of Lemma 6.12.

Recall that we consider inputs with $C_i = \left\lfloor \frac{p_i n}{\mathsf{E}[\mathsf{X}]} \right\rfloor$ colors of type i, for $i = 0, \ldots, k-1$, where each color of type i appears a_i times, and with $C_k = n - \sum_{i=0}^{k-1} \left\lfloor \frac{p_i n}{\mathsf{E}[\mathsf{X}]} \right\rfloor \cdot a_i$ additional colors that appear once each. We say that the C_k "left-over" colors are of type k, and define $a_k = 1$. Note that $C_k < \sum_{i=0}^{k-1} a_i$ (since $\mathsf{E}[\mathsf{X}] = \sum_{i=0}^{k-1} p_i \cdot a_i$ and so $n = \sum_{i=0}^{k-1} \frac{p_i n}{\mathsf{E}[\mathsf{X}]} \cdot a_i$).

The independence of the number of different colors (see Lemma 6.11(b)) makes it easy to understand the distribution on the number of ℓ-way collisions. For a color that appears a_i times in the input, the number of balls of that color in the sample is distributed according to $\mathrm{Po}(\lambda_i)$, where $\lambda_i = \frac{a_i s}{n}$. The probability that this color appears exactly ℓ times in the sample is $\frac{\lambda_i^\ell}{\ell!} e^{-\lambda_i}$.

Proof of Lemma 6.13. Consider any particular color of type i. The probability that the algorithm sees k or more balls of that color is $\Pr[\mathrm{Po}(\lambda_i) \geq k] \leq \frac{\lambda_i^k}{k!}$. Summing over all colors (according to types), we can bound the probability that some color appears k or more times by:

$$\begin{aligned}
\sum_{i=0}^{k} C_i \cdot \frac{\lambda_i^k}{k!} &= \sum_{i=0}^{k} C_i \cdot \frac{1}{k!} \left(\frac{a_i s}{n}\right)^k && (60)\\
&< \frac{s^k}{k! \cdot \mathsf{E}[\mathsf{X}] \cdot n^{k-1}} \cdot \sum_{i=0}^{k-1} p_i a_i^k + \frac{a_{k-1} \cdot s^k}{n^k} && (61)\\
&= \frac{s^k}{k! \cdot n^{k-1}} \cdot \left(\frac{\mathsf{E}[\mathsf{X}^k]}{\mathsf{E}[\mathsf{X}]} + \frac{a_{k-1}}{n} \right) . && (62)
\end{aligned}$$

Finally, we can bound the ratio $\frac{\mathsf{E}[\mathsf{X}^k]}{\mathsf{E}[\mathsf{X}]}$ by $(a_{k-1})^{k-1}$. This yields the desired bound. ■

Proof of Lemma 6.14. Observe that F_ℓ, the number of ℓ-way collisions, is a sum of independent Bernoulli random variables, one for each color, with probability $\frac{1}{\ell!} \cdot e^{-\frac{as}{n}} \cdot \left(\frac{as}{n}\right)^\ell$ of being 1 if the color appeared a times in the input. Hence, the number of ℓ-way collisions is a sum of independent binomial random variables, one for each type. That is,

$$\mathsf{F}_\ell \sim \sum_{i=0}^{k} \mathrm{Bin}\left(C_i \, , \frac{e^{-\lambda_i} \lambda_i^\ell}{\ell!} \right) , \tag{63}$$

where $\mathrm{Bin}(m, p)$ is the number of heads in a sequence of m independent coin flips, each of which has probability p of heads.

