Approximate Trace Reconstruction from a Single Trace*

Xi Chen†
Columbia University
xichen@cs.columbia.edu

Anindya De‡
University of Pennsylvania
anindyad@cis.upenn.edu

Chin Ho Lee§
Harvard University
chlee@seas.harvard.edu

Rocco A. Servedio¶
Columbia University
rocco@cs.columbia.edu

Sandip Sinha‖
Columbia University
sandip@cs.columbia.edu

Abstract

The well-known *trace reconstruction problem* is the problem of inferring an unknown source string $x \in \{0,1\}^n$ from independent "traces", i.e. copies of x that have been corrupted by a δ-deletion channel which independently deletes each bit of x with probability δ and concatenates the surviving bits. The current paper considers the extreme data-limited regime in which only a single trace is provided to the reconstruction algorithm. In this setting exact reconstruction is of course impossible, and the question is to what accuracy the source string x can be approximately reconstructed.

We give a detailed study of this question, providing algorithms and lower bounds for the high, intermediate, and low deletion rate regimes in both the worst-case (x is arbitrary) and average-case (x is drawn uniformly from $\{0,1\}^n$) models. In several cases the lower bounds we establish are matched by computationally efficient algorithms that we provide.

We highlight our results for the high deletion rate regime: roughly speaking, they show that

- Having access to a single trace is already quite useful for worst-case trace reconstruction: an efficient algorithm can perform much more accurate reconstruction, given one trace that is even only a few bits long, than it could given no traces at all. But in contrast,
- in the average-case setting, having access to a single trace is provably not very useful: no algorithm, computationally efficient or otherwise, can achieve significantly higher accuracy given one trace that is $o(n)$ bits long than it could with no traces.

*The full version of the paper can be accessed at https://arxiv.org/abs/2211.03292

†Supported by NSF grants CCF-1703925, IIS-1838154, CCF-2106429 and CCF-2107187.

‡Supported by NSF grants CCF-1926872, CCF-1910534 and CCF-2045128.

§Supported by the Croucher Foundation, the Simons Collaboration on Algorithms and Geometry, and Madhu Sudan's and Salil Vadhan's Simons Investigator Awards. Part of this work was done at Columbia University.

¶Supported by NSF grants IIS-1838154, CCF-2106429, CCF-2211238 and by the Simons Collaboration on Algorithms and Geometry.

‖Supported by NSF grants CCF-1714818, CCF-1822809, IIS-1838154, CCF-1617955, CCF-1740833, and by the Simons Collaboration on Algorithms and Geometry.

1 Introduction

The *trace reconstruction problem* [Kal73, Lev01b, Lev01a, BKKM04] is one of the oldest and most basic algorithmic problems involving the deletion channel. In this problem the goal of the reconstruction algorithm is to infer an unknown n-bit source string $x \in \{0,1\}^n$ given access to a source of independent "traces" of x, where a trace of x is a draw from $\mathrm{Del}_\delta(x)$. Here $\mathrm{Del}_\delta(\cdot)$ is the "deletion channel," which independently deletes each bit of x with probability δ and outputs the concatenation of the surviving bits. The goal of the reconstruction algorithm is to correctly reconstruct the source string x using as few traces and as little computation time as possible.

A surge of recent work [MPV14, DOS17, NP17, PZ17, HPP18, HHP18, BCF+19, BCSS19, Cha21a, KMMP19, HPPZ19, Cha21b, NR21, CDL+21b, CDL+21a, CP21, CDL+22] has addressed many different aspects and variants of the trace reconstruction problem. The version described above corresponds to a "worst-case" setting, since the n-bit source string can be completely arbitrary; despite intensive research [HMPW08, DOS17, NP17, Cha21b], the best algorithm known for this problem, for constant deletion rate δ, requires $\exp(\tilde{O}(n^{1/5}))$ traces. Many papers such as [MPV14, PZ17, HPP18, BCSS19, HPPZ19, Cha21a, CDL+21b] have also considered an "average-case" version of the problem in which the source string $\boldsymbol{x} \sim \{0,1\}^n$ is assumed to be a uniform random n-bit string; this average-case problem is known to be significantly easier than the worst-case problem (we refer the reader to [HPPZ19, Cha21a] for state-of-the-art algorithmic results and lower bounds on average-case trace reconstruction at constant deletion rates). Other problem variants which have been studied include "population recovery" versions in which there is a distribution over source strings rather than a single unknown source string [BCF+19, BCSS19, NR21]; the "low deletion rate" ($\delta = o_n(1)$) and "high deletion rate" ($\delta = 1 - o_n(1)$) settings [BKKM04, HMPW08, MPV14, BCF+19, NR21]; and *approximate* trace reconstruction [SDDF18, DRRS21, SB21, GSZ21, CP21, CDK21, CDL+22], in which the goal is only to obtain an approximate rather than an exact reconstruction of the unknown source string x, and which is the focus of the current work.

Prior work on approximate reconstruction from few traces. The best algorithms known for even the easiest versions of exact trace reconstruction, such as the $\delta = O(1/\log n)$, average-case problem setting considered by [BKKM04], typically require a number of traces that grows with n to achieve exact reconstruction.[1] An attractive feature of the recent works [CP21, CDL+22] is that they give provable performance guarantees for *approximate* trace reconstruction even when only *constantly* many traces are available. In more detail, [CDL+22] gave near-matching upper and lower bounds on the best possible reconstruction accuracy that any algorithm can achieve given $M = O(1/\delta)$ traces from $\mathrm{Del}_\delta(\boldsymbol{x})$ in the average-case setting. [CP21] showed that for any constants δ, ε, there is some constant $M = M(\varepsilon, \delta)$ such that an M-trace algorithm can achieve reconstruction error at most ε given traces from $\mathrm{Del}_\delta(\boldsymbol{x})$ in the average-case setting.[2]

Results such as [CP21, CDL+22], which shed light on what can be achieved given constantly many traces, can be particularly valuable in settings where only a severely limited number of traces are available and the goal is to do as well as possible with the data at hand. Such settings motivate the present paper, which, as we now describe, studies trace reconstruction in the ultimate data-constrained regime.

This work: Approximate reconstruction from a single trace. We consider the problem of recovering an unknown n-bit source string x as accurately as possible given only *a single trace* from $\mathrm{Del}_\delta(x)$. Despite the simplicity and naturalness of this problem, it does not seem to have been considered in prior work.

We give a detailed study of this problem, analyzing both the worst-case setting of an arbitrary unknown source string x as well as the average-case setting of a uniform random $\boldsymbol{x} \sim \{0,1\}^n$. In each of these settings we consider both the low ($\delta = o_n(1)$), medium ($\delta = \Theta(1)$), and high ($\delta = 1 - o_n(1)$) deletion rate regimes. In a number of cases we give upper bounds on the approximate reconstruction accuracy that any one-trace algorithm can achieve, which are essentially matched by corresponding one-trace algorithms that we provide. (All of the algorithms we give are computationally efficient.) For some problem variants our upper bounds on the best achievable accuracy extend beyond one-trace algorithms to algorithms that receive multiple traces.

We view our results as a first investigation of one-trace reconstruction, and reiterate that very little was previously known for any of the problem variants that we consider. We describe the state of prior knowledge in the context of different specific problem variants when describe our results in Section 1.1 below.

[1] Indeed, at deletion rate $\delta = O(1/\log n)$, it is easy to see that given a sample of $o(\frac{\log n}{\log\log n})$ traces, with high probability there will be coordinates of the source string that are deleted from all of the traces in the sample.

[2] Several other recent papers [SDDF18, DRRS21, GSZ21, CDK21, SB21] have also studied approximate trace reconstruction, but focusing on different aspects that make them less relevant to the present paper; see [CDL+22] for a detailed discussion of those works.

1.1 Our results. We are interested in the abilities and limitations of algorithms A which receive as input a single trace y from an unknown n-bit source string x and which output an n-bit hypothesis string $\widehat{x} = A(y)$. We measure the accuracy of $\widehat{x}$ with respect to x by the length of the longest common subsequence $|\mathsf{LCS}(x, \widehat{x})|$. LCS is closely related to edit distance, since if $|\mathsf{LCS}(x, \widehat{x})| = n - k$ for two n-bit strings x and $\widehat{x}$, then $\widehat{x}$ can be converted into x by a sequence of k deletions and k insertions (and this is best possible). The goal of an approximate reconstruction algorithm in this setting is to output a hypothesis string $\widehat{x}$ for which the expectation[3] of $|\mathsf{LCS}(x, \widehat{x})|$ is guaranteed to be as large as possible; thus positive (algorithmic) results in our setting yield *lower bounds* on how large an expected value of $|\mathsf{LCS}(x, \widehat{x})|$ can be achieved, while impossibility results for algorithms give *upper bounds* on the best achievable expected $|\mathsf{LCS}(x, \widehat{x})|$.

As alluded to earlier, we consider both the setting of a worst-case (arbitrary) $x \in \{0,1\}^n$ and the setting of a uniform random $\boldsymbol{x} \sim \{0,1\}^n$. We note that algorithmic results (lower bounds on $\mathbf{E}[|\mathsf{LCS}(x, \widehat{x})|]$ for the worst-case setting carry over to the average-case setting, while impossibility results (upper bounds on $\mathbf{E}[|\mathsf{LCS}(x, \widehat{x})|]$) for the average-case setting carry over to the worst-case setting.

Section 1.1.1 presents our results for the worst-case setting and Section 1.1.2 presents our results for the average-case setting. In each of these sections we first present our results (upper and lower bounds) for the high and medium deletion rate regimes, and then the low deletion rate regime.

1.1.1 Worst-case one-trace reconstruction. We first consider the high deletion rate regime. It is convenient to let $\rho := 1 - \delta$ denote the *retention rate*, so in the high deletion rate regime we have $\rho = o(1)$.

If ρ is too small (as a function of n) then it is easy to see that no nontrivial performance is possible. In particular, if $\rho = o(1/n)$, then by Markov's inequality with probability $1 - o(1)$ a trace $\boldsymbol{y} \sim \mathrm{Del}_\delta(x)$ is zero bits long, and in this case a reconstrution algorithm cannot even distinguish between the two possibilities $x = 0^n$ and $x = 1^n$. Consequently, if $\rho = o(1/n)$ then the largest expected LCS achievable by a one-trace algorithm is at most $(1/2 + o_n(1))n$ (and $n/2$ is trivially achieved by outputting any string with an equal number of 0's and 1's).

Our first positive result shows that — perhaps surprisingly — if ρ is only slightly larger, then it is already possible to do much better than the above trivial bound:

THEOREM 1.1. (WORST-CASE ALGORITHM, SMALL RETENTION RATE, INFORMAL STATEMENT) *For any $\rho = \omega(\log(n)/n)$, there is a worst-case one-trace algorithm that achieves expected LCS at least $(2/3 - o(1))n$. Moreover, for any retention rate $\rho \geq \omega(1/n^{1/3})$, there is a worst-case one-trace algorithm that achieves expected LCS at least $(2/3 + c\rho)n$, where $c > 0$ is an absolute constant.*

The key to Theorem 1.1 is a (to the best of our knowledge novel) notion of an *LCS-cover*, and a simple construction of an extremely small LCS-cover consisting of just two strings. This already suffices to give the first sentence of Theorem 1.1; the second sentence, improving the LCS bound to $(2/3 + c\rho)$, is obtained via a win-win analysis which considers whether or not the single received trace has many "long runs". Roughly speaking, if the trace has many long runs then this indicates that the source string x is highly structured in a way (containing many long segments that are almost all-0 or almost all-1) that makes it easy to achieve a large LCS, and if the trace has few long runs then the source string x must have many 01 alternations, which can be leveraged to get an LCS larger than $2n/3$.

Theorem 1.1 can be viewed as saying that having a $\log n$-bit trace already makes it possible to achieve an LCS of at least $(2/3 - o(1))n$. Complementing Theorem 1.1, we show that even having a $n^{0.999}$-bit trace does not make it possible to achieve an LCS of $(2/3 + c)n$ for any $c > 0$:

THEOREM 1.2. (WORST-CASE UPPER BOUND ON ANY ALGORITHM, SMALL RETENTION RATE, INFORMAL STATEMENT) *Fix any $\varepsilon > 0$. For retention rate $\rho = 1/n^\varepsilon$, no one-trace algorithm can achieve expected LCS greater than $(2/3 + o(1))n$ in the worst-case setting.*

See Theorem 3.2 for a detailed theorem statement, which extends Theorem 1.2 to give an upper bound on the performance of algorithms that receive multiple traces. Theorem 1.2 leverages a recent deep result of Guruswami, Haeupler, and Shahrasbi [GHS20] analyzing a code due to Bukh and Ma [BM14]. We take advantage of the highly

[3] In the worst-case setting this expectation is over the random draw of the trace $\boldsymbol{y}$ from $\mathrm{Del}_\delta(x)$; in the average-case setting, this expectation is also over the uniform random draw of the source string $\boldsymbol{x} \sim \{0,1\}^n$. We give more details and a precise formulation in Section 2.2.

repetitive structure of the Bukh–Ma codewords to combine the [GHS20] result with a construction of a family of distributions over Bukh–Ma codes such that the k-*decks*[4] of all of the different distributions coincide. This in turn lets us show that a single trace does not have enough information to make more accurate reconstruction than (essentially) LCS $2n/3$ possible.

Theorem 1.1 sheds light on the high deletion rate and medium deletion rate regimes of one-trace reconstruction. Turning to the medium and low deletion rate regimes, if the retention rate ρ is large enough (at least some absolute constant), then the algorithm used for Theorem 1.2 is no longer best possible, since it would be better to simply output any string $\widehat{x}$ that contains the trace $\boldsymbol{y}$ as a subsequence. This is because, as observed in [CDL+22], any such string $\widehat{x}$ achieves expected LCS at least $\mathbf{E}[|\boldsymbol{y}|] = \rho n = (1-\delta)n$.

Can better performance than this naive $(1-\delta)n$-length LCS be achieved in the medium and low deletion rate regimes? We give an improvement by constructing a hypothesis string $\widehat{\boldsymbol{x}}$ that randomly intermingles random bits with the bits of $\boldsymbol{y}$. A careful analysis of the LCS between this $\widehat{\boldsymbol{x}}$ and the source string x yields the following:

THEOREM 1.3. (WORST-CASE ALGORITHM, SMALL DELETION RATE, INFORMAL STATEMENT) *There is a worst-case one-trace algorithm that achieves expected LCS at least* $(1-\delta+\delta^2/2-\delta^3/2+\delta^4/2-\delta^5/2-o(1))n$ *for deletion rate* δ.

Given Theorem 1.3, it is natural to ask about limitations of one-trace reconstruction in the low deletion rate regime. Taking $M = 1$ in the main lower bound result (Theorem 1.2) of [CDL+22], that result shows that no one-trace algorithm can achieve expected LCS greater than $(1-\delta^C)n$ in the worst-case setting, where C is some absolute (large) constant. In Section 1.1.2 we will see that Theorem 1.6 establishes a stronger and near-optimal bound even for the more challenging average-case setting.

1.1.2 Average-case one-trace reconstruction. The average-case setting of one-trace reconstruction turns out to present some unexpected challenges due to connections with difficult unresolved problems in the combinatorics of words. To see this, let us first consider the problem of average-case trace reconstruction from *zero* traces; so the reconstruction algorithm receives no input at all, and simply aims to output the n-bit string $\widehat{x}$ which maximizes the expected value of $|\mathsf{LCS}(\widehat{x}, \boldsymbol{x})|$ across uniform random $\boldsymbol{x} \sim \{0,1\}^n$. In contrast with the worst-case setting (where no zero-trace algorithm can achieve expected LCS better than $1/2$ because the source string x could be chosen uniformly at random from $\{0^n, 1^n\}$), in the average-case setting the hypothesis string $\widehat{x} = (01)^{n/2}$ already achieves $\mathbf{E}[|\mathsf{LCS}(\widehat{x}, \boldsymbol{x})|] \geq (3/4 - o(1))n$: first greedily match the 0's in $\widehat{x}$ with the 0's in $\boldsymbol{x}$ from left to right, and then opportunistically augment these $\approx n/2$ matching edges with edges matching pairs of 1's where possible. So nontrivial performance is possible, even with zero traces, in the average-case setting.

Can we do better with a smarter choice of the hypothesis string $\widehat{x}$? A natural idea is to select $\widehat{\boldsymbol{x}}$ uniformly at random from $\{0,1\}^n$. The performance of this zero-trace algorithm is captured by the *Chvátal–Sankoff constant*

$$\gamma_2 := \lim_{n\to\infty} \frac{\mathbf{E}_{\boldsymbol{x},\widehat{\boldsymbol{x}}\sim\{0,1\}^n}[|\mathsf{LCS}(\boldsymbol{x}, \widehat{\boldsymbol{x}})|]}{n}$$

(the "2" is because we are working with the binary alphabet); the existence of this limit is an easy consequence of the superadditivity of LCS between random strings (using Fekete's Lemma [Wik22b]). Despite much investigation over more than 40 years, the value of γ_2 is not known: in 1975 Chvátal and Sankoff showed that $0.727273 \leq \gamma_2 \leq 0.866595$, and the current state of the art bounds, due to Lueker [Lue09], are that $0.788071 \leq \gamma_2 \leq 0.826280$ [Wik22a].

A superadditivity argument similarly establishes the existence of the limit

$$c_2 := \lim_{n\to\infty} \max_{\widehat{x}\in\{0,1\}^n} \frac{\mathbf{E}_{\boldsymbol{x}\sim\{0,1\}^n}[|\mathsf{LCS}(\boldsymbol{x}, \widehat{x})|]}{n}, \tag{1.1}$$

which corresponds to the performance of the information-theoretic optimal zero-trace algorithm for the average-case setting. Even less is known about c_2 than γ_2; Bukh and Cox [BC22] have shown (via an involved argument and an automated search) that $c_2 \geq 0.82118$, and we show in Section A that $c_2 \leq 0.88999$, but more detailed

[4]The k-deck of a single string is the multiset of all length-k subsequences of the string, and the k-deck of a distribution over strings is the corresponding mixture of k-decks of the constituent strings in the mixture; see Section 3.2.1 for detailed definitions.

bounds on the value of c_2 do not seem to be known, nor is it known what strings might achieve this optimal bound [Buk22].