When p is small, the Poisson distribution $\mathrm{Po}(\lambda = pm)$ is a good approximation to $\mathrm{Bin}(m, p)$; the statistical difference between the two is at most p (see Lemma B.1, Item (3)). Since the sum of independent Poisson variables is also a Poisson variable (see Lemma B.1, Item (2)),

$$\mathsf{F}_\ell \approx_{\gamma_\ell} \mathrm{Po}\left(\lambda^{(\ell)} = \sum_{i=0}^{k} C_i \cdot \frac{\lambda_i^\ell}{\ell!} e^{-\lambda_i} \right) \tag{64}$$

where

$$\gamma_\ell \leq \sum_{i=0}^{k} \frac{e^{-\lambda_i} \lambda_i^\ell}{\ell!} \leq \sum_{i=0}^{k} \lambda_i \leq \frac{k \cdot a_{k-1} \cdot s}{n} . \tag{65}$$

In the last equality we have used the fact that $a_k = 1$ and $a_i < a_{k-1}$ for every $i < k-1$.

To bound the statistical difference between $\hat{\mathsf{F}}_\ell$ and $\tilde{\mathsf{F}}_\ell$ from above, it is enough to bound the difference between $\hat{\lambda}^{(\ell)}$ and $\tilde{\lambda}^{(\ell)}$, since the statistical difference between $\mathrm{Po}(\hat{\lambda}^{(\ell)})$ and $\mathrm{Po}(\tilde{\lambda}^{(\ell)})$ is at most $|\hat{\lambda}^{(\ell)} - \tilde{\lambda}^{(\ell)}|$ (see Lemma B.1, Item (5)).

Substituting $\frac{a_i}{n} \cdot s$ for λ_i and using the fact that $e^{-\lambda_i} = \sum_{j=0}^{k-\ell-1}(-1)^j \cdot \frac{\lambda_i^j}{j!} + (-1)^{k-\ell} \cdot O\left(\frac{\lambda_i^{k-\ell}}{(k-\ell)!}\right)$, (where we define $0! = 1$) we get that

$$\lambda^{(\ell)} = \frac{1}{\ell!} \cdot \sum_{j=0}^{k-\ell} T_j^{(\ell)} \tag{66}$$

where

$$T_j^{(\ell)} = (-1)^j \cdot \frac{1}{j!} \cdot \frac{s^{\ell+j}}{n^{\ell+j}} \cdot \sum_{i=0}^{k} C_i \cdot a_i^{\ell+j} \tag{67}$$

for $0 \leq j \leq k-\ell-1$, and

$$T_{k-\ell}^{(\ell)} = (-1)^{k-\ell} \cdot O\left(\frac{1}{(k-\ell)!} \cdot \frac{s^k}{n^k} \cdot \sum_{i=0}^{k} C_i \cdot a_i^k\right) \tag{68}$$

For each j, $0 \leq j \leq k-\ell$, we have that

$$\begin{aligned}
\frac{s^{\ell+j}}{n^{\ell+j}} \cdot \sum_{i=0}^{k} C_i \cdot a_i^{\ell+j} &= \frac{s^{\ell+j}}{n^{\ell+j}} \cdot \left(\sum_{i=0}^{k-1} \left\lfloor \frac{p_i n}{\mathsf{E}[\mathsf{X}]} \right\rfloor \cdot a_i^{\ell+j} + C_k\right) && (69)\\
&\leq \frac{s^{\ell+j}}{n^{\ell+j-1}} \cdot \frac{1}{\mathsf{E}[\mathsf{X}]} \cdot \sum_{i=0}^{k-1} p_i \cdot a_i^{\ell+j} + \frac{s^{\ell+j}}{n^{\ell+j}} \cdot \sum_{i=0}^{k-1} a_i && (70)\\
&= \frac{s^{\ell+j}}{n^{\ell+j-1}} \cdot \frac{\mathsf{E}[\mathsf{X}^{\ell+j}]}{\mathsf{E}[\mathsf{X}]} + \frac{s^{\ell+j}}{n^{\ell+j}} \cdot \sum_{i=0}^{k-1} a_i && (71)
\end{aligned}$$