Given these challenges in understanding zero-trace reconstruction in the average-case setting, the prospects of analyzing one-trace average-case reconstruction may appear dim. Perhaps surprisingly, for the low deletion rate regime and medium deletion rate regime it turns out that the difficulty of analyzing zero-trace reconstruction is the only barrier to showing an upper bound on average-case one-trace reconstruction. This is shown in the following theorem, which gives an upper bound on average-case one-trace reconstruction in terms of the quantity c_2 from Equation (1.1):

THEOREM 1.4. (AVERAGE-CASE UPPER BOUND ON ANY ALGORITHM, SMALL RETENTION RATE, INFORMAL STATEMENT) *Let $L_{1,\mathrm{avg}}(\delta, n)$ denote the best expected LCS achievable by any one-trace algorithm at deletion rate δ in the average-case setting. Then we have $c_2 \leq \lim_{n\to\infty} \frac{L_{1,\mathrm{avg}}(\delta,n)}{n} \leq c_2 + \rho$.*

Theorem 1.4 tells us that for any $\rho = o_n(1)$ retention rate, it is not possible to asymptotically improve on the performance of the best zero-trace algorithm. In fact, in Theorem 5.1 we give a generalization of Theorem 1.4 which gives an upper bound on the performance of algorithms that receive more than one trace. The proof is based on a careful analysis, using a coupling argument, of the *a posteriori* distribution of the random source string $\boldsymbol{x}$ given the received collection of traces.

Finally, we consider upper and lower bounds which are applicable for the medium and small deletion rate regime. In the average-case setting, the algorithm of Theorem 1.3 can be shown to have better performance than was established in Theorem 1.3 for the worst-case setting:

THEOREM 1.5. (AVERAGE-CASE ALGORITHM, SMALL DELETION RATE, INFORMAL STATEMENT) *There is an average-case one-trace algorithm that achieves expected LCS at least $\left(1 - \delta + \frac{1}{2}\delta^2 + \frac{17}{8}\delta^4 + \frac{55}{8}\delta^5 - o(1)\right) n$ for deletion rate δ.*

Given Theorem 1.5, it is natural to investigate the best possible performance of any one-trace algorithm in the average-case setting for small δ. A relatively simple probabilistic argument (which is based on a union bound across all possible matchings, and which we give in Section B) shows that the expected LCS achieved by any one-trace algorithm can be at most $(1 - \Omega(\delta/\log(1/\delta))) \cdot n$. Via a more involved probabilistic argument we strengthen this to a $1 - \Theta(\delta)$ bound:

THEOREM 1.6. (AVERAGE-CASE UPPER BOUND ON ANY ALGORITHM, SMALL RETENTION RATE, INFORMAL STATEMENT) *For any deletion rate $\delta = \omega(1/n)$, no one-trace algorithm can achieve expected LCS greater than $(1 - c\delta)n$ in the average-case setting, where c is some absolute constant.*

We observe that by virtue of Theorem 1.5, Theorem 1.6 is best possible up to the hidden multiplicative constant on the δ-term.

1.2 Future work. A natural first goal for future work is to obtain sharper results. For example, if the deletion rate δ is 0.1, what is the largest constant c such that expected LCS cn can be achieved in the worst-case setting? In the average-case setting? What if $\delta = 0.9$? We do not currently have sharp answers to questions such as these.

A different goal is to go beyond one-trace reconstruction. While our negative results Theorem 1.2 and Theorem 1.4 extend to algorithms that receive multiple traces, it would be interesting to extend positive results such as Theorem 1.1, Theorem 1.3 and Theorem 1.5 to the setting of multiple traces. In this context we mention the work of Chakraborty et al. [CDK21], which gave an average-case algorithm for approximate trace reconstruction from three traces in an insertion-deletion model.

2 Preliminaries

Notation. Given a positive integer n, we write $[n]$ to denote $\{1, \ldots, n\}$. Given two integers $a \leq b$ we write $[a : b]$ to denote $\{a, \ldots, b\}$. We write ln to denote natural logarithm and log to denote logarithm to the base 2. We denote the set of non-negative integers by $\mathbb{Z}_{\geq 0}$. We write "$a = b \pm c$" to indicate that $b - c \leq a \leq b + c$. It will be convenient for us to index a binary string $x \in \{0,1\}^n$ using $[1 : n]$ as $x = (x_1, \ldots, x_n)$.

Distributions. When we use bold font such as $\mathbf{D}, \boldsymbol{y}, \boldsymbol{z}$, etc., it indicates that the entity in question is a random variable. We write "$\boldsymbol{r} \sim \mathcal{P}$" to indicate that random variable $\boldsymbol{r}$ is distributed according to probability distribution $\mathcal{P}$. If S is a finite set we write "$\boldsymbol{r} \sim S$" to indicate that $\boldsymbol{r}$ is distributed uniformly over S.

We write $\mathsf{Geometric}(\rho)$ to denote the geometric distribution with parameter ρ, i.e. the number of Bernoulli trials with success probability ρ needed to get one success, supported on the set $\{1, 2, \dots\}$. We will use the following tail bound for sums of independent geometric random variables:

CLAIM 2.1. *Let $\rho \in [0,1]$ and let $\mathbf{G}_1, \dots, \mathbf{G}_m$ be m independent geometric random variables with each $\mathbf{G}_i \sim \mathsf{Geometric}(\rho)$. For any $\gamma \in [0,1]$, we have*

$$\mathbf{Pr}\left[\left|\sum_{i=1}^{m} \mathbf{G}_i - \rho^{-1}m\right| \geq \gamma\rho^{-1}m\right] \leq e^{-\Omega(\gamma^2 m)}.$$

Proof. By coupling $(\mathbf{G}_1, \dots, \mathbf{G}_m)$ with a draw from the Binomial distribution $\mathrm{Bin}(n, \rho)$, we observe that $\sum_{i=1}^{m} \mathbf{G}_i \geq n$ if and only if $\mathrm{Bin}(n, \rho) < m$. Let $n_h := (1+\gamma)\rho^{-1}m$ and $n_\ell := (1-\gamma)\rho^{-1}m$. We have

$$\begin{aligned}
&\mathbf{Pr}\left[\left|\sum_{i=1}^{m} \mathbf{G}_i - \rho^{-1}m\right| \geq \gamma\rho^{-1}m\right] \\
&= \mathbf{Pr}\Big[\Big(\mathrm{Bin}\big((1+\gamma)\rho^{-1}m, \rho\big) < m\Big) \vee \Big(\mathrm{Bin}\big((1-\gamma)\rho^{-1}m, \rho\big) > m\Big)\Big] \\
&= \mathbf{Pr}\left[\left(\mathrm{Bin}\big(n_h, \rho\big) < \frac{1}{1+\gamma} \cdot \rho n_h\right) \vee \left(\mathrm{Bin}\big(n_\ell, \rho\big) > \frac{1}{1-\gamma} \cdot \rho n_\ell\right)\right] \\
&= \mathbf{Pr}\left[\left(\mathrm{Bin}\big(n_h, \rho\big) < \left(1 - \frac{\gamma}{1+\gamma}\right)\rho n_h\right) \vee \left(\mathrm{Bin}\big(n_\ell, \rho\big) > \left(1 + \frac{1}{1-\gamma}\right)\rho n_\ell\right)\right] \\
&\leq e^{-\Omega(\gamma^2 m)},
\end{aligned}$$

where the inequality is a standard Chernoff bound. ∎

Deletion channel and traces. Throughout this paper the parameter $0 < \delta < 1$ denotes the *deletion probability*. Given a string $x \in \{0,1\}^n$, we write $\mathrm{Del}_\delta(x)$ to denote the distribution of the string that results from passing x through the δ-deletion channel (so the distribution $\mathrm{Del}_\delta(x)$ is supported on $\{0,1\}^{\leq n}$), and we refer to a string in the support of $\mathrm{Del}_\delta(x)$ as a *trace* of x. Recall that a random trace $\boldsymbol{y} \sim \mathrm{Del}_\delta(x)$ is obtained by independently deleting each bit of x with probability δ and concatenating the surviving bits. [5] We may view the draw of a trace $\boldsymbol{y}$ from $\mathrm{Del}_\delta(x)$ as a two-step process: first a set $\mathbf{D}$ of *deletion locations* is obtained by including each element of $[n]$ independently with probability δ, and then $\boldsymbol{y}$ is set to be $x_{[n]\setminus\mathbf{D}}$.

LCS and matchings. We write $\mathsf{LCS}(x, x')$ to denote the longest common subsequence between two strings x and x' and $|\mathsf{LCS}(x, x')|$ to denote its length. A *matching* M between two strings $x, x' \in \{0,1\}^*$ is a list of pairs $(v_1, v_1'), (v_2, v_2'), \dots$ such that $v_1 \leq v_2 \leq \cdots$, $v_1' \leq v_2' \leq \cdots$, and for every t we have $x_{v_t} = x'_{v_t'}$. The *size* of a matching is the number of pairs. We note that the largest matching between x and x' is of length $|\mathsf{LCS}(x, x')|$.

An asymptotic bound on binomial coefficients. We recall the following standard bound on binomial coefficients:

FACT 2.1. ([vL82], THEOREM 1.4.5) *For $0 \leq k \leq n/2$, we have $\sum_{i=0}^{k} \binom{n}{i} \leq 2^{H(k/n)n}$, where $H(x) = x\log(1/x) + (1-x)\log(1/(1-x))$ is the binary entropy function.*

2.1 The average-case setting.

We record the following simple observation, which is useful for analyses of the average-case setting:

OBSERVATION 2.1. (*A posteriori* DISTRIBUTION OF A UNIFORM RANDOM SOURCE STRING GIVEN ONE TRACE) *Let $\boldsymbol{x}$ be a uniform random source string from $\{0,1\}^n$. Given any fixed outcome $y \in \{0,1\}^m$ of a single trace $y = \boldsymbol{y} \sim \mathrm{Del}_\delta(\boldsymbol{x})$, the* a posteriori *distribution of $\boldsymbol{x}$ given y is as follows:*

[5] In this work we assume that the deletion probability δ is known to the reconstruction algorithm.

1. *Draw a uniform random m-element subset* $\mathbf{S} \sim \binom{[n]}{[m]}$ *of* $[n]$ *(say* $\mathbf{S} = \{\boldsymbol{s}_1, \ldots, \boldsymbol{s}_m\}$ *where* $1 \le \boldsymbol{s}_1 < \cdots < \boldsymbol{s}_m \le n$*);*

2. *For each* $i \in [m]$ *set* $\boldsymbol{x}_{\boldsymbol{s}_i} = y_i$ *(i.e. fill in the locations in* $\mathbf{S}$ *from left to right with the bits of* y*), and for each* $j \notin \mathbf{S}$ *set* $\boldsymbol{x}_j$ *to an independent uniform element of* $\{0,1\}$*.*

We write "$\boldsymbol{x} \sim y$" to indicate that $\boldsymbol{x}$ has the distribution described above. We note that a somewhat counterintuitive corollary of Observation 2.1 is the following: in the average-case setting (when $\boldsymbol{x}$ is uniform random), even if the received trace is the string 1^m, the *a posteriori* distribution of the $n-m$ "unseen bits" of $\boldsymbol{x}$ is that they are independent and uniform random.

An easy corollary of Observation 2.1 is the following:

COROLLARY 2.1. *For* $\boldsymbol{x}$ *a uniform random source string from* $\{0,1\}^n$*, given any fixed outcome* $y \in \{0,1\}^m$ *of a single trace* $y = \boldsymbol{y} \sim \mathrm{Del}_\delta(\boldsymbol{x})$*, the* a posteriori *distribution of the other* $n - |y|$ *bits* $\boldsymbol{x}_{\mathbf{D}}$ *of* $\boldsymbol{x}$ *is that they are distributed as a uniform random element of* $\{0,1\}^{[n]\setminus|y|}$*.*

2.2 One-trace and few-trace algorithms.

Optimal worst-case algorithms. We introduce the notation $L_{1,\mathrm{worst}}(\delta, n)$ to denote the largest possible LCS that can be achieved in expectation by any one-trace algorithm under deletion rate δ in the worst-case setting, i.e.

$$L_{1,\mathrm{worst}}(\delta, n) := \max_A \min_{x \in \{0,1\}^n} \mathop{\mathbf{E}}_{\boldsymbol{y} \sim \mathrm{Del}_\delta(x)} [|\mathsf{LCS}(A(\boldsymbol{y}, n), x)|], \tag{2.2}$$

where the maximum is taken over all algorithms A that take as input the values n, δ and a single trace $\boldsymbol{y}$, and output an n-bit hypothesis string (denoted $A(\boldsymbol{y}, n)$ in the expression above). We observe that (2.2) could be extended to allow the algorithm A to be randomized (and have the expectation be also over the randomness of A), but we do not do this since the optimal algorithm in (2.2) can without loss of generality be taken to be deterministic.

We will sometimes consider the optimal performance of t-trace algorithms for $t > 1$, so we extend the above definition in the obvious way to algorithms that are given t independent traces, i.e.

$$L_{t,\mathrm{worst}}(\delta, n) := \max_A \min_{x \in \{0,1\}^n} \mathop{\mathbf{E}}_{\boldsymbol{y}^{(1)}, \ldots, \boldsymbol{y}^{(t)} \sim \mathrm{Del}_\delta(x)} [|\mathsf{LCS}(A(\boldsymbol{y}^{(1)}, \ldots, \boldsymbol{y}^{(t)}, n), x)|]. \tag{2.3}$$

Optimal average-case algorithms. We use similar notation to capture the optimal performance of one-trace and t-trace algorithms in the average-case setting:

$$L_{1,\mathrm{avg}}(\delta, n) := \max_A \mathop{\mathbf{E}}_{\boldsymbol{x} \sim \{0,1\}^n} \mathop{\mathbf{E}}_{\boldsymbol{y} \sim \mathrm{Del}_\delta(\boldsymbol{x})} [|\mathsf{LCS}(A(\boldsymbol{y}, n), \boldsymbol{x})|], \tag{2.4}$$

$$L_{t,\mathrm{avg}}(\delta, n) := \max_A \mathop{\mathbf{E}}_{\boldsymbol{x} \sim \{0,1\}^n} \mathop{\mathbf{E}}_{\boldsymbol{y}^{(1)}, \ldots, \boldsymbol{y}^{(t)} \sim \mathrm{Del}_\delta(\boldsymbol{x})} [|\mathsf{LCS}(A(\boldsymbol{y}^{(1)}, \ldots, \boldsymbol{y}^{(t)}, n), \boldsymbol{x})|]. \tag{2.5}$$

3 Worst-case one-trace reconstruction, small retention rate

3.1 An efficient algorithm. We prove Theorem 1.1 in this subsection. We start with the first part of Theorem 1.1, i.e., when the retention rate ρ is large enough ($\omega(\log(n)/n)$) that a nontrivial number of bits are expected to be present in a random trace, then a simple computationally efficient one-trace algorithm can achieve an LCS significantly better than $n/2$.

3.1.1 A useful structural result and a $(2/3 - o(1))$-LCS algorithm for $\rho = \omega(\log(n)/n)$. It is helpful for us to consider the following preliminary problem: we are not given any traces, and the goal is to output *a list* of m-bit candidates such that the unknown source string $x \in \{0,1\}^n$ has large LCS with one of the candidate strings in our list. This motivates the following definition:

DEFINITION 3.1. (LCS-COVER) *Let m and n be two positive integer. We say a set $S \subseteq \{0,1\}^m$ is an h-*LCS cover *for strings of length n if for every $x \in \{0,1\}^n$ we have*

$$\big|\mathsf{LCS}(S,x)\big| := \max_{s\in S}\big|\mathsf{LCS}(s,x)\big| \geq h.$$

The following simple claim shows that when m is within a factor of two of n, there is a (perhaps surprisingly) good LCS cover consisting of at most two strings:

CLAIM 3.1. *For every $m \in [n/2, 2n]$, there exists a $((n+m)/3)$-LCS-cover of size at most 2.*

Proof. We first consider the extreme settings of $m = 2n$ and $m = n/2$. When $m = 2n$, we have $|\mathsf{LCS}((01)^n, x)| = n$ for every $x \in \{0,1\}^n$, and thus $\{(01)^n\}$ is an n-LCS cover (of size 1). When $m = n/2$, every $x \in \{0,1\}^n$ either contains $n/2$ many 1s or this many 0s, and so either $|\mathsf{LCS}(0^{n/2}, x)| \geq n/2$ or $|\mathsf{LCS}(1^{n/2}, x)| \geq n/2$, and hence the set $\{0^{n/2}, 1^{n/2}\}$ is a $(n/2)$-LCS cover of size 2.

We interpolate between these two cases to handle general m's. Write $m = 2a + b$ and $n = a + 2b$ for some a and b (so $a = m - (n+m)/3$ and $b = n - (n+m)/3$). Consider

$$S := \big\{(01)^a 0^b, (01)^a 1^b\big\} \subseteq \{0,1\}^m. \tag{3.6}$$

Given $x \in \{0,1\}^n$, we can write $x = x_1 \circ x_2$ where $x_1 \in \{0,1\}^a$ and $x_2 \in \{0,1\}^{2b}$, and we get that

$$\big|\mathsf{LCS}(S,x)\big| \geq \Big|\mathsf{LCS}\big((01)^a, x_1\big)\Big| + \Big|\mathsf{LCS}\big(\{0^b, 1^b\}, x_2\big)\Big| \geq a + b = (n+m)/3.$$

∎

We observe that taking $m = n$ in Equation (3.6), we have a $(2n/3)$-LCS cover consisting of the two strings $(01)^{n/3}0^{n/3}$ and $(01)^{n/3}1^{n/3}$. This suggests a one-trace algorithm that returns an n-bit string $\widehat{x}$ that achieves $|\mathsf{LCS}(\widehat{x}, x)| \geq (2/3 - o(1))n$ with high probability when $\rho = \omega(\log n/n)$: to determine which one of the two n-bit strings $(01)^{n/3}0^{n/3}, (01)^{n/3}1^{n/3}$ to output, it simply needs to determine (with high probability) from the trace $\boldsymbol{y} \sim \mathrm{Del}_\delta(x)$ whether the majority of the last $2n/3$ bits of the unknown x is 0 or 1, which can be done (to accuracy $o(1)$) by simply taking the majority of the last $2\rho n/3$ bits of y. A routine computation now gives the first sentence of Theorem 1.1.

We further note that the simple $(2n/3)$-LCS cover given by $\{(01)^{n/3}0^{n/3}, (01)^{n/3}1^{n/3}\}$ is essentially best possible among all covers of constant size; more precisely, for any positive constant ε, any $(2/3+\varepsilon)n$-LCS cover must have size $\Omega(\log n)$. This is a consequence of a recent result of Guruswami, Haeupler, and Shahrasbi [GHS20]; we give the proof in Section C.

3.1.2 A $(2/3 + \Omega(\rho))$-LCS algorithm for $\rho = \omega(1/n^{1/3})$. Next we prove the second part of Theorem 1.1. It follows from the following theorem:

THEOREM 3.1. (WORST-CASE ALGORITHM, SMALL RETENTION RATE) *There exists an absolute constant $c > 0$ such that the following holds. Let the retention rate $\rho := \rho(n) = 1 - \delta(n)$ such that $\rho = \omega(1/n^{1/3})$. There is an $O(n)$-time algorithm A which is given as input the values n, δ, and a single trace $\boldsymbol{y} \sim \mathrm{Del}_\delta(x)$, where $x \in \{0,1\}^n$ is an unknown source string. With probability at least $1 - e^{-\Omega(\rho^3 n)}$ over the randomness of $\boldsymbol{y} \sim \mathrm{Del}_\delta(x)$, A outputs a hypothesis string $\widehat{\boldsymbol{x}} \in \{0,1\}^n$ satisfying*

$$\big|\mathsf{LCS}(\widehat{\boldsymbol{x}}, x)\big| \geq (2/3 + c\rho) \cdot n.$$

An easy computation using the high-probability bound provided by Theorem 3.1 shows that if $\rho \geq \omega(1/n^{1/3})$, then we get that $L_{1,\mathrm{worst}}(\delta, n) \geq (2/3 + \Omega(\rho)) \cdot n$, giving the bound on expected LCS that is claimed in Theorem 1.1.

The algorithm for Theorem 3.1 improves on the $(2/3 - o(1))n$-LCS algorithm described in Section 3.1.1. The high-level idea is to do better than the $(n+m)/3$ benchmark given by Claim 3.1 on the $(n/3)$-prefix $x^{(1)}$ of x. For intuition, suppose we could find an $\widehat{x}^{(1)} \in \{0,1\}^*$ such that

$$\big|\mathsf{LCS}(\widehat{x}^{(1)}, x^{(1)})\big| \geq \frac{|\widehat{x}^{(1)}| + |x^{(1)}|}{3} + \varepsilon n.$$

Then we could potentially apply the approach of the one-trace algorithm from the previous subsection on the remaining bits of x, and outputs $\widehat{x}$ that extends $\widehat{x}^{(1)}$ to achieve an LCS of roughly

$$\frac{|\widehat{x}^{(1)}| + |x^{(1)}|}{3} + \varepsilon n + \frac{(n - |\widehat{x}^{(1)}|) + (n - |x^{(1)}|)}{3} = \frac{2n}{3} + \varepsilon n.$$

We now discuss how to beat the $(n+m)/3$ benchmark in more detail. Let $L = [\rho n/3]$ and y_L be the $(\rho n/3)$-length prefix of the trace y. We divide y_L into blocks of size 2000. If a block contains only 0s, then it is very likely (probability at least, say, 0.9) that there is a corresponding subword in x of size about $2000/\rho$ that contains mostly 0s; such a subword has large LCS (say, at least $1999/\rho$) with the string $0^{2000/\rho}$. So if most blocks contain only 0s or only 1s (Case 2 in the description of Algorithm A given below), then by outputting an $\widehat{x}^{(1)}$ which is a corresponding sequence of $0^{2000/\rho}$'s and $1^{2000/\rho}$'s, such an $\widehat{x}^{(1)}$ will have an LCS with $x^{(1)}$ that is much larger than $(|x^{(1)}| + |\widehat{x}^{(1)}|)/3$.

On the other hand, if most blocks contain both a 0-bit and a 1-bit, then we know that the string $x^{(1)}$ must alternate between 0s and 1s at least $t := \Omega(\rho n)$ times. In this case (Case 1 in the algorithm description), we can use the shorter string $(01)^{n/3-t}$ to achieve an LCS of size $n/3$ with $x^{(1)}$, which also gives us an $\Omega(\rho n)$ savings.

The rest of Section 3 gives a formal proof of Theorem 3.1.

3.1.3 The Algorithm A. In this subsection we describe the algorithm A to prove Theorem 3.1. Let $\gamma := \rho/720000$. We show that given a trace $\boldsymbol{y} \sim \mathrm{Del}_\delta(x)$ for any unknown $x \in \{0,1\}^n$, the algorithm A returns $\widehat{\boldsymbol{x}}$ satisfying

$$\left|\mathsf{LCS}(x, \widehat{\boldsymbol{x}})\right| \geq \frac{2n}{3} + \frac{\rho n}{90000} - 4\gamma n \tag{3.7}$$

with probability at least $1 - e^{-\Omega(\gamma^2 \rho n)}$. Setting $c = 1/180000$ in Theorem 3.1 finishes the proof.

Given a trace y of $x \in \{0,1\}^n$, A outputs $\widehat{x} := 0^n$ if its input trace y has $|y| < (\rho - \gamma)n$. We refer to this case as Case 0; henceforth we will assume $|y| \geq (\rho - \gamma)n$ below.

Let $L := [\rho n/3]$ and y_L be the first $\rho n/3$ bits of y. Divide y_L into $B := \rho n/6000$ many blocks $y_{L_1}, \ldots, y_{L_B}$ of length 2000 each (so $L_i := \{2000(i-1)+1, \ldots, 2000i\}$). Algorithm A identifies the y_{L_i}'s that contain only 0s or only 1s. Specifically, let

$$B' := \left\{i \in [B] : y_{L_i} = z_i^{2000} \text{ for some } z_i \in \{0,1\}\right\}.$$

There are two cases:

Case 1: $|B'| < 0.8B$**.** (In this case, a significant number of blocks are "not pure.") Let

$$c := \frac{n}{3} - \frac{\rho n}{60000}, \quad a := \frac{\rho n}{45000} \quad \text{and} \quad b := \frac{n}{3} - \frac{\rho n}{90000}. \tag{3.8}$$

Let $z \in \{0,1\}$ be the majority of the last $2\rho b$ bits of y. A outputs the n-bit $\widehat{x} := (01)^{c+a} z^b$.

Case 2: $|B'| \geq 0.8B$**.** (Most blocks are "pure.") Let $z \in \{0,1\}$ be the majority of the last $2\rho n/9$ bits of y. A outputs the following n-bit string

$$\widehat{x} := \widehat{x}^{(1)} \circ (01)^{2n/9} \circ z^{2n/9},$$

where $\widehat{x}^{(1)}$ is the concatenation of $z_i^{2000/\rho}$ for each $i \in [B]$ with z_i being the bit such that $y_{L_i} = z_i^{2000}$ when $i \in B'$ and $z_i = 0$ when $i \notin B'$ (so $\widehat{x}^{(1)}$ has length $n/3$).

3.1.4 Analysis of Algorithm A. Let $x \in \{0,1\}^n$ be the unknown source string. We start by describing an equivalent process of drawing $\boldsymbol{y} \sim \mathrm{Del}_\delta(x)$. Let x_∞ be the infinite string obtained from x by padding infinitely many copies of a special symbol $*$ at the end. Consider sampling an infinite subsequence $\boldsymbol{y}_\infty$ of x_∞ by the following infinite process: For each round $j = 1, 2, \ldots$, we sample a prefix $\boldsymbol{x}^j$ of x_∞ of length $|\boldsymbol{x}^j| \sim \mathsf{Geometric}(\rho)$, then output the last bit of $\boldsymbol{x}^j$ as the j-th bit of $\boldsymbol{y}_\infty$ and delete the prefix $\boldsymbol{x}^j$ from x_∞ before moving on to the next value of j. Finally, we set $\boldsymbol{y}$ to be the longest prefix of $\boldsymbol{y}_\infty$ that does not contain any special symbol $*$. It is easy to check that $\boldsymbol{y}$ drawn from this process is identically distributed to $\mathrm{Del}_\delta(x)$.

We introduce some notation for working with $\boldsymbol{x}^j$ as a byproduct of the above random process of drawing $\boldsymbol{y} \sim \mathrm{Del}_\delta(x)$. For a subset $S \subseteq \mathbb{N}$ (e.g., L introduced in the description of the algorithm), we write $\boldsymbol{x}^S$ to denote the concatenation of $\boldsymbol{x}^j : j \in S$, where $\boldsymbol{x}^j$ is the prefix drawn in the j-th round. Note that the string $\boldsymbol{x}^{[|\boldsymbol{y}|]}$ does not necessarily contain the source string x (it may not contain some of its last few bits) but the string $\boldsymbol{x}^{[|\boldsymbol{y}|+1]}$ always contains x as a prefix.

Let $\boldsymbol{y} \sim \mathrm{Del}_\delta(x)$ be a trace drawn using the process above, and let $\widehat{\boldsymbol{x}}$ be the string returned by the algorithm A when running on $\boldsymbol{y}$. We say A succeeds (on $\boldsymbol{y}$) if $\widehat{\boldsymbol{x}}$ satisfies Equation (3.7) and A fails otherwise. It suffices to show that all three probabilities $\mathbf{Pr}_{\boldsymbol{y}\sim\mathrm{Del}_\delta(x)}[\boldsymbol{y} \text{ in Case 0}]$,

$$\mathbf{Pr}_{\boldsymbol{y}\sim\mathrm{Del}_\delta(x)}\left[\boldsymbol{y} \text{ in Case 1 and } A \text{ fails}\right] \quad \text{and} \quad \mathbf{Pr}_{\boldsymbol{y}\sim\mathrm{Del}_\delta(x)}\left[\boldsymbol{y} \text{ in Case 2 and } A \text{ fails}\right]$$

are at most $e^{-\Omega(\gamma^2\rho n)}$. The upper bound for Case 0 follows by the Chernoff bound (which is indeed $e^{-\Omega(\rho n)}$). Below we analyze the two main cases of the algorithm separately.

Case 1: $|B'| < 0.8B$. Recall from the description of A that we are in Case 1 if $\boldsymbol{y} \sim \mathrm{Del}_\delta(x)$ has length at least $(\rho-\gamma)n$ and $|\mathbf{B}'| < 0.8B$. Recall from Equation (3.8) our choices of a, b and c, and let $\boldsymbol{z} \in \{0,1\}$ be the majority of the last $2\rho b$ bits in $\boldsymbol{y}$. We partition x into $x^{(1)} \circ x^{(2)} \circ x^{(3)}$ with

$$|x^{(1)}| = n/3, \quad |x^{(2)}| = a \quad \text{and} \quad |x^{(3)}| = 2b.$$

Our goal is to show that

$$\mathbf{Pr}_{\boldsymbol{y}\sim\mathrm{Del}_\delta(x)}\left[\boldsymbol{y} \text{ in Case 1 and } A \text{ fails}\right] \le e^{-\Omega(\gamma^2\rho n)}.$$

This follows from the following two claims: Let E_1 denote the event of $|\boldsymbol{x}^L| \ge n/3 + \gamma n$ and E_2 denote the event of $\boldsymbol{z}$ appearing less than $b - \gamma n$ many times in $x^{(3)}$.

Claim 3.2. *For any string $x \in \{0,1\}^n$, we have*

$$\mathbf{Pr}_{\boldsymbol{y}\sim\mathrm{Del}_\delta(x)}\left[\boldsymbol{y} \textit{ in Case } 1 \wedge (E_1 \vee E_2)\right] \le e^{-\Omega(\gamma^2\rho n)}.$$

Claim 3.3. *The algorithm A succeeds whenever $\boldsymbol{y} \sim \mathrm{Del}_\delta(x)$ satisfies (1) $\boldsymbol{y}$ falls in Case 1; (2) $\overline{E_1}$: $|\boldsymbol{x}^L| < n/3 + \gamma n$; and (3) $\overline{E_2}$: $\boldsymbol{z}$ appears at least $b - \gamma n$ many times in $x^{(3)}$.*

Proof. [Proof of Claim 3.2] It follows from Claim 2.1 that the probability of E_1 alone is at most $e^{-\Omega(\gamma^2\rho n)}$. So it suffices to upper bound $\mathbf{Pr}_{\boldsymbol{y}}[\boldsymbol{y} \text{ in Case 1 and } E_2]$. Assume without loss of generality that $x^{(3)}$ has at least $b + \gamma n$ many z's for some $z \in \{0,1\}$; otherwise the probability above is trivially 0. By Chernoff bound we have

$$\mathbf{Pr}_{\boldsymbol{y}\sim\mathrm{Del}_\delta(x)}\left[\#\text{ of bits in } x^{(3)} \text{ that survive in } \boldsymbol{y} \ge 2\rho b + \rho\gamma n/3\right] \le e^{-\Omega(\gamma^2\rho n)} \quad \text{and}$$
$$\mathbf{Pr}_{\boldsymbol{y}\sim\mathrm{Del}_\delta(x)}\left[\#\text{ of } z\text{'s in } x^{(3)} \text{ that survive in } \boldsymbol{y} \le \rho b + 2\rho\gamma n/3\right] \le e^{-\Omega(\gamma^2\rho n)}.$$

So with probability at least $1 - e^{-\Omega(\gamma^2\rho n)}$, the number of bits in $x^{(3)}$ that survive in $\boldsymbol{y}$ is at most $2\rho b + \rho\gamma n/3$ and among them at least $\rho b + 2\rho\gamma n/3$ bits are z. In this case it cannot happen that $\boldsymbol{y}$ falls in Case 1 and $\boldsymbol{z} \ne z$. It follows that $\mathbf{Pr}_{\boldsymbol{y}}[\boldsymbol{y} \text{ in Case 1 and } E_2] \le e^{-\Omega(\gamma^2\rho n)}$. ∎

Proof. [Proof of Claim 3.3] When $\boldsymbol{y} \sim \mathrm{Del}_\delta(x)$ falls in Case 1, the string $\widehat{\boldsymbol{x}}$ returned by A is

$$\widehat{\boldsymbol{x}} := (01)^c \circ (01)^a \circ \boldsymbol{z}^b.$$

We lowerbound $|\mathsf{LCS}(x, \widehat{\boldsymbol{x}})|$ by

$$\left|\mathsf{LCS}(x^{(1)}, (01)^c)\right| + \left|\mathsf{LCS}(x^{(2)}, (01)^a)\right| + \left|\mathsf{LCS}(x^{(3)}, \boldsymbol{z}^b)\right|$$

and below we bound each of the three terms separately. The following simple fact will be useful:

Fact 3.1. *Suppose $x \in \{0,1\}^n$ has t many disjoint 01's. Then $\mathsf{LCS}(x, (01)^m) = n$ when $m \ge n - t$.*

We start with $|\mathsf{LCS}(x^{(1)}, (01)^c)|$. Because $|\mathbf{B}'| < 0.8B$, we have that at least $0.2B$ of the $\boldsymbol{y}_{L_i}$'s contain both 0 and 1, and thus there are at least $0.1B$ many disjoint 01's appearing in $\boldsymbol{y}_L$. Given that $|\boldsymbol{x}^L| < n/3 + \gamma n$, the first $n/3 + \gamma n$ bits of x contain at least $0.1B$ many disjoint 01's, and so the first $n/3$ bits of x (i.e. $x^{(1)}$) contains at least $0.1B - \gamma n$ many disjoint 01's. Using $c = n/3 - 0.1B$ and Fact 3.1, we have

$$\left|\mathsf{LCS}(x^{(1)}, (01)^c)\right| = \left|\mathsf{LCS}\left(x^{(1)}, (01)^{n/3-0.1B}\right)\right| \geq \left|\mathsf{LCS}\left(x^{(1)}, (01)^{n/3-(0.1B-\gamma n)}\right)\right| - 2\gamma n \geq \frac{n}{3} - 2\gamma n.$$

Next, given that $x^{(2)}$ only has length a and $x^{(3)}$ contains at least $b - \gamma n$ many $\boldsymbol{z}$'s, trivially we have

$$\left|\mathsf{LCS}(x^{(2)}, (01)^a)\right| = a \quad \text{and} \quad \left|\mathsf{LCS}(x^{(3)}, \boldsymbol{z}^b)\right| \geq b - \gamma n.$$

It follows that

$$\left|\mathsf{LCS}(x, \widehat{\boldsymbol{x}})\right| \geq \frac{n}{3} + a + b - 3\gamma n = \frac{2n}{3} + \frac{\rho n}{90000} - 3\gamma n$$

and A succeeds. This finishes the proof of the claim. ∎

Case 2: $|B'| \geq 0.8B$. Recall that we are in Case 2 if $\boldsymbol{y} \sim \mathrm{Del}_\delta(x)$ has length at least $(\rho - \gamma)n$ and $|\mathbf{B}'| \geq 0.8B$. For each $i \in [B]$ we set $\boldsymbol{z}_i$ to be the bit such that $\boldsymbol{y}_{L_i} = \boldsymbol{z}_i^{2000}$ if $i \in \mathbf{B}'$ and set $\boldsymbol{z}_i = 0$ if $i \notin \mathbf{B}'$. We also write $\boldsymbol{z}$ to denote the majority of the last $2\rho n/9$ bits of $\boldsymbol{y}$.

The proof proceeds in a similar fashion as Case 1. Let $x = x^{(1)} \circ x^{(2)} \circ x^{(3)}$ with

$$|x^{(1)}| = n/3, \quad |x^{(2)}| = 2n/9 \quad \text{and} \quad |x^{(3)}| = 4n/9.$$

Our goal is to show that

$$\mathbf{Pr}_{\boldsymbol{y} \sim \mathrm{Del}_\delta(x)}\left[\boldsymbol{y} \text{ in Case 2 and } A \text{ fails}\right] \leq e^{-\Omega(\gamma^2 \rho n)}.$$

Let E_1 denote the event of $|\boldsymbol{x}^L| \geq n/3 + \gamma n$, E_2 denote the event of $\boldsymbol{z}$ appearing less than $2n/9 - \gamma n$ many times in $x^{(3)}$, and E_3 denote the following event:

E_3: For at least $0.02B$ of $i \in \mathbf{B}'$, the subword $\boldsymbol{x}^{L_i}$ contains at most $0.9 \cdot 2000/\rho$ many $\boldsymbol{z}_i$'s.