and similarly

$$\begin{aligned}
\frac{s^{\ell+j}}{n^{\ell+j}} \cdot \sum_{i=0}^{k} C_i \cdot a_i^{\ell+j} &= \frac{s^{\ell+j}}{n^{\ell+j}} \cdot \left(\sum_{i=0}^{k-1} \left\lfloor \frac{p_i n}{\mathsf{E}[\mathsf{X}]} \right\rfloor \cdot a_i^{\ell+j} + C_k\right) && (72)\\
&\geq \frac{s^{\ell+j}}{n^{\ell+j-1}} \cdot \frac{1}{\mathsf{E}[\mathsf{X}]} \cdot \sum_{i=0}^{k-1} p_i \cdot a_i^{\ell+j} - \frac{s^{\ell+j}}{n^{\ell+j}} \cdot \sum_{i=0}^{k-1} a_i^{\ell+j} && (73)\\
&= \frac{s^{\ell+j}}{n^{\ell+j-1}} \cdot \frac{\mathsf{E}[\mathsf{X}^{\ell+j}]}{\mathsf{E}[\mathsf{X}]} - \frac{s^{\ell+j}}{n^{\ell+j}} \cdot \sum_{i=0}^{k-1} a_i^{\ell+j} && (74)
\end{aligned}$$

The moment condition on $\hat{\mathsf{X}}$ and $\tilde{\mathsf{X}}$ states that $\frac{\mathsf{E}[\hat{\mathsf{X}}^{\ell+j}]}{\mathsf{E}[\hat{\mathsf{X}}]} = \frac{\mathsf{E}[\tilde{\mathsf{X}}^{\ell+j}]}{\mathsf{E}[\tilde{\mathsf{X}}]}$ for $j = 0, \ldots, k-\ell-1$. Thus,

$$\left|\hat{\lambda}^{(\ell)} - \tilde{\lambda}^{(\ell)}\right| = O\left(\frac{1}{\ell!} \cdot \sum_{j=0}^{k-\ell} \frac{s^{\ell+j}}{n^{\ell+j}} \cdot 2 \sum_{i=0}^{k-1} a_i^{\ell+j} + \frac{1}{\ell!(k-\ell)!} \cdot \frac{s^k}{n^{k-1}} \cdot \max\left\{\frac{\mathsf{E}[\hat{\mathsf{X}}^k]}{\mathsf{E}[\hat{\mathsf{X}}]}, \frac{\mathsf{E}[\tilde{\mathsf{X}}^k]}{\mathsf{E}[\tilde{\mathsf{X}}]}\right\}\right) \tag{75}$$

The ratio $\frac{\mathsf{E}[\mathsf{X}^k]}{\mathsf{E}[\mathsf{X}]}$ is at most $(a_{k-1})^{k-1}$, the expression $\frac{1}{\ell!(k-\ell)!}$ is maximized for $\ell = \lfloor \frac{k}{2} \rfloor$, and

$$\frac{1}{\ell!} \cdot \sum_{j=0}^{k-\ell} \frac{s^{\ell+j}}{n^{\ell+j}} \cdot 2 \sum_{i=0}^{k-1} a_i^{\ell+j} \leq \frac{1}{\ell!} \cdot \left(\frac{s \cdot a_{k-1}}{n}\right)^{\ell} \cdot 2k \cdot \sum_{j=0}^{k-\ell} \left(\frac{s \cdot a_{k-1}}{n}\right)^{j} = O\left(\frac{k \cdot a_{k-1} \cdot s}{n}\right) \tag{76}$$

where the last equality uses the fact that $\frac{s \cdot a_{k-1}}{n} \leq \frac{1}{2}$. Therefore,

$$\left|\hat{\lambda}^{(\ell)} - \tilde{\lambda}^{(\ell)}\right| = O\left(\frac{1}{\lfloor \frac{k}{2} \rfloor ! \cdot \lceil \frac{k}{2} \rceil !} \cdot \left(\frac{a_{k-1}}{n}\right)^{k-1} \cdot s^k + \frac{k \cdot a_{k-1} \cdot s}{n}\right) \tag{77}$$

Summing this together with the error, denoted γ_ℓ, introduced by approximating a sum of binomials with a Poisson variable, proves the lemma. ■

In order to prove Lemma 6.15 we need the following lemma concerning multinomial variables, whose proof is provided in Appendix B.