This follows from the following two claims:

Claim 3.4. *For any string $x \in \{0,1\}^n$, we have*

$$\mathbf{Pr}_{\boldsymbol{y} \sim \mathrm{Del}_\delta(x)}\left[\boldsymbol{y} \textit{ in Case } 1 \wedge (E_1 \vee E_2 \vee E_3)\right] \leq e^{-\Omega(\gamma^2 \rho n)}.$$

Claim 3.5. *The algorithm A succeeds whenever $\boldsymbol{y} \sim \mathrm{Del}_\delta(x)$ satisfies (1) $\boldsymbol{y}$ falls in Case 2; (2) $\overline{E_1}$: $|\boldsymbol{x}^L| < n/3 + \gamma n$; (3) $\overline{E_2}$: $\boldsymbol{z}$ appears at least $2n/9 - \gamma n$ many times in $x^{(3)}$; and (4) $\overline{E_3}$: At most $0.02B$ of $i \in \mathbf{B}'$ has $\boldsymbol{x}^{L_i}$ contain at most $0.9 \cdot 2000/\rho$ many $\boldsymbol{z}_i$'s.*

Proof. [Proof of Claim 3.4] Events E_1 and E_2 can be handled similarly as in the proof of Claim 3.2. Below we show that $\mathbf{Pr}_{\boldsymbol{y}}[E_3] \leq e^{-\Omega(\gamma^2 \rho n)}$. To this end, note that E_3 means there are at least $0.02B$ many $i \in [B]$ such that $\boldsymbol{y}_{L_i}$ is all $\boldsymbol{z}_i$ for some $\boldsymbol{z}_i \in \{0,1\}$ while $\boldsymbol{x}^{L_i}$ has at most $0.9 \cdot 2000/\rho$ many $\boldsymbol{z}_i$.

Let $\mathbf{Z}_i$ be the indicator random variable for the event above for each $i \in [B]$. We show below that conditioning on any outcomes of $\boldsymbol{x}^1, \ldots, \boldsymbol{x}^{2000(i-1)}$, the probability of $\mathbf{Z}_i = 1$ is at most 0.01. It follows that E_2 occurs with probability at most $e^{-\Omega(B)} = e^{-\Omega(\rho n)}$.

For each $i \in [B]$, after fixing any outcomes of $\boldsymbol{x}^1, \ldots, \boldsymbol{x}^{2000(i-1)}$, a necessary condition for $\mathbf{Z}_i$ to be 1 is that among the first $0.9 \cdot 2000/\rho$ many 0's in the current x_∞, at least 2000 of them survive in $\boldsymbol{y}_\infty$, or among the first $0.9 \cdot 2000/\rho$ many 1's in x_∞, at least 2000 of them survive in $\boldsymbol{y}_\infty$. The probability of $\mathbf{Z}_i = 1$ can be bounded from above by 0.01 using the Chernoff bound. ∎

Proof. [Proof of Claim 3.5] When $\boldsymbol{y} \sim \mathrm{Del}_\delta(x)$ falls in Case 2, the algorithm A returns

$$\widehat{\boldsymbol{x}} = \widehat{\boldsymbol{x}}^{(1)} \circ (01)^{2n/9} \circ \boldsymbol{z}^{2n/9},$$

where $\widehat{\boldsymbol{x}}^{(1)}$ is the concatenation of $\boldsymbol{z}_i^{2000/\rho}$, $i \in [B]$. We lowerbound $|\mathsf{LCS}(x, \widehat{\boldsymbol{x}})|$ by

$$\begin{aligned}&\left|\mathsf{LCS}(x^{(1)}, \widehat{\boldsymbol{x}}^{(1)})\right| + \left|\mathsf{LCS}(x^{(2)}, (01)^{2n/9})\right| + \left|\mathsf{LCS}(x^{(3)}, \boldsymbol{z}^{2n/9})\right| \\ &\geq \left|\mathsf{LCS}(x^{(1)}, \widehat{\boldsymbol{x}}^{(1)})\right| + 2n/9 + 2n/9 - \gamma n.\end{aligned}$$

To bound $|\mathsf{LCS}(x^{(1)}, \widehat{\boldsymbol{x}}^{(1)})|$, we write $\mathbf{B}''$ to denote the set of $i \in \mathbf{B}'$ such that $\boldsymbol{x}^{L_i}$ contains at least $0.9 \cdot 2000/\rho$ many $\boldsymbol{z}_i$'s. It follows from Item (4) in Claim 3.5 that $|\mathbf{B}''| \geq 0.98 \cdot 0.8B \geq 0.78B$. We have

$$\begin{aligned}\left|\mathsf{LCS}(x^{(1)}, \widehat{\boldsymbol{x}}^{(1)})\right| &\geq \left|\mathsf{LCS}(\boldsymbol{x}^{L}, \widehat{\boldsymbol{x}}^{(1)})\right| - \gamma n \\ &\geq \sum_{i \in \mathbf{B}''} \left|\mathsf{LCS}(\boldsymbol{x}^{L_i}, \boldsymbol{z}_i^{2000/\rho})\right| - \gamma n \\ &\geq 0.78 \cdot \frac{\rho n}{6000} \cdot 0.9 \cdot \frac{2000}{\rho} - \gamma n \\ &= 0.702 \cdot \frac{n}{3} - \gamma n.\end{aligned}$$

Therefore, altogether we have

$$\left|\mathsf{LCS}(x, \widehat{\boldsymbol{x}})\right| \geq \frac{0.702 \cdot 3 + 4}{9} \cdot n - 2\gamma n \geq (0.678 - 2\gamma)n$$

and A succeeds. This finishes the proof of the claim. ∎

3.2 Bounds on the performance of any one-trace (or few-trace) algorithms. Complementing Theorem 3.1, we show that for worst-case approximate trace reconstruction, even if the total number of bits obtained across multiple traces is $n^{0.999}$, it is not possible to achieve expected LCS of $(2/3 + c)n$ for any constant $c > 0$. The following theorem gives a more detailed version of Theorem 1.2.

THEOREM 3.2. (WORST-CASE UPPER BOUND ON ANY FEW-TRACE ALGORITHM, SMALL RETENTION RATE) *Let $\kappa > 0$ be any absolute constant and let $t(n), \rho(n) = 1 - \delta(n)$ be such that $t(n)\rho(n) \leq 1/n^{\kappa}$. For sufficiently large n, we have*

$$L_{t(n),\mathrm{worst}}(\delta(n), n) \leq (2/3 + o_n(1))n.$$

In order to prove Theorem 3.2, we first introduce some additional notation.

3.2.1 Notation.

Decks. For $k \in \mathbb{N}$, the *k-deck* of a string $z \in \{0,1\}^n$, denoted $\mathsf{D}_k(z)$, is the vector in $\mathbb{Z}^{\{0,1\}^k}$ whose y-th element (for $y \in \{0,1\}^k$) is the number of occurrences of y as a length-k subsequence of z.

Let $\mathcal{M}$ be a mixture of n-bit strings with mixing weights $p_1, \ldots, p_m$ on strings $z^1, \ldots, z^m \in \{0,1\}^n$ (in other words $\mathcal{M}$ is a distribution over n-bit strings). The *k-deck of $\mathcal{M}$*, denoted $\mathsf{D}_k(\mathcal{M})$, is defined to be the following vector in $\mathbb{R}^{\{0,1\}^k}$:

$$\mathsf{D}_k(\mathcal{M}) = \sum_{i=1}^{m} p_i \mathsf{D}_k(z^i).$$

Given $y \in \{0,1\}^k$ we write $\mathsf{D}_k(z)_y$ to denote the y-th element of $\mathsf{D}_k(z)$ and $\mathsf{D}_k(\mathcal{M})_y$ to denote the y-th element of $\mathsf{D}_k(\mathcal{M})$. Note that for any string $z \in \{0,1\}^n$ we have $\sum_{y \in \{0,1\}^k} \mathsf{D}_k(z)_y = \binom{n}{k}$, and likewise $\sum_{y \in \{0,1\}^k} \mathsf{D}_k(\mathcal{M})_y = \binom{n}{k}$ for any mixture $\mathcal{M}$ of n-bit strings.

Segments. We view an n-bit source string $x \in \{0,1\}^n$ as being composed of n/ℓ consecutive segments of length ℓ, for some $\ell = \ell(n)$.

Average LCS of a set. Given any set of strings $\mathcal{S} \subseteq \{0,1\}^n$, define

$$\mathsf{AvgLCS}(\mathcal{S}) := \max_{x' \in \{0,1\}^n} \frac{1}{|\mathcal{S}|} \sum_{s \in \mathcal{S}} |\mathsf{LCS}(x', s)|,$$

i.e., $\mathsf{AvgLCS}(\mathcal{S})$ is the largest possible value (over all possible hypothesis strings $x' \in \{0,1\}^n$) of the average LCS between an element of $\mathcal{S}$ and x'.

We will relate $L_{t(n),\mathrm{worst}}(\delta(n), n)$ to $\mathsf{AvgLCS}(\mathcal{S})$ of a set $\mathcal{S}$ which is (a slight modification of) the *Bukh–Ma code*, a set of n-bit strings that was first studied in [BM14] and further analyzed in [GHS20].

3.2.2 The Bukh–Ma code. Fix a segment length $\ell = \ell(n)$ which divides n. Take ε to be a suitable $o_n(1)$ value, and let $C_{n,\varepsilon}$ be the Bukh–Ma code analyzed in [GHS20]:

$$C_{n,\varepsilon} = \left\{ (0^r 1^r)^{\frac{n}{2r}} : r = \frac{1}{\varepsilon^{4u}}, u = 1, \ldots, \frac{1}{2}\log_{1/\varepsilon^4} \ell \right\}. \tag{3.9}$$

We denote the string $(0^r 1^r)^{\frac{n}{2r}}$ where $r = \frac{1}{\varepsilon^{4u}}$ by A_u, for $u = 1, \ldots, \frac{1}{2}\log_{1/\varepsilon^4} \ell$. We remark that for each string A_u in the Bukh–Ma code above, the "period" $2r = 2/\varepsilon^{4u}$ divides the segment length ℓ.

THEOREM 3.3. (IMPLICIT IN THE PROOF OF [GHS20], THEOREM 1.4) *For any $x \in \{0,1\}^n$, there can be at most $\frac{1200}{\varepsilon^3}$ many strings $A_u \in C_{n,\varepsilon}$ that have $|\mathsf{LCS}(x, A_u)| \geq (2/3 + \varepsilon/6)n$.*

Proof sketch: We explain how Theorem 3.3 is implicit in the proof of Theorem 1.4 of [GHS20]. In [GHS20], it is shown (see Section 3, starting after the proof of their Lemma 3.1) that for any $x \in \{0,1\}^n$, if a set of m strings from $C_{n,\varepsilon}$ is such that each of the m strings (call the string s) has $\mathrm{adv}(x,s) > \varepsilon/2$, then we must have $m \leq 1200/\varepsilon^3$. Since $\mathrm{adv}(x,s) = \frac{3|\mathsf{LCS}(x,s)| - |x| - |s|}{|x|}$ (see [GHS20]'s Definitions 2.4 and 2.5), having $\mathrm{adv}(x,s) > \varepsilon/2$ is equivalent to having $|\mathsf{LCS}(x,s)| \geq (2/3 + \varepsilon/6)n$.

Fix $x \in \{0,1\}^n$. Using Theorem 3.3, we can upper bound the average LCS of x with $C_{n,\varepsilon}$ by

$$\frac{1}{|C_{n,\varepsilon}|} \sum_{s \in C_{n,\varepsilon}} |\mathsf{LCS}(x,s)| \leq \frac{2 \cdot 1200/\varepsilon^3}{\log_{1/\varepsilon^4} \ell} \cdot n + (2/3 + \varepsilon/6)n = (2/3 + o(1))n. \tag{3.10}$$

As this is true for all $x \in \{0,1\}^n$, we conclude that

$$\mathsf{AvgLCS}(C_{n,\varepsilon}) \leq (2/3 + o(1))n. \tag{3.11}$$

3.2.3 Relating $L_{t(n),\mathrm{worst}}(\delta(n), n)$ to $\mathsf{AvgLCS}(\mathcal{S})$. The following claim will allow us to upper bound the performance of any algorithm that receives $t = t(n)$ traces at deletion rate $\delta(n)$ by (essentially) $\mathsf{AvgLCS}(\mathcal{S})$ for any set $\mathcal{S}$ satisfying certain properties.

CLAIM 3.6. *Let ℓ be such that both ℓ and n^ℓ are at least n^c for some positive constant c. Let $\mathcal{S}_\ell = \{s_\ell^{(1)}, s_\ell^{(2)}, \ldots, s_\ell^{(m)}\} \subset \{0,1\}^\ell$ be a set of ℓ-bit strings. Define the set of n-bit strings $\mathcal{S}_n = \{s_n^{(1)}, s_n^{(2)}, \cdots, s_n^{(m)}\} \subset \{0,1\}^n$, where each string $s_n^{(u)}$ is constructed by concatenating n/ℓ copies of $s_\ell^{(u)}$. For each $u \in [m]$ let $\mathcal{M}^{(u)}$ be a mixture of ℓ-bit strings with the following properties:*

1. *With probability $1 - o(1)$, a random ℓ-bit string $\boldsymbol{z}$ drawn from $\mathcal{M}^{(u)}$ has $\mathsf{LCS}(\boldsymbol{z}, s_\ell^{(u)}) \geq (1 - o(1))\ell$;*
2. *For each $u \in [m]$ the k-deck $\mathsf{D}_k(\mathcal{M}^{(u)})$ is the same.*

Let $\rho(n) = 1 - \delta(n)$. Then we have

$$L_{t(n),\mathrm{worst}}(\delta(n), n) \leq t(n) \cdot \ell^k \cdot \rho(n)^{k+1} \cdot n^2 + \mathsf{AvgLCS}(\mathcal{S}_n) + o(n).$$

Proof. Let $\mathcal{M}$ be the following distribution over n-bit strings: to draw $\boldsymbol{x} \sim \mathcal{M}$, first draw a uniform $\boldsymbol{u} \sim [m]$, then independently draw n/ℓ many ℓ-bit strings $\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(n/\ell)} \sim \mathcal{M}^{(\boldsymbol{u})}$, and concatenate them to yield $\boldsymbol{x} = \boldsymbol{x}^{(1)} \cdots \boldsymbol{x}^{(n/\ell)}$.

Let A be any algorithm that takes as input $t := t(n)$ traces $\boldsymbol{y}^{(1)}, \cdots, \boldsymbol{y}^{(t)}$ of $\boldsymbol{x}$, and outputs an n-bit hypothesis string. We suppose that in addition to the input traces, A is also told, for each trace, how many bits of the trace come from each of the n/ℓ segments of the source string; we upper bound $L_{t(n),\mathrm{worst}}(\delta(n), n)$ by upper bounding the performance of any algorithm that also receives this extra auxiliary information.

The probability that any of the n/ℓ many ℓ-bit segments of $\boldsymbol{x}$ has at least $k+1$ bits from it surviving into any of the t traces is at most $t \cdot (n/\ell) \cdot (\rho(n)\ell)^{k+1} = tn\ell^k \rho(n)^{k+1}$. In this case we trivially upper bound the LCS between $\boldsymbol{x}$ and the output of A by n.

Otherwise, at most k bits survive from each segment in each trace. The distribution of these bits is the same, regardless of the random $\boldsymbol{u} \sim [m]$ chosen in the construction of $\boldsymbol{x}$. This follows from property (2.) above and the easily observable fact that if the k-deck $\mathsf{D}_k(\mathcal{M}^{(u)})$ is the same for each $u \sim [m]$, then the k'-deck $\mathsf{D}_{k'}(\mathcal{M}^{(u)})$ is also the same for each $u \sim [m]$, for all $k' \le k$. In this case, the optimal string for algorithm A to output is the n-bit string x^* that achieves $\mathsf{AvgLCS}(\mathcal{S}_n)$.

By property (1.) above and a standard Chernoff bound, with $1 - o(1)$ probability we have that a $1 - o(1)$ fraction of the n/ℓ strings $\boldsymbol{x}^{(1)}, \cdots, \boldsymbol{x}^{(n/\ell)}$ drawn from $\mathcal{M}^{(\boldsymbol{u})}$ satisfy $|\mathsf{LCS}(\boldsymbol{x}^{(i)}, s_\ell^{(\boldsymbol{u})}| \ge (1 - o(1))\ell$, so with $1 - o(1)$ probability the string $\boldsymbol{x} = \boldsymbol{x}^{(1)} \cdots \boldsymbol{x}^{(n/\ell)}$ has $|\mathsf{LCS}(\boldsymbol{x}, s_n^{(\boldsymbol{u})})| \ge (1 - o(1))n$.