Lemma 6.16 *Consider a k-sided die, whose sides are numbered $0, \ldots, k-1$, where side ℓ has probability q_ℓ and $q_0 \geq 1/2$. Let $Z_0, \ldots, Z_{k-1}$ be random variables that count the number of occurrences of each side in a sequence of independent rolls. Let $Z'_1, \ldots, Z'_{k-1}$ be* independent *random variables, where for each ℓ, the variable Z'_ℓ is distributed identically to Z_ℓ. Then $(Z_1, ..., Z_{k-1}) \approx_{\delta_4} (Z'_1, \ldots, Z'_{k-1})$ for $\delta_4 = O(k(1-q_0)^{2/3})$.*

Proof of Lemma 6.15. We can write F_ℓ as a sum $\mathsf{F}_\ell = \mathsf{F}_\ell^{(1)} + \cdots + \mathsf{F}_\ell^{(k)}$, where $\mathsf{F}_\ell^{(i)}$ is the number of ℓ-way collisions among colors of type i. Since the types are independent, it is sufficient to show that for each i, the variables $\mathsf{F}_1^{(i)}, \ldots, \mathsf{F}_{k-1}^{(i)}$ are close to being independent. We can then sum the distances over the types to prove the lemma.

Let $\mathsf{F}_0^{(i)}$ denote the number of colors of type i that occur either 0 times, or k or more times, in the sample. The vector $\mathsf{F}_0^{(i)}, \mathsf{F}_1^{(i)}, \ldots, \mathsf{F}_{k-1}^{(i)}$ follows a multinomial distribution. It counts the outcomes of an experiment in which C_i independent, identical dice are rolled, and each one produces outcome ℓ with probability $e^{-\lambda_i}\lambda_i^\ell/\ell!$, for $\ell \in [k-1]$, and outcome 0 with the remaining probability. On each roll, outcome 0 occurs with probability at least $e^{-\lambda_i} \geq 1 - \lambda_i \geq 1/2$ (recall that $\lambda_i = \frac{a_i \cdot s}{n} \leq 1/2$).

Lemma 6.16 shows that when one outcome occupies almost all the mass in such an experiment, the counts of the remaining outcomes are close to independent — within distance $O(k \cdot \lambda_i^{2/3})$. Summing over all types, the distance of $\mathsf{F}_1, \ldots, \mathsf{F}_{k-1}$ from independent is $O\left(k \cdot \sum_i \lambda_i^{2/3}\right) = O\left(k^2 \cdot \left(\frac{a_{k-1}s}{n}\right)^{2/3}\right)$. ■

References

[AMS99] Noga Alon, Yossi Matias, and Mario Szegedy. The space complexity of approximating the frequency moments. *J. Comput. Syst. Sci.*, 58(1):137–147, 1999.

[Bar02] Ziv Bar-Yossef. *The complexity of Massive Data Set Computations.* PhD thesis, Computer Science Division, U.C. Berkeley, 2002.

[BDKR05] Tugkan Batu, Sanjoy Dasgupta, Ravi Kumar, and Ronitt Rubinfeld. The complexity of approximating the entropy. *SIAM Journal on Computing*, 35(1):132–150, 2005.

[BKS01] Ziv Bar-Yossef, Ravi Kumar, and D. Sivakumar. Sampling algorithms: lower bounds and applications. In *STOC '01: Proceedings of the thirty-third annual ACM symposium on Theory of computing*, pages 266–275, New York, NY, USA, 2001. ACM Press.

[BS05] Mickey Brautbar and Alex Samorodnitsky. Approximating the entropy of large alphabets. *Manuscript*, 2005.

[CCMN00] Moses Charikar, Surajit Chaudhuri, Rajeev Motwani, and Vivek R. Narasayya. Towards estimation error guarantees for distinct values. In *PODS*, pages 268–279. ACM, 2000.

[CT91] T. Cover and J. Thomas. *Elements of Information Theory*. John Wiley & Sons, Inc., New York, 1991.