Recall that $x^* \in \{0,1\}^n$ is the string achieving $\mathsf{AvgLCS}(\mathcal{S}_n)$. We will use the triangle inequality on the edit distance $d_{edit}(z, z') := n - |\mathsf{LCS}(z, z')|$ (which is a metric). We have

$$d_{edit}(\boldsymbol{x}, x^*) \ge d_{edit}(x^*, s_n^{(\boldsymbol{u})}) - d_{edit}(s_n^{(\boldsymbol{u})}, \boldsymbol{x}).$$

Rewriting this inequality in terms of LCS, we have

$$|\mathsf{LCS}(\boldsymbol{x}, x^*)| \le |\mathsf{LCS}(x^*, s_n^{(\boldsymbol{u})})| + n - |\mathsf{LCS}(\boldsymbol{x}, s_n^{(\boldsymbol{u})})| \le |\mathsf{LCS}(x^*, s_n^{(\boldsymbol{u})})| + o(n).$$

We emphasize that $\boldsymbol{x}$ is a function of the random $\boldsymbol{u} \sim [m]$, while x^* is independent of $\boldsymbol{u}$. Taking expectation over $\boldsymbol{u}$, we get that

$$\mathbf{E}_{\boldsymbol{u}}[|\mathsf{LCS}(\boldsymbol{x}, x^*)|] \le \mathsf{AvgLCS}(\mathcal{S}_n) + o(n).$$

Combining the two cases above, we obtain the lemma. ∎

Proof. [Proof of Theorem 3.2 using Claim 3.6] In Lemma 3.1 below, for any constant $k \in \mathbb{N}$, we will exhibit a set of mixtures $\mathcal{M}^{(u)}$ satisfying the properties in Claim 3.6, with the set $\mathcal{S}_n$ being $C_{n,\varepsilon}$. Choosing $k = 4/\kappa$ (constant), $\ell = n^{1/k}$, and using the fact that $\mathsf{AvgLCS}(C_{n,\varepsilon}) \le (2/3 + o(1))n$ (recall Equation (3.11)), we conclude that

$$\begin{aligned} L_{t(n),\mathrm{worst}}(\delta(n), n) &\le t(n)\, \ell^k\, \rho(n)^{k+1}\, n^2 + \mathsf{AvgLCS}(\mathcal{S}_n) + o(n) \\ &\le (t(n)\rho(n))^{k+1} n^3 + (2/3 + o(1))n \\ &\le n^{3-(k+1)\kappa} + (2/3 + o(1))n \\ &\le (2/3 + o(1))n. \end{aligned}$$

∎

3.3 Construction of $\mathcal{M}$ satisfying Claim 3.6 for any constant k.

Let $\mathcal{S}_\ell$ be the set of $m := \frac{1}{2}\log_{1/\varepsilon^4} \ell$ many ℓ-bit strings

$$\mathcal{S}_\ell = \left\{ (0^{1/\varepsilon^{4u}} 1^{1/\varepsilon^{4u}})^{\ell/(2/\varepsilon^{4u})} \right\}, \quad u = 1, \ldots, m.$$

Fix any positive integer k (which should be thought of as a fixed constant, while $\ell \to \infty$). In this section we construct a collection of m mixtures $\mathcal{M}^{(1)}, \ldots, \mathcal{M}^{(m)}$, where each $\mathcal{M}^{(u)}$ is a mixture of ℓ-bit strings, which meet the conditions required by Claim 3.6. In more detail, we show that the mixtures $\mathcal{M}^{(1)}, \ldots, \mathcal{M}^{(m)}$ that we construct satisfy the following:

LEMMA 3.1. *For each $u \in [m]$ we have the following:*

1. *With probability* $1 - o_\ell(1)$*, a random* ℓ*-bit string* $\boldsymbol{z}$ *drawn from* $\mathcal{M}^{(u)}$ *has*

$$\left|\mathsf{LCS}\left(\boldsymbol{z}, (0^{1/\varepsilon^{4u}}1^{1/\varepsilon^{4u}})^{\ell/(2/\varepsilon^{4u})}\right)\right| \geq (1 - o_\ell(1))\ell.$$

2. *For each* $u \in [m]$ *the* k*-deck* $\mathsf{D}_k(\mathcal{M}^{(u)})$ *is the same.*

The mixture $\mathcal{M}^{(u)}$**.** Fix $u \in [m]$ and let $r_0 := 1/\varepsilon^{4u}$. For t dividing ℓ, let $x^{(t)}$ denote the ℓ-bit string

$$x^{(t)} := (0^t 1^t)^{\ell/(2t)},$$

so $x^{(r_0)}$ is the u-th string $(0^{1/\varepsilon^{4u}}1^{1/\varepsilon^{4u}})^{\ell/(2/\varepsilon^{4u})}$ in $\mathcal{S}_\ell$. The mixture $\mathcal{M}^{(u)}$ will be supported on k strings in $\{0,1\}^\ell$,

$$\mathrm{supp}(\mathcal{M}^{(u)}) = \{x^{(r_0)}, x^{(r_1)}, \ldots, x^{(r_{k-1})}\},$$

where $r_1, \ldots, r_{k-1}$ are values that will satisfy $r_0 \ll r_1 \ll \cdots \ll r_{k-1} \ll \ell$ and that will be specified later. The mixing weight p_j on the j-th string $x^{(r_j)}$ will be chosen so that each $p_j \geq 0$, $\sum_{j=0}^{k-1} p_j = 1$ (so $\mathcal{M}^{(u)}$ is indeed a valid distribution), and $p_0 = 1 - o_\ell(1)$, which gives item (1) of Lemma 3.1.

To achieve item (2) of Lemma 3.1 we will carefully choose the weights $p_0, \ldots, p_{k-1}$ so that for each $y \in \{0,1\}^k$, the value $\mathsf{D}_k(\mathcal{M}^{(u)})_y$ is a function only of ℓ (and in particular is independent of the value of u). Towards this end, let us begin to analyze the k-deck of a single string $x^{(t)}$. The following is easily verified:

Claim 3.7. *Fix any* $y \in \{0,1\}^k$*. The value* $\mathsf{D}_k(x^{(t)})_y$ *is of the form*

$$\mathsf{D}_k(x^{(t)})_y = \sum_{i=0}^{k-1} t^i f_{y,i}(\ell) \tag{3.12}$$

for some polynomials $f_{y,0}(\ell), \ldots, f_{y,k-1}(\ell)$*.*

From Equation (3.12) we immediately get that

$$\mathsf{D}_k(\mathcal{M}^{(i)})_y = \sum_{j=0}^{k-1} p_j \left(\sum_{i=0}^{k-1} r_j^i f_{y,i}(\ell)\right) = \sum_{i=0}^{k-1} \left(\sum_{j=0}^{k-1} p_j r_j^i\right) f_{y,i}(\ell). \tag{3.13}$$

Recall that $r_0 = 1/\varepsilon^{4u}$, so clearly r_0 depends on u, and that we have yet to choose $r_1, \ldots, r_{k-1}$. Equation (3.13) leads us to consider the following linear system:

$$Vp = b \tag{3.14}$$

where V is the $k \times k$ Vandermonde matrix whose rows and columns we index by $i \in \{0, \ldots, k-1\}$ and $j \in \{0, \ldots, k-1\}$,

$$V_{i,j} = r_j^i, \tag{3.15}$$

and p and b are $k \times 1$ column vectors

$$p = \begin{pmatrix} p_0 \\ \vdots \\ p_{k-1} \end{pmatrix}, \qquad b = \begin{pmatrix} b_0 \\ \vdots \\ b_{k-1} \end{pmatrix}.$$

We will prove the following claim:

Claim 3.8. *There are values* $b_0, \ldots, b_{k-1}$ *that have no dependence on* u *so that the solution*

$$p = V^{-1}b \tag{3.16}$$

to the system (3.14) has each $p_j \geq 0$*,* $\sum_{j=0}^{k-1} p_j = 1$*, and* $p_0 = 1 - o_\ell(1)$*.*

By Equation (3.13) this means that the k-deck

$$\mathsf{D}_k(\mathcal{M}^{(i)})_y = \sum_{i=0}^{k-1} b_i f_{y,i}(\ell), \qquad y \in \{0,1\}^k,$$

has no dependence on u, giving item (2) of Lemma 3.1 and completing its proof. It thus remains to prove Claim 3.8.

3.3.1 Proof of Claim 3.8. We start by recalling an explicit formula for the inverse of a Vandermonde matrix:

FACT 3.2. ([TUR66]) *Let $V = (V_{ij})_{i,j\in\{0,\dots,k-1\}}$ be the $k \times k$ Vandermonde matrix $V_{i,j} = r_j^i$ as specified in Equation (3.15). Let $e_j^{(i)}$ be the j-th elementary symmetric polynomial on the $k-1$ variables $r_0, \dots, r_{i-1}, r_{i+1}, \dots, r_{k-1}$. Then the inverse matrix V^{-1} is given by*

$$V_{i,j}^{-1} = \frac{(-1)^j \cdot e_{k-1-j}^{(i)}}{\prod_{s\neq i}(r_s - r_i)}. \tag{3.17}$$

It will be convenient for us to rewrite Equation (3.17) in a way which makes the denominator always positive (recall that we will have $r_0 \ll r_1 \ll \cdots \ll r_{k-1}$. Doing this, we obtain

$$V_{i,j}^{-1} = \frac{(-1)^{i+j} \cdot e_{k-1-j}^{(i)}}{\left(\prod_{0\le s\le i-1}(r_i - r_s)\right) \cdot \left(\prod_{i+1\le s\le k-1}(r_s - r_i)\right)}, \tag{3.18}$$

and consequently we have that

$$p = V^{-1}b, \quad \text{where for } i = 0, \dots, k-1, \quad p_i = \frac{\sum_{j=0}^{k-1}(-1)^{j+i} e_{k-1-j}^{(i)} \cdot b_j}{\left(\prod_{0\le s\le i-1}(r_i - r_s)\right) \cdot \left(\prod_{i+1\le s\le k-1}(r_s - r_i)\right)} \tag{3.19}$$

(note that the denominator of Equation (3.19) is independent of j).

We now choose $r_j, b_j : j \in [k-1]$ appropriately and show that the p_i's satisfy the conditions in Claim 3.8. Recall that $m = \frac{1}{2}\log_{1/\varepsilon^4}\ell$, so $r_0 = 1/\varepsilon^{4u} \le 1/\varepsilon^{4m} = \sqrt{\ell}$. For $j \in [k-1]$, we define

$$b_j := \frac{1}{(\log\log\ell)^j} \cdot \prod_{s=1}^{j} r_j \quad \text{and} \quad r_j := \ell^{2/3} \cdot (\log\ell)^j$$

(observe that r_0 is already fixed to $1/\varepsilon^{4u}$, and that the first row of the Vandermonde matrix system of equations is all-1's, which means that $b_0 = p_0 + \cdots + p_{k-1} = 1$). These settings are chosen so that in the summation in the numerator of the expression for p_i in Equation (3.19), the $(j = i)$-th term, which is always positive, dominates the sum of the rest of the terms in magnitude. Specifically, we will show that for $j < i$, the quantity $e_{k-1-j}^{(i)} b_j$ is at most $O((\log\ell)^{-1}) \cdot \prod_{s=1}^{k-1} r_s$ and for $j \ge i$, we have $e_{k-1-j}^{(i)} b_j = (\log\log\ell)^{-j}(1 + o_\ell(1)) \prod_{s=1}^{k-1} r_s$. So the numerator is at least $(\log\log\ell)^{-i}(1 - o_\ell(1)) \prod_{s=1}^{k-1} r_s \ge 0$, and thus the p_i's are positive because the denominator is positive. Moreover, the denominator of p_0 in Equation (3.19) is at most $\prod_{s=1}^{k-1} r_s$. This shows $p_0 = 1 - o_\ell(1)$.

We now give the full calculation. First observe that for every $0 \le j \le k-2$ and every $S \subseteq [k-1]$ of size $k-1-j$ not equal to $\{j+1, \dots, k-1\}$, we have

$$\prod_{s\in S} r_s \le \ell^{2|S|/3} \cdot (\log\ell)^{\sum_{s\in S} s} \le \ell^{2|S|/3} \cdot (\log\ell)^{(\sum_{s=j+1}^{k-1} j)-1} = \frac{1}{\log\ell} \prod_{s=j+1}^{k-1} r_s. \tag{3.20}$$

So for $j < i$ we have $i \in \{j+1, \dots, k-1\}$ and so the (positive) quantity $e_{k-1-j}^{(i)} \cdot b_j$ is "small," i.e. at most

$O((\log \ell)^{-1}) \cdot \prod_{s=1}^{k-1} r_s$:

$$
\begin{aligned}
e^{(i)}_{k-1-j} \cdot b_j &= \left(\sum_{\substack{S \subseteq \{0,\dots,k-1\} \setminus \{i\} \\ |S| = k-1-j}} \prod_{s \in S} r_s \right) \cdot \left((\log\log \ell)^{-j} \prod_{s=1}^{j} r_j \right) \\
&\le \left(\binom{k-1}{j} \frac{1}{\log \ell} \prod_{s=j+1}^{k-1} r_s \right) \cdot \left((\log\log \ell)^{-j} \prod_{s=1}^{j} r_s \right) \\
&\le (\log\log \ell)^{-j} \cdot \left(\prod_{s=1}^{k-1} r_s \right) \cdot \frac{k^j}{\log \ell} \qquad (3.21)
\end{aligned}
$$

For $j = i$ the (positive) quantity $e^{(i)}_{k-1-j} \cdot b_j$ is "large," i.e. at least $(\log\log \ell)^{-i} \prod_{s=1}^{k-1} r_s$; more precisely, we have

$$
\begin{aligned}
e^{(i)}_{k-1-j} \cdot b_j &= \left(\sum_{\substack{S \subseteq \{0,\dots,k-1\} \setminus \{i\} \\ |S| = k-1-j}} \prod_{s \in S} r_s \right) \cdot \left((\log\log \ell)^{-j} \prod_{s=1}^{j} r_j \right) \\
&\ge \left(\prod_{s=j+1}^{k-1} r_s \right) \cdot \left((\log\log \ell)^{-j} \prod_{s=1}^{j} r_j \right) \\
&= (\log\log \ell)^{-j} \cdot \left(\prod_{s=1}^{k-1} r_s \right). \qquad (3.22)
\end{aligned}
$$

For $j > i$ the (positive) quantity $e^{(i)}_{k-1-j} \cdot b_j$ is again "small," i.e. $(\log\log \ell)^{-j}(1 + o_\ell(1)) \prod_{s=1}^{k-1} r_s$:

$$
\begin{aligned}
e^{(i)}_{k-1-j} \cdot b_j &= \left(\prod_{s=j+1}^{k-1} r_s + \sum_{\substack{S \subseteq \{0,\dots,k-1\} \setminus \{i\} \\ |S| = k-1-j \\ S \neq \{j+1,\dots,k-1\}}} \prod_{s \in S} r_s \right) \cdot \left((\log\log \ell)^{-j} \prod_{s=1}^{j} r_j \right) \\
&\le \left(\left(\prod_{s=j+1}^{k-1} r_s \right) \cdot \left(1 + \binom{k-1}{j} \frac{1}{\log \ell} \right) \right) \cdot \left((\log\log \ell)^{-j} \prod_{s=1}^{j} r_j \right) \\
&= (\log\log \ell)^{-j} \cdot \left(\prod_{s=1}^{k-1} r_s \right) \cdot \left(1 + \frac{k^j}{\log \ell} \right). \qquad (3.23)
\end{aligned}
$$

Therefore for every $i \in \{0, \dots, k-1\}$, the alternating sum is dominated by the contribution from $j = i$: more precisely, we have

$$
\begin{aligned}
\sum_{j=0}^{k-1} (-1)^{j+i} e^{(i)}_{k-1-j} \cdot b_j &\ge \left(\prod_{s=1}^{k-1} r_s \right) \left(\sum_{j=i}^{k-1} (-1)^{i+j} (\log\log \ell)^{-j} - \frac{1}{\log \ell} \sum_{\substack{0 \le j \le k-1 \\ j \neq i}} \left(\frac{k}{\log\log \ell} \right)^j \right) \\
&\ge \left(\prod_{s=1}^{k-1} r_s \right) \left((\log\log \ell)^{-i} \left(1 - \frac{1}{\log\log \ell} \right) - \frac{2}{\log \ell} \right) \\
&\ge 0.
\end{aligned}
$$

Since, as noted earlier, the denominator of Equation (3.19) is positive, this shows that $p_i \ge 0$ for every

Algorithm 1 SMALL-RATE-RECONSTRUCT

1: Set $j = 1$ and $p_y = 1$.
2: While $p_y \leq |\boldsymbol{y}|$ do:
3: With probability $1 - \delta$ set $\widehat{\boldsymbol{x}}_j := \boldsymbol{y}_{p_y}$ and increment p_y;
4: with the remaining δ probability set $\widehat{\boldsymbol{x}}_j$ to a uniform bit from $\{0, 1\}$.
5: Set $j := j + 1$.
6: If $|\widehat{\boldsymbol{x}}| < n$ then append $0^{n-|\widehat{\boldsymbol{x}}|}$ to $\widehat{\boldsymbol{x}}$, and if $|\widehat{\boldsymbol{x}}| > n$ then delete bits $\widehat{\boldsymbol{x}}_{n+1}, \ldots$ from $\widehat{\boldsymbol{x}}$.
7: Output the n-bit string $\widehat{\boldsymbol{x}}$.

$i \in \{0, \ldots, k-1\}$. Moreover, we have

$$\begin{aligned} p_0 &= \frac{\sum_{j=0}^{k-1} (-1)^j e^{(i)}_{k-1-j} \cdot b_j}{\prod_{s=1}^{k-1}(r_s - r_0)} \\ &\geq \frac{\left(\prod_{s=1}^{k-1} r_s\right)\left(1 - \frac{1}{\log\log \ell} - \frac{2}{\log \ell}\right)}{\prod_{s=1}^{k-1} r_s} \\ &\geq 1 - \frac{2}{\log\log \ell}. \end{aligned}$$

This completes the proof of Claim 3.8.

4 Worst-case one-trace reconstruction, medium and small deletion rate

In this section we consider the medium and small deletion rate regime. In particular, throughout this section we suppose that $\delta \leq 1/2$. (Note that if $\delta > 1/2$, then the quantity $1 - \delta + \delta^2/2 - \delta^3/2 + \delta^4/2 - \delta^5/2$ is less than $2/3$, so the performance guarantee given by Theorem 4.1 is weaker than the guarantee given by Theorem 1.1 / Theorem 3.1.)

4.1 An efficient algorithm achieving expected LCS $(1 - \delta + \delta^2/2 - \delta^3/2 + \delta^4/2 - \delta^5/2 - o(1))n$.

As mentioned in the introduction, it is very easy for a one-trace algorithm to achieve expected LCS at least $(1 - \delta)n$: this can be accomplished simply by having the hypothesis string $\widehat{x}$ be any string that contains the input trace $\boldsymbol{y}$ as a subsequence. The expected LCS of such a hypothesis string is clearly at least $\mathbf{E}[|\boldsymbol{y}|]$, which is $(1 - \delta)n$ by linearity of expectation.