[LS02] Eric Lehman and Abhi Shelat. Approximation algorithms for grammer-based compression. In *Proceedings of the Thirteenth annual ACM–SIAM symposium on Discrete Algorithms*, pages 205–212, 2002.

[Pro53] Yu. V. Prohorov. Asymptotic behavior of the binomial distribution (Russian). *Uspekhii Matematicheskiikh Nauk*, 8(3):135–142, 1953. Moscow.

[Shl81] A. Shlosser. On estimation of the size of the dictionary of a long text on the basis of a sample. *Engineering Cybernetics*, 19:97–102, 1981.

[Web99] Michael Weba. Bounds for the total variation distance between the binomial and the poisson distribution in case of medium-sized success probabilities. *J. Appl. Probab.*, 36(1):97–104, 1999.

[Yao77] A. C. Yao. Probabilistic computation, towards a unified measure of complexity. In *Proceedings of the Eighteenth Annual Symposium on Foundations of Computer Science*, pages 222–227, 1977.

[ZL77] Jacob Ziv and Abraham Lempel. A universal algorithm for sequential data compression. *IEEE Transactions on Information Theory*, 23:337–343, 1977.

A A Lower Bound for Approximating the Entropy

The following problem was introduced by Batu *et al.* [BDKR05]. Let $\mathbf{p} = \langle p_1, \ldots, p_n \rangle$ be a discrete distribution over n elements, where p_i is the probability of the ith element. Given access to independent samples generated according to the distribution $\mathbf{p}$, we would like to approximate its entropy: $H(\mathbf{p}) = -\sum_{i=1}^{n} p_i \log p_i$. Batu *et al.* showed how to obtain an A-estimate in time $\tilde{O}\left(n^{\frac{1+\eta}{A^2}}\right)$, provided that $H(\mathbf{p}) = \Omega\left(\frac{A}{\eta}\right)$. They also proved a lower bound of $\Omega\left(n^{\frac{1}{2A^2}}\right)$ that holds even when $H(\mathbf{p}) = \Omega\left(\frac{\log n}{A^2}\right)$. (Without a lower bound on $H(\mathbf{p})$, the time complexity is unbounded.)

Here we use our technique to obtain a lower bound of $\Omega\left(n^{\frac{2}{6A^2-3+o(1)}}\right)$, improving on the $\Omega\left(n^{\frac{1}{2A^2}}\right)$ lower bound for relatively small A. When A is close to 1, the bound is close to $n^{2/3}$ (rather than $n^{1/2}$).

Moments conditions. We first provide a different construction of random variables that satisfy the conditions of Theorem 6.4 for the special case of $k = 3$ and $B = B(n)$, which we set momentarily. Specifically, let $a_0 = 1$, $a_1 = 2B$, and $a_2 = 4B - 2$, and set

$$\Pr[\hat{\mathsf{X}} = a_0] = 1 - \frac{1}{4B-3}, \;\; \Pr[\hat{\mathsf{X}} = a_1] = 0, \;\; \Pr[\hat{\mathsf{X}} = a_2] = \frac{1}{4B-3}\,, \tag{78}$$

and

$$\Pr[\tilde{\mathsf{X}} = a_0] = 0, \;\; \Pr[\tilde{\mathsf{X}} = a_1] = 1, \;\; \Pr[\tilde{\mathsf{X}} = a_2] = 0\,. \tag{79}$$

By definition of $\hat{\mathsf{X}}$ and $\tilde{\mathsf{X}}$,

$$\mathsf{E}[\hat{\mathsf{X}}] = \left(1 - \frac{1}{4B-3}\right) \cdot 1 + \frac{1}{4B-3} \cdot (4B-2) = \frac{4B-4+4B-2}{4B-3} \;\; = \;\; 2 \tag{80}$$

and

$$\mathsf{E}[\hat{\mathsf{X}}^2] = \left(1 - \frac{1}{4B-3}\right) \cdot 1^2 + \frac{1}{4B-3} \cdot (4B-2)^2 = \frac{4B-4+16B^2-16B+4}{4B-3} \;\; = \;\; 4B \tag{81}$$

while

$$\mathsf{E}[\tilde{\mathsf{X}}] = 1 \cdot 2B = 2B \quad \text{and} \quad \mathsf{E}[\tilde{\mathsf{X}}^2] = 4B^2 \tag{82}$$

so that

$$\frac{\mathsf{E}[\tilde{\mathsf{X}}]}{\mathsf{E}[\hat{\mathsf{X}}]} = B, \quad \text{and} \quad \frac{\mathsf{E}[\hat{\mathsf{X}}^2]}{\mathsf{E}[\hat{\mathsf{X}}]} = \frac{\mathsf{E}[\tilde{\mathsf{X}}^2]}{\mathsf{E}[\tilde{\mathsf{X}}]} \tag{83}$$

as required.

The two distributions and their entropies. Similarly to what was shown in Subsection 6.4.1, given the two random variables $\hat{\mathsf{X}}$ and $\tilde{\mathsf{X}}$, define two distributions over n elements (or, more precisely, two families of distributions). One distribution, denoted $\mathbf{p}_{\hat{\mathsf{X}}}$, has support on $\frac{n}{2} \cdot \frac{4B-4}{4B-3}$ elements of weight $\frac{1}{n}$ each and $\frac{n}{2} \cdot \frac{1}{4B-3}$ elements of weight $\frac{4B-2}{n}$ each. The second distribution, denoted $\mathbf{p}_{\tilde{\mathsf{X}}}$, has support on $\frac{n}{2B}$ elements of weight $\frac{2B}{n}$ each. As stated above, we can define two families of distributions, $F_{\hat{\mathsf{X}}}$ and $F_{\tilde{\mathsf{X}}}$, respectively, where we allow all permutations over the names (colors) of the elements. Let $D'_{\hat{\mathsf{X}}}$ denote the uniform distribution over $F_{\hat{\mathsf{X}}}$, and let $D'_{\tilde{\mathsf{X}}}$ denote the uniform distribution over $F_{\tilde{\mathsf{X}}}$.

Let $B = B(n)$ be of the form $B = \frac{1}{2}n^{1-\beta}$ for $\beta < 1$. Then the entropy of each distribution in

$F_{\tilde{\mathsf{X}}}$ is $\beta \log n$, and the entropy of each distribution in $F_{\hat{\mathsf{X}}}$ is:

$$\begin{aligned}
&\frac{2B-2}{4B-3} \cdot \log n + \frac{2B-1}{4B-3} \cdot \log \frac{n}{4B-2} \\
&= \frac{1}{2} \cdot \left(\log n + \log \frac{n}{4B-2} \right) - \frac{1}{8B-6} \cdot \left(\log n - \log \frac{n}{4B-2} \right) && (84) \\
&\geq \frac{1}{2} \cdot \left(\log n + \log n^{\beta} - 1 \right) - \frac{\log(2n^{1-\beta})}{4n^{1-\beta}-6} && (85) \\
&\geq \frac{1+\beta}{2} \log n - 1 && (86)
\end{aligned}$$

where the last inequality holds for sufficiently large n. Therefore, the ratio between the entropies is $\frac{1+\beta}{2\beta} - o(1)$.

While Lemma 6.5 was stated for the distributions on strings, $D_{\hat{\mathsf{X}}}$ and $D_{\tilde{\mathsf{X}}}$, and any algorithm that takes uniform samples from an input string of length n, it is not hard to verify that it also holds for the distributions $D'_{\hat{\mathsf{X}}}$ and $D'_{\tilde{\mathsf{X}}}$ and any algorithm that is provided with samples from distributions over n elements. Since $k = 3$ and $a_2 = 2n^{1-\beta}$, in order to distinguish the two distributions it is necessary to observe $\Omega\left(\left(\frac{n}{a_2}\right)^{2/3}\right) = \Omega\left(n^{2\beta/3}\right)$ samples. In other words, $\Omega\left(n^{2\beta/3}\right) = \Omega\left(n^{\frac{2}{6A^2-3+o(1)}}\right)$ samples are required for $A = \left(\sqrt{\frac{1+\beta}{2\beta}} - o(1)\right)$-estimating the entropy.