The following theorem shows how to improve on this naive bound:

THEOREM 4.1. (WORST-CASE ALGORITHM, SMALL DELETION RATE) *Let $\delta = \delta(n) \leq 1/2$ be the deletion rate. There is an $O(n)$-time (randomized) algorithm* SMALL-RATE-RECONSTRUCT *which is given as input the values n and δ and a single trace $\boldsymbol{y} \sim \mathrm{Del}_\delta(x)$, where $x \in \{0,1\}^n$ is an unknown and arbitrary source string. For any $\gamma \leq 1$,* SMALL-RATE-RECONSTRUCT *outputs a hypothesis string $\widehat{\boldsymbol{x}} \in \{0,1\}^n$ which satisfies*

$$\mathbf{E}\left[|\mathsf{LCS}(\widehat{\boldsymbol{x}}, x)|\right] \geq \left(1 - e^{-\Omega(\gamma^2 n)}\right)\left(1 - \delta + \frac{\delta^2}{2} - \frac{\delta^3}{2} + \frac{\delta^4}{2} - \frac{\delta^5}{2}\right) n - 3\gamma n$$

(so in particular, taking $\omega(1/\sqrt{n}) \leq \gamma \leq o(1)$, we get that the expected value of $|\mathsf{LCS}(\widehat{\boldsymbol{x}}, x)|$ is at least $(1 - \delta + \delta^2/2 - \delta^3/2 + \delta^4/2 - \delta^5/2 - o(1))n$).

Intuition. The algorithm SMALL-RATE-RECONSTRUCT is given as Algorithm 1. To analyze the algorithm it is convenient to consider the string $\widehat{\boldsymbol{x}}'$ which is $\widehat{\boldsymbol{x}}$ immediately before Step 5 is performed (i.e. with no padding with 0s or deletion applied). We will show later that $\widehat{\boldsymbol{x}}'$ is with high probability "very close to $\widehat{\boldsymbol{x}}$", so we can chiefly reason about $\widehat{\boldsymbol{x}}'$ and take care of the minor difference between $\widehat{\boldsymbol{x}}'$ and $\widehat{\boldsymbol{x}}$ at the end of the argument.

We first observe that $\widehat{\boldsymbol{x}}'$ clearly contains $\boldsymbol{y}$ as a subsequence. The main idea of the proof is that a non-negligible fraction of the times that Step 3 is performed, one or more uniform random bits from $\{0,1\}$ will be placed in between consecutive bits $\boldsymbol{y}_{p_y}$ and $\boldsymbol{y}_{p_y+1}$ in creating the hypothesis string $\widehat{\boldsymbol{x}}'$ exactly when one or more bits of x

were deleted in between $\boldsymbol{y}_{p_y}$ and $\boldsymbol{y}_{p_y+1}$ in the creation of the trace $\boldsymbol{y}$. Each time this happens there is a $1/2$ chance that at least one "additional bit" beyond the subsequence $\boldsymbol{y}$ of $\widehat{\boldsymbol{x}}'$ can be incorporated into a matching between x and $\widehat{\boldsymbol{x}}'$. This is the source of the extra $(\delta^2/2 - \delta^3/2 + \delta^4/2 - \delta^5/2)n$ in the LCS bound. Intuitively, the number of additional bits between every $\boldsymbol{y}_{p_y}$ and $\boldsymbol{y}_{p_y+1}$ in each of x and $\widehat{\boldsymbol{x}}'$ is distributed according to $\mathsf{Geometric}(1-\delta)-1$, so there are at least $\min\{\mathsf{Geometric}(1-\delta)-1, \mathsf{Geometric}(1-\delta)-1\} = \mathsf{Geometric}(1-\delta^2)-1$ many additional bits between $\boldsymbol{y}_{p_y}$ and $\boldsymbol{y}_{p_y+1}$ in *both* of x and $\widehat{\boldsymbol{x}}'$, and there is a $1/2$ chance each uniform additional $\{0,1\}$ bit in $\widehat{\boldsymbol{x}}'$ matches an additional bit in x.

We now provide formal details.

Proof. [Proof of Theorem 4.1] Let $x \in \{0,1\}^n$ be the unknown source string. Consider appending infinitely many copies of a special symbol $*$ to x to form an infinite string x_∞. We sample an infinite subsequence $\boldsymbol{y}_\infty$ of x_∞ by the following infinite process: For each $j = 1, 2, \ldots,$ we sample a prefix $\boldsymbol{x}^j$ of x_∞ of length $|\boldsymbol{x}^j| \sim \mathsf{Geometric}(1-\delta)$, then output the last bit of $\boldsymbol{x}^j$ as the j-th bit of $\boldsymbol{y}_\infty$ and delete the prefix $\boldsymbol{x}^j$ from x_∞ before moving on to the next value of j.

Note that the longest prefix of $\boldsymbol{y}_\infty$ that does not contain any $x_i : i > n$ is identically distributed as the trace $\boldsymbol{y} \sim \mathrm{Del}_\delta(x)$. Equivalently, the concatenation of the last bit in each of $\boldsymbol{x}^1, \ldots, \boldsymbol{x}^{|\boldsymbol{y}|}$ is identically distributed as $\boldsymbol{y} \sim \mathrm{Del}_\delta(x)$.

Let $\mathbf{T} = \{t_1 < \cdots < t_{|\boldsymbol{y}|}\}$ be the set of $|\boldsymbol{y}|$ many locations $j \in \{1, 2, \ldots, \}$ such that $\widehat{\boldsymbol{x}}'_j$ was set to $\boldsymbol{y}_{p_y}$ in some execution of Step 3 of Small-rate-reconstruct. Note that the elements of $\mathbf{T}$ are the indices of the $\mathbf{T}$race bits in $\widehat{\boldsymbol{x}}'$ and that t_i is the index such that $\widehat{\boldsymbol{x}}'_{t_i}$ was set to $\boldsymbol{y}_i$ in Step 3 of the execution of Small-rate-reconstruct. Let

$$\widehat{\boldsymbol{x}}'^1 := \widehat{\boldsymbol{x}}'_{[1:t_1]}, \widehat{\boldsymbol{x}}'^2 := \widehat{\boldsymbol{x}}'_{[t_1+1:t_2]}, \ldots, \widehat{\boldsymbol{x}}'^{|\boldsymbol{y}|} := \widehat{\boldsymbol{x}}'_{[t_{|\boldsymbol{y}|-1}+1:t_{|\boldsymbol{y}|}]}.$$

Observe that for each $i \in [|\boldsymbol{y}|]$, both $\boldsymbol{x}^i$ and $\widehat{\boldsymbol{x}}'^i$ are identically distributed. In particular, their lengths $|\boldsymbol{x}^i|$ and $|\widehat{\boldsymbol{x}}'^i|$ are distributed according to $\mathsf{Geometric}(1-\delta)$, and so the minimum of both lengths, i.e. $\boldsymbol{d}_i := \min\{|\boldsymbol{x}^i|, |\widehat{\boldsymbol{x}}'^i|\}$, is distributed according to $\mathsf{Geometric}(1-\delta^2)$. Moreover, the last bits in $\boldsymbol{x}^i$ and $\widehat{\boldsymbol{x}}'^i$ are equal to $\boldsymbol{y}_i$, and the rest of the bits in $\widehat{\boldsymbol{x}}^i$ are uniform.

For each $i \in [|\boldsymbol{y}|]$, since the length-$(\boldsymbol{d}_i - 1)$ prefix of $\widehat{\boldsymbol{x}}'^i$ is random, in expectation (over the randomness of $\widehat{\boldsymbol{x}}'^i$) it agrees with the length-$(\boldsymbol{d}_i - 1)$ prefix of $\boldsymbol{x}^i$ on $(\boldsymbol{d}_i - 1)/2$ of the bits. Also, the last bit of both $\boldsymbol{x}^i$ and $\widehat{\boldsymbol{x}}'^i$ are the same. Hence, we have

$$\mathbf{E}_{\widehat{\boldsymbol{x}}'^i}\Big[|\mathsf{LCS}(\boldsymbol{x}^i, \widehat{\boldsymbol{x}}^i)|\Big] \geq (\boldsymbol{d}_i - 1)/2 + 1.$$

Observe that the concatenation of $\boldsymbol{x}^i : i \in [|\boldsymbol{y}|]$ is a prefix of x, because the last bit of $\boldsymbol{x}^{|\boldsymbol{y}|}$ is the last bit of $\boldsymbol{y} \sim \mathrm{Del}_\delta(x)$, and the concatenation of $\widehat{\boldsymbol{x}}'^i : i \in [|\boldsymbol{y}|]$ is exactly $\widehat{\boldsymbol{x}}'$. Thus,

$$\mathbf{E}_{\widehat{\boldsymbol{x}}'}\Big[|\mathsf{LCS}(x, \widehat{\boldsymbol{x}}')|\Big] \geq \sum_{i=1}^{|\boldsymbol{y}|}\Big[\mathbf{E}_{\boldsymbol{x}^i, \widehat{\boldsymbol{x}}'^i}|\mathsf{LCS}(\boldsymbol{x}^i, \widehat{\boldsymbol{x}}'^i)|\Big] \geq \sum_{i=1}^{|\boldsymbol{y}|}\Big((\boldsymbol{d}_i - 1)/2 + 1\Big) = |\boldsymbol{y}| + \sum_{i=1}^{|\boldsymbol{y}|}(\boldsymbol{d}_i - 1)/2.$$

Since $\boldsymbol{d}_i \sim \mathsf{Geometric}(1-\delta^2)$, we have $\mathbf{E}[\boldsymbol{d}_i - 1] = \frac{1}{1-\delta^2} - 1 = \frac{\delta^2}{1-\delta^2}$. So taking expectation over $|\boldsymbol{y}|$, we obtain

$$\begin{aligned}
\mathbf{E}\Big[|\mathsf{LCS}(x, \widehat{\boldsymbol{x}}')|\Big] &\geq \mathbf{E}\big[|\boldsymbol{y}|\big] + \mathbf{E}\big[|\boldsymbol{y}|\big]\frac{\delta^2}{2(1-\delta^2)} \\
&= (1-\delta)n \cdot \left(1 + \frac{\delta^2}{2(1-\delta^2)}\right) \\
&= \left(1 - \delta + \frac{\delta^2}{2(1-\delta^2)} - \frac{\delta^3}{2(1-\delta^2)}\right) n \\
&\geq \left(1 - \delta + \frac{\delta^2}{2} - \frac{\delta^3}{2} + \frac{\delta^4}{2} - \frac{\delta^5}{2}\right) n.
\end{aligned}$$

To finish the proof, we relate $\mathbf{E}[|\mathsf{LCS}(\widehat{\boldsymbol{x}}, x)|]$ to $\mathbf{E}[|\mathsf{LCS}(\widehat{\boldsymbol{x}}', x)|]$. We observe that

$$|\mathsf{LCS}(\widehat{\boldsymbol{x}}, x)| \geq \max\{0, |\mathsf{LCS}(\widehat{\boldsymbol{x}}', x)| - \boldsymbol{k}\},$$

where $\boldsymbol{k}$ is the number of bits $\widehat{\boldsymbol{x}}_{n+1}, \ldots$ deleted from $\widehat{\boldsymbol{x}}$ in Step 5 of SMALL-RATE-RECONSTRUCT if bits are deleted in that step (and $\boldsymbol{k} = 0$ otherwise). So it remains only to show that with high probability $\boldsymbol{k}$ is small.

Recall that the value of $|\boldsymbol{y}|$ is distributed as $\mathrm{Bin}(n, 1-\delta)$, and given a particular outcome of the value of $|\boldsymbol{y}|$, the number of bits deleted in Step 5 is distributed as $\min\{0, \mathbf{G}_1 + \cdots + \mathbf{G}_{|\boldsymbol{y}|} - n\}$ where the $\mathbf{G}_i$'s are independent geometric random variables with each $\mathbf{G}_i \sim \mathsf{Geometric}(\rho)$ (recall that $\rho = 1-\delta$). We will use two tail bounds: first, by a multiplicative Chernoff bound, we have

CLAIM 4.1. *For $\gamma \le 1$, we have $\mathbf{Pr}[|\boldsymbol{y}| \ge (1+\gamma)(1-\delta)n] \le e^{-\Omega(\gamma^2 n)}$.*

The second tail bound shows that $\mathbf{G}_1 + \cdots + \mathbf{G}_{|\boldsymbol{y}|}$ is unlikely to be much larger than n:

CLAIM 4.2. *Fix an outcome of $|\boldsymbol{y}|$ such that $|\boldsymbol{y}| \le (1+\gamma)(1-\delta)n$, where $\gamma \le 1$. Then $\mathbf{Pr}[\mathbf{G}_1 + \cdots + \mathbf{G}_{|\boldsymbol{y}|} \ge (1+3\gamma)n] \le e^{-\Omega(\gamma^2 n)}$.*

Proof. Recall that Claim 2.1 upper bounds the probability that $\mathbf{G}_1 + \cdots + \mathbf{G}_{(1+\gamma)(1-\delta)n} \ge \frac{1+\gamma}{1-\delta} \cdot (1+\gamma)(1-\delta)n = (1+\gamma)^2 n$. As $\gamma \le 1$, we get that

$$\mathbf{Pr}[\mathbf{G}_1 + \cdots + \mathbf{G}_{|\boldsymbol{y}|} \ge (1+3\gamma)n] \le e^{-\Omega(\gamma^2 |\boldsymbol{y}|)} = e^{-\Omega(\gamma^2 n)},$$

where the final inequality is by Claim 2.1. ∎

Combining Claim 4.1 and Claim 4.2, we get that $\boldsymbol{k} \le 3\gamma n$ except with probability $e^{-\Omega(\gamma^2 n)}$. Consequently, we have that

$$\begin{aligned}
\mathbf{E}[|\mathsf{LCS}(\widehat{\boldsymbol{x}}, x)|] &\ge \left(1 - e^{-\Omega(\gamma^2 n)}\right)\left(\mathbf{E}[|\mathsf{LCS}(\widehat{\boldsymbol{x}}', x)|] - 3\gamma n\right) \\
&\ge \left(1 - e^{-\Omega(\gamma^2 n)}\right)\left(1 - \delta + \frac{\delta^2}{2} - \frac{\delta^3}{2} + \frac{\delta^4}{2} - \frac{\delta^5}{2}\right) n - 3\gamma n,
\end{aligned}$$

and the theorem is proved. ∎

4.2 Bounds on the performance of one-trace algorithms in the low deletion rate regime. As noted in the Introduction, it is natural to try to complement Theorem 4.1 by proving an upper bound on the best expected LCS that can be achieved by any one-trace algorithm in the low deletion rate regime. An average-case bound is of course stronger than a worst-case bound of this sort; in Section 1.1.2 we will show that even in the average-case setting, the best achievable LCS given a single trace is at most $(1 - \Theta(\delta))n$.

5 Average-case one-trace reconstruction, high and medium deletion rate

In this section we bound the performance of any average-case few-trace algorithm when the retention rate is low. Given the length of the source string n, we write $L_{0,\mathrm{avg}}(n)$ to denote the performance of an optimal zero trace algorithm:

$$L_{0,\mathrm{avg}}(n) = \max_{z \in \{0,1\}^n} \mathop{\mathbf{E}}_{\boldsymbol{x} \sim \{0,1\}^n} \left[|\mathsf{LCS}(\boldsymbol{x}, z)|\right]$$

(note that this quantity does not depend on δ), and recall from Section 2.2 that for $t > 0$, $L_{t,\mathrm{avg}}(\delta, n)$ captures the information-theoretic optimal performance of any t-trace algorithm at deletion rate δ.

We show that when $t\rho$ is small, where t is the number of traces and ρ is the retention rate, it is not possible to do much better than $L_{0,\mathrm{avg}}(n)$:

THEOREM 5.1. (AVERAGE-CASE UPPER BOUND ON ANY ALGORITHM, SMALL RETENTION RATE) *Let n be the length of the source string, t be the number of traces and $\rho = 1 - \delta$ be the retention rate. Then*

$$L_{0,\mathrm{avg}}(n) \le L_{t,\mathrm{avg}}(\delta, n) \le L_{0,\mathrm{avg}}(n) + t\rho \cdot n.$$

We note that as a special case of the theorem above, if $t(n)\rho(n) = o(n)$ then the leading constant of what can be achieved with t traces is no better than if no traces were given. This is in contrast with the worst-case setting, as witnessed by Theorem 1.1 and the discussion immediately preceding it.

The lower bound is immediate so in the rest of this section we prove the upper bound. We start with some notation for working with multiple traces in the proof of Theorem 5.1. Given n and t, let $R_1, \dots, R_t \subseteq [n]$ be t subsets which should be viewed as locations retained from an n-bit string to obtain its t traces (so if the source string is x then the traces are $y^{(s)} = x_{R_s}$ for $s = 1, \dots, t$). Let $i_1 < \cdots < i_m$ be an enumeration of indices in $R_1 \cup \cdots \cup R_t$. We write $C = (C_1, \dots, C_m)$ to denote the tuple where C_j is the set of those $s \in [t]$ such that $i_j \in R_s$. We will refer to C as the *collision information* of $R_1, \dots, R_t$, denoted by $C = C(R_1, \dots, R_t)$.