B Properties of the Poisson Distribution and Proof of Lemma 6.16

We start with some useful properties of the Poisson distribution:

Lemma B.1 *1. If $X \sim \mathrm{Po}(\lambda)$, then $\mathsf{E}[X] = \mathsf{Var}[X] = \lambda$.*

2. If $X \sim \mathrm{Po}(\lambda)$, $Y \sim \mathrm{Po}(\lambda')$ and X, Y are independent, then $X + Y \sim \mathrm{Po}(\lambda + \lambda')$.

3. The statistical difference between $\mathrm{Bin}(m, p)$ and $\mathrm{Po}(mp)$ is at most p.

4. For $\lambda > 0$, the statistical difference between $Po(\lambda)$ and $Po(\lambda + \epsilon\sqrt{\lambda})$ is $O(\epsilon)$.

5. The statistical difference between $\mathrm{Po}(\lambda)$ and $\mathrm{Po}(\lambda')$ is at most $|\lambda - \lambda'|$.

Note: Item (5) provides a good bound when λ is near or equal to 0. In most settings, Item (4) is more useful.

Proof: Items (1) and (2) can be found in any standard probability text. For Item (3) (and other bounds on the Poisson approximation to the binomial), see [Pro53] or [Web99, Bound b_1]. To prove item (4), first compute the *relative entropy* (also called *Kullback-Liebler divergence*) between $\mathrm{Po}(\lambda')$ and $\mathrm{Po}(\lambda)$. For probability distributions p, q, the relative entropy is $D(p\|q) = \sum_x p(x) \ln \frac{p(x)}{q(x)}$. The statistical difference between p and q is at most $\sqrt{2 \ln(2) D(p\|q)}$ (see, e.g., [CT91, Lemma 12.6.1]). If $X \sim \mathrm{Po}(\lambda + \Delta)$, then the relative entropy in our case is

$$D(\mathrm{Po}(\lambda+\epsilon)\|\mathrm{Po}(\lambda)) = \mathop{\mathsf{E}}_{X}[\ln(\tfrac{e^{-\lambda-\Delta}(\lambda+\Delta)^X/X!}{e^{-\lambda}\lambda^X/X!})] = -\Delta + (\lambda+\Delta)\ln(\tfrac{\lambda+\Delta}{\lambda}).$$

Since $\ln(1+x) \leq x$, the relative entropy is at most Δ^2/λ, and the statistical difference is at most $\Delta\sqrt{\frac{2\ln(2)}{\lambda}}$. Setting $\Delta = \epsilon\sqrt{\lambda}$, we obtain the desired bound.

Finally, to prove Item (5) write $\mathrm{Po}(\lambda+\Delta)$ as a sum of two independent Poisson variables X_λ, X_Δ with parameters λ and Δ respectively. Conditioned on the event $X_\Delta = 0$, the sum is distributed as $\mathrm{Po}(\lambda)$. This event occurs with probability $e^{-\Delta} \geq 1-\Delta$. The statistical difference between $\mathrm{Po}(\lambda)$ and $\mathrm{Po}(\lambda+\Delta)$ is thus at most Δ, as desired. ■

We next repeat and prove Lemma 6.16.

Lemma 6.16 *Consider a k-sided die, whose sides are numbered $0,\ldots,k-1$, where side ℓ has probability q_ℓ and $q_0 \geq 1/2$. Let $Z_0,\ldots,Z_{k-1}$ be random variables that count the number of occurrences of each side in a sequence of independent rolls. Let $Z'_1,\ldots,Z'_{k-1}$ be* independent *random variables, where for each ℓ, the variable Z'_ℓ is distributed identically to Z_ℓ. Then $(Z_1,...,Z_{k-1}) \approx_{\delta_4} (Z'_1,\ldots,Z'_{k-1})$ for $\delta_4 = O(k(1-q_0)^{2/3})$.*

Proof: Suppose the die is rolled m times. For each $\ell > 1$, the number of occurrences of side ℓ is a binomial: $Z_\ell \sim \mathrm{Bin}(m, q_\ell)$. By Item (3) in Lemma B.1, the difference between $\mathrm{Bin}(m,q_\ell)$ and $\mathrm{Po}(\lambda = mq_\ell)$ is at most q_ℓ.

Consider Z_ℓ, conditioned on the values of $Z_1,\ldots,Z_{\ell-1}$. The distribution of Z_ℓ is still binomial but has different parameters: $Z_\ell \sim \mathrm{Bin}\left(m - S_\ell, \frac{q_\ell}{1-Q_\ell}\right)$, where $S_\ell = \sum_{i=1}^{\ell-1} Z_i$ and $Q_\ell = \sum_{i=1}^{\ell-1} q_i$. We can approximate this by a Poisson variable with parameter $\lambda' = (m-S_\ell)\frac{q_\ell}{1-Q_\ell}$. This approximation introduces an error (statistical difference) of at most q_ℓ.

Now the sum S_ℓ is also binomial, with parameters m, Q_ℓ. It has expectation mQ_ℓ and variance $mQ_\ell(1-Q_\ell)$. By Chebyshev's inequality, $S_\ell \in mQ_\ell \pm \sqrt{\frac{mQ_\ell(1-Q_\ell)}{\gamma}}$ with probability at least $1-\gamma$. When this occurs,

$$\lambda' = (m-S_\ell)\tfrac{q_\ell}{1-Q_\ell} = \tfrac{mq_\ell}{1-Q_\ell}\left(1 - Q_\ell \pm \sqrt{\tfrac{Q_\ell(1-Q_\ell)}{m\gamma}}\right) = mq_\ell \pm \sqrt{mq_\ell}\cdot\sqrt{\tfrac{q_\ell Q_\ell}{\gamma(1-Q_\ell)}}\,.$$

Thus, $\lambda' = \lambda + \sqrt{\frac{q_\ell Q_\ell}{\gamma(1-Q_\ell)}}\sqrt{\lambda}$ with probability at least $1-\gamma$. The statistical difference between $\mathrm{Po}(\lambda)$ and $\mathrm{Po}(\lambda')$ is $O\big(\sqrt{\frac{q_\ell Q_\ell}{\gamma(1-Q_\ell)}}\big)$, by Lemma B.1, Item (4).

Putting these approximations together: we can replace Z_ℓ by an independent copy of itself, Z'_ℓ, and change the distribution on the vector $(Z_1,...,Z_\ell)$ by at most $\gamma + 2q_\ell + O(\sqrt{\frac{q_\ell Q_\ell}{\gamma(1-Q_\ell)}})$. This is minimized when $\gamma = \sqrt[3]{\frac{q_\ell Q_\ell}{1-Q_\ell}}$. In symbols:

$$(Z_1,\ldots,Z_{\ell-1},Z_\ell) \approx_{O\left(\sqrt[3]{\frac{q_\ell Q_\ell}{1-Q_\ell}}\right)} (Z_1,...,Z_{\ell-1},Z'_\ell)\,.$$

Now replace the Z_ℓs with Z'_ℓs one at a time. By the triangle inequality, the distance between $(Z_1,\ldots,Z_{k-1})$ and $(Z'_1,...,Z'_{k-1})$ is at most the sum of the errors introduced at each step. This sum is $O(k\sqrt[3]{\frac{q_\ell Q_\ell}{1-Q_\ell}})$. Since $q_0 \geq \frac{1}{2}$, the total distance is $O\left(k\cdot(1-q_0)^{2/3}\right)$. ■

ECCC ISSN 1433-8092
http://www.eccc.uni-trier.de/eccc
ftp://ftp.eccc.uni-trier.de/pub/eccc
ftpmail@ftp.eccc.uni-trier.de, subject 'help eccc'