EXAMPLE 5.1. *Consider the case that $n = 8$, $t = 3$, the source string x is* 11010011*, and the three traces $y^{(1)} =$* 1100*, $y^{(2)} =$* 110*, $y^{(3)} =$* 1001 *are obtained from x as shown below:*

$$\begin{array}{rcccccccc} x: & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 1 \\ y^{(1)}: & 1 & 1 & & & 0 & 0 & & \\ y^{(2)}: & & 1 & & 1 & 0 & & & \\ y^{(3)}: & & 1 & & & 0 & 0 & 1 & \end{array}$$

In this case we have that $m = 6$, $i_1 = 1$, $i_2 = 2$, $i_3 = 4$, $i_4 = 5$, $i_5 = 6$, $i_6 = 7$, and $C_1 = \{1\}$, $C_2 = \{1, 2, 3\}$, $C_3 = \{2\}$, $C_4 = \{1, 2, 3\}$, $C_5 = \{1, 3\}$, $C_6 = \{3\}$. As discussed in Observation 5.1 below, given the traces $y^{(1)}, y^{(2)}, y^{(3)}$ and the collision information C, it is possible to reconstruct an m-bit subsequence y (in this example $y =$ 111001*) of x, but not the $n - m$ bits of x that are missing from y nor the locations of where the m bits of y are situated in x.*

We will consider average-case algorithms that are given not only t traces $\boldsymbol{y}_1, \dots, \boldsymbol{y}_t \sim \mathrm{Del}_\delta(\boldsymbol{x})$ of a random string $\boldsymbol{x} \sim \{0,1\}^n$ but also the collision information $\mathbf{C} = C(\mathbf{R}_1, \dots, \mathbf{R}_t)$, where $\mathbf{R}_s \subseteq [n]$ is the set of locations that are retained in obtaining trace $\boldsymbol{y}^{(s)}$ from $\boldsymbol{x}$ for each $s \in [t]$. Let $L^*_{t,\mathrm{avg}}(\delta, n)$ denote the performance of the best algorithm A under this setting:

$$L^*_{t,\mathrm{avg}}(\delta, n) := \max_A \mathop{\mathbf{E}}_{\boldsymbol{x} \sim \{0,1\}^n} \mathop{\mathbf{E}}_{\mathbf{R}_1, \dots, \mathbf{R}_t \sim \mathcal{R}_\rho} \Big[\big| \mathsf{LCS}\big(A(\boldsymbol{x}_{\mathbf{R}_1}, \dots, \boldsymbol{x}_{\mathbf{R}_t}, \mathbf{C}), \boldsymbol{x}\big) \big| \Big],$$

where we write $\mathcal{R}_\rho$ to denote the distribution where $\mathbf{R} \sim \mathcal{R}_\rho$ is drawn by including each element in $[n]$ independently with probability ρ and $\mathbf{C} = C(\mathbf{R}_1, \dots, \mathbf{R}_t)$ is the collision information of sets $\mathbf{R}_1, \dots, \mathbf{R}_t$. It is clear that $L^*_{t,\mathrm{avg}}(\delta, n) \geq L_{t,\mathrm{avg}}(\delta, n)$. We prove Theorem 5.1 by showing that

$$L^*_{t,\mathrm{avg}}(\delta, n) \leq L_{0,\mathrm{avg}}(n) + t\rho \cdot n.$$

OBSERVATION 5.1. (*A posteriori* DISTRIBUTION OF A UNIFORM RANDOM SOURCE STRING GIVEN t TRACES AND THEIR COLLI
Let $\boldsymbol{x}$ be a uniform random source string drawn from $\{0,1\}^n$. Let $I = (y^{(1)}, \dots, y^{(t)}, C)$ be any fixed outcome of t traces from $\mathrm{Del}_\delta(\boldsymbol{x})$ together with the collision information of their locations retained. Then the a posteriori *distribution of $\boldsymbol{x}$ given I is as follows:*

1. *Let $C = (C_1, \dots, C_m)$ for some $m \leq n$. We define an m-bit string y as follows. For each $j \in [m]$, pick an $s \in C_j$ and set $z_j = y^{(s)}_k$ where k is the number of $j' \leq j$ such that $s \in C_{j'}$. (Note that the value of z_j does not depend on the choice of $s \in C_j$.)*

2. *The rest of the process is the same as the description of the a posteriori distribution of $\boldsymbol{x}$ given one trace y (see Observation 2.1); for convenience we will write $\mathcal{D}_y$ to denote the distribution of $\boldsymbol{x}$ described below. Draw a uniform random m-element subset of $[n]$ (say $\mathbf{S} = \{\boldsymbol{s}_1, \dots, \boldsymbol{s}_m\}$ where $1 \leq \boldsymbol{s}_1 < \cdots < \boldsymbol{s}_m \leq n$);*

3. *For each $j \in [m]$ set $\boldsymbol{x}_{\boldsymbol{s}_j} = z_j$, and for each $i \notin \mathbf{S}$ set $\boldsymbol{x}_i$ to an independent uniform bit.*

We are now ready to prove Theorem 5.1.

Proof. [Proof of Theorem 5.1] Let A be an optimal algorithm that achieves $L^*_{t,\mathrm{avg}}(\delta, n)$. Let $\boldsymbol{x} \sim \{0,1\}^n$ and let $\mathbf{I} = (\boldsymbol{y}^{(1)}, \ldots, \boldsymbol{y}^{(t)}, \mathbf{C})$ be the input of A, where A outputs $A(\mathbf{I}) \in \{0,1\}^n$. Given an outcome I of $\mathbf{I}$, we write $y(I)$ to denote the string derived from $\mathbf{I}$ as in Step 1 of Observation 5.1. Then

$$L^*_{t,\mathrm{avg}}(\delta, n) = \sum_I \mathbf{Pr}\left[\mathbf{I} = I\right] \cdot \mathop{\mathbf{E}}_{\boldsymbol{x} \sim \mathcal{D}_{y(I)}} \Big[\big|\mathsf{LCS}\big(A(I), \boldsymbol{x}\big)\big|\Big], \tag{5.24}$$

where the sum is over all possible inputs I of A. We need the following claim:

Claim 5.1. *Fix any string $y \in \{0,1\}^m$ for some $m \le n$. For any string $z \in \{0,1\}^n$, we have*

$$\mathop{\mathbf{E}}_{\boldsymbol{x} \sim \mathcal{D}_y} \Big[\big|\mathsf{LCS}\big(z, \boldsymbol{x}\big)\big|\Big] \le L_{0,\mathrm{avg}}(n) + m.$$

Proof. Consider the following coupling $(\boldsymbol{x}, \boldsymbol{x}') \sim \mathcal{E}$ of the uniform distribution over $\{0,1\}^n$ and $\mathcal{D}_y$: first draw $\boldsymbol{x} \sim \{0,1\}^n$; then draw a size-$m$ subset $\mathbf{S}$ of $[n]$ uniformly at random and replace bits of $\boldsymbol{x}$ at $\mathbf{S}$ by y to obtain $\boldsymbol{x}'$. It is easy to verify that $\mathcal{E}$ is a coupling of the uniform distribution over $\{0,1\}^n$ and $\mathcal{D}_y$. For any string $z \in \{0,1\}^n$, we have

$$\begin{aligned}\mathop{\mathbf{E}}_{\boldsymbol{x}' \sim \mathcal{D}_y} \Big[\big|\mathsf{LCS}\big(z, \boldsymbol{x}'\big)\big|\Big] &= \mathop{\mathbf{E}}_{(\boldsymbol{x}, \boldsymbol{x}') \sim \mathcal{E}} \Big[\big|\mathsf{LCS}\big(z, \boldsymbol{x}'\big)\big|\Big] \\ &\le \mathop{\mathbf{E}}_{(\boldsymbol{x}, \boldsymbol{x}') \sim \mathcal{E}} \Big[\big|\mathsf{LCS}\big(z, \boldsymbol{x}\big)\big|\Big] + m = \mathop{\mathbf{E}}_{\boldsymbol{x} \sim \{0,1\}^n} \Big[\big|\mathsf{LCS}\big(z, \boldsymbol{x}\big)\big|\Big] + m \le L_{0,\mathrm{avg}}(n) + m,\end{aligned}$$

where the inequality used the fact that $(\boldsymbol{x}, \boldsymbol{x}') \sim \mathcal{E}$ always have Hamming distance at most m. ∎

Combining (5.24) with Claim 5.1, we have

$$L^*_{t,\mathrm{avg}}(\delta, n) \le \sum_I \mathbf{Pr}\left[\mathbf{I} = I\right] \cdot \Big(L_{0,\mathrm{avg}}(n) + |y|\Big) = L_{0,\mathrm{avg}}(n) + \mathbf{E}\big[|\boldsymbol{y}|\big].$$

By linearity of expectation, we have

$$\mathbf{E}\big[|\boldsymbol{y}|\big] = n(1-\delta^t) = n(1-(1-\rho)^t) \le \rho t \cdot n.$$

This finishes the proof of the theorem. ∎

6 Average-case one-trace reconstruction, small deletion rate

6.1 An efficient algorithm improving on the Theorem 4.1 bound. In this section we show that the algorithm Small-rate-reconstruct of Theorem 4.1, that was shown to achieve LCS $(1 - \delta + \delta^2/2 - \delta^3/2 + \delta^4/2 - \delta^5/2 - o(1))n$ for worst-case source strings, in fact does better than this for average-case source strings. The high level idea is that when there are j additional bits between two trace bits in $\boldsymbol{x}$ and k additional bits between two trace bits in $\widehat{\boldsymbol{x}}'$, rather than matching only $\min\{j,k\}/2$ using the randomness of $\widehat{\boldsymbol{x}}'$, we take advantage of the facts that (i) both the $\boldsymbol{x}$-bits and the $\widehat{\boldsymbol{x}}$-bits are uniform random, and (ii) if j or k is greater than 1, then the expected LCS between a random j-bit string and a random k-bit string is strictly larger than $\min\{j,k\}/2$, to obtain (on average) a better matching between these two blocks and hence a larger overall matching. This intuition motivates the following definition:

Definition 6.1. *For integers $j, k > 0$, we define $\mathsf{CS}(j,k)$ as*

$$\mathsf{CS}(j,k) := \mathop{\mathbf{E}}_{\boldsymbol{x} \sim \{0,1\}^j, \boldsymbol{x}' \sim \{0,1\}^k} \big[|\mathsf{LCS}(\boldsymbol{x}, \boldsymbol{x}')|\big].$$

Note that by definition, $\mathsf{CS}(j,k) = \mathsf{CS}(k,j)$, i.e., the function $\mathsf{CS}(\cdot,\cdot)$ is symmetric in its arguments.

While it is not clear if there is a simple explicit formula for $\mathsf{CS}(j,k)$, we note that a brute force algorithm can be used to compute this function. Further, for the special case of $j = k$, the function $\mathsf{CS}(\cdot,\cdot)$ has been studied previously in the literature [CS75]. In particular, for any $d > 0$, $\mathsf{CS}(d,d)$ is the same as the function $f(d,2)$ defined in [CS75, Section 2]. Further, once $d \to \infty$, $\mathsf{CS}(d,d)/d$ is the same as the so-called Chvatal–Sankoff constant for the binary alphabet [KLM05, CS75]. Table 1 gives the values of $\mathsf{CS}(j,k)$ for all $j + k \le 6$.

$k=5$	31/32				
$k=4$	15/16	53/32			
$k=3$	7/8	23/16	29/16		
$k=2$	3/4	9/8	23/16	53/32	
$k=1$	1/2	3/4	7/8	15/16	31/32
value of $\mathsf{CS}(j,k)$	$j=1$	$j=2$	$j=3$	$j=4$	$j=5$

Table 1: Table for values of $\mathsf{CS}(j,k)$ for $j+k \le 6$.

THEOREM 6.1. *Let $\delta = \delta(n)$ be the deletion rate. The $O(n)$-time algorithm* SMALL-RATE-RECONSTRUCT *given in Theorem 4.1 has the following property: for any $\gamma > 0$ and sufficiently large n, algorithm* SMALL-RATE-RECONSTRUCT *outputs a hypothesis string $\widehat{\boldsymbol{x}} \in \{0,1\}^n$ satisfying*

$$\begin{aligned}&\mathop{\mathbf{E}}_{\boldsymbol{x}\in\{0,1\}^n}\mathop{\mathbf{E}}_{\boldsymbol{y}\sim\mathrm{Del}_\delta(\boldsymbol{x})}\left[|\mathsf{LCS}(\widehat{\boldsymbol{x}},\boldsymbol{x})|\right]\\ &\ge \left(1-e^{-\Omega(\gamma^2 n)}\right)(1-\delta)\cdot\left(1+(1-\delta)^2\sum_{j,k=1}^{\infty}\mathsf{CS}(j,k)\cdot\delta^{j+k}\right)n-3\gamma n.\end{aligned}$$

As an example, instantiating with the values of $\mathsf{CS}(j,k)$ from Table 1, the above theorem gives us that

$$L_{1,\mathrm{avg}}(\delta,n) \ge \left(1-\delta+\frac{1}{2}\delta^2+\frac{17}{8}\delta^4+\frac{55}{8}\delta^5+o(\delta^5)\right)n,$$

which improves on the $(1-\delta+\delta^2/2-\delta^3/2+\delta^4/2-\delta^5/2-o(1))n$ bound of Theorem 4.1.

Proof. The proof is analogous to the one in Theorem 4.1. We first replace x in the proof of Theorem 4.1 with a uniform random $\boldsymbol{x}\sim\{0,1\}^n$ in the proof.

Now, for each $i\in[|\boldsymbol{y}|]$, let us define $\boldsymbol{d}_i$ to be $|\boldsymbol{x}^i|$ and $\boldsymbol{d}'_i$ to be $|\widehat{\boldsymbol{x}}'^i|$. Since the length-$(\boldsymbol{d}_i-1)$ prefix of both $\boldsymbol{x}^i$ and the length-$(\boldsymbol{d}'_i-1)$ prefix of $\widehat{\boldsymbol{x}}'^i$ are independent random strings, and the last bit of both $\boldsymbol{x}^i$ and $\widehat{\boldsymbol{x}}'^i$ are the same, we have

$$\mathbf{E}_{\boldsymbol{x}^i,\widehat{\boldsymbol{x}}'^i}\left[|\mathsf{LCS}(\boldsymbol{x}^i,\widehat{\boldsymbol{x}}^i)|\right] \ge \mathsf{CS}(\boldsymbol{d}_i-1,\boldsymbol{d}'_i-1)+1.$$

As $\boldsymbol{d}_i\sim\mathsf{Geometric}(1-\delta)$ and $\boldsymbol{d}'_i\sim\mathsf{Geometric}(1-\delta)$, we have

$$\begin{aligned}\mathbf{E}\big[\mathsf{CS}(\boldsymbol{d}_i-1,\boldsymbol{d}'_i-1)\big] &= \sum_{j,k=1}^{\infty}\Big(\mathsf{CS}(j,k)\cdot\mathbf{Pr}[\boldsymbol{d}_i=j+1 \text{ and } \boldsymbol{d}'_i=k+1]\Big)\\ &= \sum_{j,k=1}^{\infty}\Big(\mathsf{CS}(j,k)\cdot\mathbf{Pr}[\mathsf{Geometric}(1-\delta)=j+1]\cdot\mathbf{Pr}[\mathsf{Geometric}(1-\delta)=k+1]\Big)\\ &= (1-\delta)^2\sum_{j,k=1}^{\infty}\mathsf{CS}(j,k)\cdot\delta^{j+k}.\end{aligned}$$

So we have

$$\begin{aligned}\mathbf{E}\Big[|\mathsf{LCS}(\boldsymbol{x},\widehat{\boldsymbol{x}}')|\Big] &\ge \sum_{i=1}^{|\boldsymbol{y}|}\mathbf{E}_{\boldsymbol{x}^i,\widehat{\boldsymbol{x}}'^i}\left[|\mathsf{LCS}(\boldsymbol{x}^i,\widehat{\boldsymbol{x}}'^i)|\right]\\ &\ge \mathbf{E}\big[|\boldsymbol{y}|\big]+\mathbf{E}\big[|\boldsymbol{y}|\big](1-\delta)^2\sum_{j,k=1}^{\infty}\mathsf{CS}(j,k)\cdot\delta^{j+k}\\ &\ge (1-\delta)n\cdot\left(1+(1-\delta)^2\sum_{j,k=1}^{\infty}\mathsf{CS}(j,k)\cdot\delta^{j+k}\right).\end{aligned}$$

We can again relate $\mathbf{E}[|\mathsf{LCS}(\widehat{\boldsymbol{x}}, \boldsymbol{x})|]$ to $\mathbf{E}[|\mathsf{LCS}(\widehat{\boldsymbol{x}}', \boldsymbol{x})|]$ using the same argument in Theorem 4.1, from which we conclude that

$$\begin{aligned}\mathbf{E}\Big[|\mathsf{LCS}(\widehat{\boldsymbol{x}}, \boldsymbol{x})|\Big] &\geq \Big(1 - e^{-\Omega(\gamma^2 n)}\Big)\Big(\mathbf{E}\Big[|\mathsf{LCS}(\widehat{\boldsymbol{x}}, \boldsymbol{x}')|\Big] - 3\gamma n\Big) \\ &\geq \Big(1 - e^{-\Omega(\gamma^2 n)}\Big)(1-\delta)n \cdot \left(1 + (1-\delta)^2 \sum_{j,k=1}^{\infty} \mathsf{CS}(j,k) \cdot \delta^{j+k}\right) - 3\gamma n,\end{aligned}$$

proving the theorem. ∎

6.2 Bounds on the performance of any one-trace algorithm. Finally, in this section we establish an upper bound on the best possible performance that any one-trace algorithm can achieve in the small-deletion-rate regime. We consider the average-case setting (which is of course more challenging for upper bounds, and yields worst-case upper bounds as an immediate consequence).

A relatively simple analysis shows that $L_{1,\mathrm{avg}}(\delta, n) \leq (1 - c\delta/\log(1/\delta))n$, where c is a universal positive constant. This argument applies a union bound across all possible matchings of a given size, and is given as Theorem B.1 in Section B. It is natural to suspect that this bound is weaker than it should be by a $\Theta(\log(1/\delta))$ factor, but establishing this turns out to be nontrivial. The following theorem establishes a bound of $L_{1,\mathrm{avg}}(\delta, n) \leq (1 - c\delta)n$, which, up to the value of the universal constant c, is best possible by Theorem 6.1 (or even by the trivial algorithm which simply outputs any n-bit string that contains the input trace $\boldsymbol{y}$, and thereby achieves an LCS of expected length at least $\mathbf{E}[|\boldsymbol{y}|] = (1-\delta)n$).

THEOREM 6.2. (AVERAGE-CASE UPPER BOUND ON ANY ALGORITHM, SMALL RETENTION RATE) *There is an absolute constant $c > 0$ such that for any deletion rate $\delta = \delta(n) = \omega(1/n)$ and sufficiently large n, we have $L_{1,\mathrm{avg}}(\delta, n) \leq (1 - c\delta)n$.*

6.2.1 Outline of the argument. Recall that under the average-case setting, the source string $\boldsymbol{x}$ is uniform random over $\{0,1\}^n$, and our goal is to upperbound the performance of any algorithm which is given as input a single trace $\boldsymbol{y} \sim \mathrm{Del}_\delta(\boldsymbol{x})$. Given any trace string y, the optimal algorithm A will return a string $z \in \{0,1\}^n$ to maximize $\mathbf{E}_{\boldsymbol{x} \sim y}[|\mathsf{LCS}(z, \boldsymbol{x})|]$ (recall Observation 2.1 for the a posteriori distribution $\boldsymbol{x} \sim y$). Let us write $\mathrm{opt}(y)$ for an optimal string $z \in \{0,1\}^n$ for the expectation. Given that $\boldsymbol{y} \sim \mathrm{Del}_\delta(\boldsymbol{x})$ and $\boldsymbol{x} \sim \{0,1\}^n$, our goal is to bound

$$L_{1,\mathrm{avg}}(\delta, n) = \mathbf{E}_{\boldsymbol{y}}\left[\mathbf{E}_{\boldsymbol{x} \sim \boldsymbol{y}}\left[|\mathsf{LCS}(\mathrm{opt}(\boldsymbol{y}), \boldsymbol{x})|\right]\right],$$

where $\boldsymbol{y}$ is a uniform random bitstring of length $\boldsymbol{k}$ where $\boldsymbol{k} \sim \mathrm{Bin}(n, 1-\delta)$. For each k, let

$$\mathsf{OPT}_k := \mathbf{E}_{\boldsymbol{y} \sim \{0,1\}^k}\left[\mathbf{E}_{\boldsymbol{x} \sim \boldsymbol{y}}\left[|\mathsf{LCS}(\mathrm{opt}(\boldsymbol{y}), \boldsymbol{x})|\right]\right].$$

It is easy to see that OPT_k is nondecreasing in k,[6] and we have (with $\boldsymbol{k} \sim \mathrm{Bin}(n, 1-\delta)$)

$$L_{1,\mathrm{avg}}(\delta, n) = \sum_{k=0}^{n} \mathbf{Pr}\left[\boldsymbol{k} = k\right] \cdot \mathsf{OPT}_k. \tag{6.25}$$

Let $\delta' = \delta/2$ and $m = (1-\delta')m$. To prove Theorem 6.2, we first show that it suffices to obtain the following upper bound for OPT_m:

$$\mathsf{OPT}_m \leq (1 - c_1\delta')n, \tag{6.26}$$

for some universal positive constant c_1. Consequently it suffices to analyze the optimal one-trace algorithm which is given as input a uniform random string $\boldsymbol{y} \sim \{0,1\}^m$.

Next, we observe that by a simple triangle inequality argument, it suffices to show that

$$\mathop{\mathbf{E}}_{\boldsymbol{y} \sim \{0,1\}^m} \mathop{\mathbf{E}}_{\boldsymbol{x}, \boldsymbol{x}' \sim \boldsymbol{y}}\left[|\mathsf{LCS}(\boldsymbol{x}, \boldsymbol{x}')|\right] \leq (1 - 2c_1\delta')n \tag{6.27}$$

[6] To see this, note that any algorithm that receives a random trace of length k can be simulated by an algorithm that receives a trace of length $k+1$ by randomly deleting one bit from its input trace.

for some constant c_1; in turn, to prove (6.27), it is enough to bound (informally)

$$\mathop{\mathbf{Pr}}_{\boldsymbol{y}\sim\{0,1\}^m,\boldsymbol{x},\boldsymbol{x}'\sim\boldsymbol{y}}\left[\boldsymbol{x} \text{ and } \boldsymbol{x}' \text{ have a "large" matching}\right]. \tag{6.28}$$

In Claim 6.2 we give an upper bound on the probability that a *fixed* large candidate matching M is a valid matching between $\boldsymbol{x}, \boldsymbol{x}' \sim \boldsymbol{y}$. This upper bound is in terms of a quantity that we call score_M, which depends on the candidate matching M and is a random variable whose randomness comes from the sets $\mathbf{S}$ and $\mathbf{S}'$ as in Observation 5.1's description of the distribution of $\boldsymbol{x} \sim \boldsymbol{y}$ and $\boldsymbol{x}' \sim \boldsymbol{y}$. As we show in Claim 6.3, to establish Equation (6.28) it is enough to show that for *every* large candidate matching M, an upper tail bound on $\mathsf{score}_M(\mathbf{S}, \mathbf{S}')$ holds. We prove such a tail bound in Lemma 6.1. The two main steps are (i) showing (in Claim 6.4) that $\mathbf{Pr}[\mathbf{S}$ is not "*well-spaced*"$]$ is very small (see Definition 6.2 for the definition of well-spaced sets), and (ii) showing (in Claim 6.5) that if $\mathbf{S} = S$ is good, then $\mathbf{Pr}_{\mathbf{S}'}[\mathsf{score}(S, \mathbf{S}')$ is large$]$ is very small.

6.2.2 Proof of Theorem 6.2. We may assume that $\delta = \delta(n)$ is at most some sufficiently small universal positive constant, since otherwise the claimed bound follows immediately from Theorem B.1. We will use this assumption in various bounds throughout the proof.

Let $\delta' = \delta/2$ and $m = (1-\delta')n$ (so δ' is $\omega(1/n)$ and at most some sufficiently small universal positive constant as well). By the well-known fact [KB80] that the median of the $\mathrm{Bin}(n, \delta)$ distribution belongs to $\{\lfloor n\delta \rfloor, \lceil n\delta \rceil\}$ (which is at least $\delta' n$ using $\delta = \omega(1/n)$), it follows from (6.25) and the monotonicity of OPT_k that

$$L_{1,\mathrm{avg}}(\delta, n) \leq 0.5 \cdot \mathsf{OPT}_m + 0.5n$$

and thus, to prove Theorem 6.2 it suffices to obtain the upper bound for OPT_m in (6.26).

Instead of working with OPT_m directly, an application of the triangle inequality lets us work with the expression on the LHS of (6.27) which (conveniently) does not involve $\mathrm{opt}(\boldsymbol{y})$:

Claim 6.1. *Suppose that Equation (6.27) holds. Then Equation (6.26) holds.*

Proof. For any $y \in \{0,1\}^m$, any n-bit string $\mathrm{opt}(y)$, and any two n-bit strings x, x', we have that the length of the LCS between x and x' is at least the number of coordinates of $\mathrm{opt}(y)$ that participate both in the optimal matching between $\mathrm{opt}(y)$ and x and in the optimal matching between $\mathrm{opt}(y)$ and x'. Since this number is at least $|\mathsf{LCS}(x, \mathrm{opt}(y))| + |\mathsf{LCS}(x', \mathrm{opt}(y))| - n$, we have that

$$n + \left|\mathsf{LCS}(x, x')\right| \geq \left|\mathsf{LCS}(x, \mathrm{opt}(y))\right| + \left|\mathsf{LCS}(x', \mathrm{opt}(y))\right|. \tag{6.29}$$

It follows that

$$\begin{aligned}
2(1 - c_1\delta')n &= n + (1 - 2c_1\delta')n \\
&\geq n + \mathop{\mathbf{E}}_{\boldsymbol{y}\sim\{0,1\}^m} \mathop{\mathbf{E}}_{\boldsymbol{x},\boldsymbol{x}'\sim\boldsymbol{y}} \left[|\mathsf{LCS}(\boldsymbol{x}, \boldsymbol{x}')|\right] \\
&\geq \mathop{\mathbf{E}}_{\boldsymbol{y}\sim\{0,1\}^m} \mathop{\mathbf{E}}_{\boldsymbol{x},\boldsymbol{x}'\sim\boldsymbol{y}} \left[|\mathsf{LCS}(\boldsymbol{x}, \mathrm{opt}(\boldsymbol{y}))| + |\mathsf{LCS}(\boldsymbol{x}', \mathrm{opt}(\boldsymbol{y}))|\right] \\
&= 2 \mathop{\mathbf{E}}_{\boldsymbol{y}\sim\{0,1\}^m} \mathop{\mathbf{E}}_{\boldsymbol{x}\sim\boldsymbol{y}} \left[|\mathsf{LCS}(\boldsymbol{x}, \mathrm{opt}(\boldsymbol{y}))|\right],
\end{aligned}$$

where the first inequality is by Equation (6.27), the second is by Equation (6.29) (averaged over $\boldsymbol{y}$, $\boldsymbol{x} \sim \boldsymbol{y}$ and $\boldsymbol{x}' \sim \boldsymbol{y}$), and the third is because $\boldsymbol{x}'$ and $\boldsymbol{x}$ are identically distributed. ∎

Given Claim 6.1, our goal in the rest of the proof is to establish Equation (6.27). We note that in Equation (6.27), given the outcome of $\boldsymbol{y}$, the two n-bit strings $\boldsymbol{x}$ and $\boldsymbol{x}'$ are *independently* distributed according to $\boldsymbol{x} \sim \boldsymbol{y}$ and $\boldsymbol{x}' \sim \boldsymbol{y}$; in particular, recalling Observation 5.1, there are two independent draws performed to obtain the sets $\mathbf{S}$ (for $\boldsymbol{x}$) and $\mathbf{S}'$ (for $\boldsymbol{x}'$). This independence will be used heavily in the rest of the argument.

Recalling Observation 5.1, we rewrite Equation (6.27) as

$$\mathop{\mathbf{E}}_{\boldsymbol{y},\mathbf{S},\mathbf{S}',\boldsymbol{r},\boldsymbol{r}'} \left[|\mathsf{LCS}(\boldsymbol{x}, \boldsymbol{x}')|\right] \leq (1 - 2c_1\delta')\, n, \tag{6.30}$$

where $\boldsymbol{y} \sim \{0,1\}^m$, $\mathbf{S}$ and $\mathbf{S}'$ are independent uniform m-element subsets of $[n]$, and $\boldsymbol{r}, \boldsymbol{r}'$ are independent uniform draws from $\{0,1\}^{n-m}$ representing the "rest of the bits" that get filled into the locations in $[n] \setminus \mathbf{S}$ and $[n] \setminus \mathbf{S}'$ to complete the n-bit strings $\boldsymbol{x}$ and $\boldsymbol{x}'$, respectively. Recall that $\boldsymbol{x}$ has $\boldsymbol{y}$ in the m locations of $\mathbf{S}$ and $\boldsymbol{r}$ in the other $n-m$ locations, and $\boldsymbol{x}'$ gets the same $\boldsymbol{y}$ in the locations of $\mathbf{S}'$ and $\boldsymbol{r}'$ in the other locations.

Since the length of the LCS between two strings is the size of the largest matching between them, to establish Equation (6.30) it suffices to prove that

$$\Pr_{\boldsymbol{y},\mathbf{S},\mathbf{S}',\boldsymbol{r},\boldsymbol{r}'}\left[\text{there exists a matching between } \boldsymbol{x} \text{ and } \boldsymbol{x}' \text{ of size } (1-4c_1\delta')n\right] \le 1/2 \tag{6.31}$$

for some universal positive constant c_1. Thus our remaining task is to establish Equation (6.31).

6.2.3 Matchings and scores. Recall from Section 2 that a matching M of size t between two n-bit strings z, z' is a sequence of pairs $M = (M_1, \dots, M_t)$, where

(a) $M_i = (v_i, v_i')$ are such that $1 \le v_1 < \cdots < v_t \le n$, $1 \le v_1' < \cdots < v_t' \le n$, and

(b) for each $i \in [t]$ we have that the two bits z_{v_i} and $z'_{v_i'}$ are the same.

Let us say that a *candidate matching* is a sequence of pairs $M = (M_1, \dots, M_t)$ satisfying (a); if moreover (b) holds for a pair of n-bit strings z and z', we say that the candidate matching M is *valid for* (z, z').

Let $M = (M_1, \dots, M_t)$ be a candidate matching and let

$$S = \{s_1 < \cdots < s_m\} \quad \text{and} \quad S' = \{s_1' < \cdots < s_m'\}$$

be two m-element subsets of $[n]$. We say that an edge $M_i = (v_i, v_i')$ of M *synchs up* with the pair (S, S') if there is some $j \in [m]$ such that $v_i = s_j$ and $v_i' = s_j'$; in words, for some j the candidate matching attempts to match up the j-th element of S with the j-th element of S'. We say the *score of* M *on* (S, S'), denoted $\mathsf{score}_M(S, S')$, is

$$\mathsf{score}_M(S, S') := \Big|\big\{i \in [t] : M_i \text{ synchs up with } (S, S')\big\}\Big|,$$

the number of edges of M that match up corresponding elements of S and S'.

Claim 6.2. *Let $M = (M_1, \dots, M_t)$ be a candidate matching of size t and let S, S' be m-element subsets of $[n]$ such that $\mathsf{score}_M(S, S') = \ell$. Then*

$$\Pr_{\boldsymbol{y},\boldsymbol{r},\boldsymbol{r}'}\left[M \text{ is a valid matching for } (\boldsymbol{x}, \boldsymbol{x}')\right] = \frac{1}{2^{t-\ell}},$$

where $\boldsymbol{x}$ and $\boldsymbol{x}'$ are defined based on $S, S', \boldsymbol{y}, \boldsymbol{r}, \boldsymbol{r}'$ as described after Equation (6.30).

Proof. For each $M_i = (v_i, v_i')$ that synchs up with (S, S'), it is clear that $\boldsymbol{x}_{v_i} = \boldsymbol{x}'_{v_i'}$, because both are the same bit $\boldsymbol{y}_j$ of the string $\boldsymbol{y}$. There are $t - \ell$ remaining equalities

$$\boldsymbol{x}_{v_{i_1}} \stackrel{?}{=} \boldsymbol{x}_{v'_{i_1}}, \dots, \boldsymbol{x}_{v_{i_{t-\ell}}} \stackrel{?}{=} \boldsymbol{x}_{v'_{i_{t-\ell}}}, \quad \text{where } i_1 < \cdots < i_{t-\ell},$$

that must all hold in order for the candidate matching M to be valid for $(\boldsymbol{x}, \boldsymbol{x}')$, corresponding to the $t - \ell$ edges of M that do not synch up with (S, S'). Each of these equalities holds independently with probability $1/2$. This can be seen by considering the $t - \ell$ edges of M successively in increasing order "from left to right": for each $j \in [t - \ell]$, for any given outcome of the bits of $\boldsymbol{y}, \boldsymbol{r}$ and $\boldsymbol{r}'$ that were involved in the first $j - 1$ edges, there is a "fresh random bit" from either $\boldsymbol{y}, \boldsymbol{r}$ or $\boldsymbol{r}'$ involved in the j-th edge that causes the j-th equality to hold with probability $1/2$. ∎

For the rest of the proof of Theorem 6.2, we fix $t := (1 - 4c_1\delta')n$ for some universal constant c_1 to be picked later (recall that that is the size of the matchings that we are concerned with in Equation (6.31)). The following claim states that it suffices to establish that for each fixed size-t candidate matching M, the probability that it has a high score is very low:

CLAIM 6.3. *Suppose that there is a universal positive constant c_1 such that the following inequality holds for each candidate matching M of size $t = (1 - 4c_1\delta')n$:*

$$\Pr_{\mathbf{S},\mathbf{S}' \sim \binom{[n]}{[m]}} \left[\mathsf{score}_M(\mathbf{S}, \mathbf{S}') \geq \big(1 - 3H(4c_1\delta')\big)n \right] \leq \frac{1}{4 \cdot 2^{2H(4c_1\delta')n}}. \tag{6.32}$$

Then Equation (6.31) holds with the same constant c_1.

Proof. We use $\mathbf{Pr}[\mathbf{A}] \leq \mathbf{Pr}[\mathbf{B}] + \mathbf{Pr}[\mathbf{A} \mid \overline{\mathbf{B}}]$ where $\mathbf{A}$ is the event "there exists some valid matching between $\boldsymbol{x}$ and $\boldsymbol{x}'$ of size t" and $\mathbf{B}$ is the event "there exists some candidate matching M of size t with $\mathsf{score}_M(\mathbf{S}, \mathbf{S}') \geq (1 - 3H(4c_1\delta'))n$." There are

$$\binom{n}{4c_1\delta' n}^2 \leq 2^{2H(4c_1\delta')n}$$

many candidate matchings M of size t. A union bound together with Equation (6.32) gives that

$$\mathbf{Pr}[\mathbf{B}] \leq 2^{2H(4c_1\delta')n} \cdot \frac{1}{4 \cdot 2^{2H(4c_1\delta')n}} = \frac{1}{4}.$$

To upperbound $\mathbf{Pr}[\mathbf{A} \mid \overline{\mathbf{B}}]$, fix any particular outcome (S, S') of $(\mathbf{S}, \mathbf{S}')$ such that $\mathsf{score}_M(S, S') < (1 - 3H(4c_1\delta'))n$ holds for every candidate matching M of size t. By Claim 6.2 we have that

$$\Pr_{\boldsymbol{y},\boldsymbol{r},\boldsymbol{r}'} \big[M \text{ is valid for } (\boldsymbol{x}, \boldsymbol{x}') \mid (\mathbf{S}, \mathbf{S}') = (S, S')\big] \leq \frac{1}{2^{t-(1-3H(4c_1\delta'))n}} < \frac{1}{2^{2.5H(4c_1\delta')n}},$$

where the second inequality uses that δ' is at most some sufficiently small absolute constant. By a union bound over all (at most $2^{2H(4c_1\delta')n}$) many candidate matchings M of size t, we see that

$$\mathbf{Pr}\left[\mathbf{A} \mid (\mathbf{S}, \mathbf{S}') = (S, S')\right] \leq 2^{2H(4c_1\delta')n} \cdot \frac{1}{2^{2.5H(4c_1\delta')n}} \leq \frac{1}{4},$$

where the inequality holds since $\delta' = \omega(1/n)$. Hence $\mathbf{Pr}[\mathbf{A} \mid \overline{\mathbf{B}}] \leq 1/4$, and the claim is proved. ∎

For the rest of the proof fix M to be any particular size-t candidate matching. By Claim 6.3, our remaining task is to establish the tail bound on $\mathsf{score}_M(\mathbf{S}, \mathbf{S}')$ that is asserted by Equation (6.32). Since δ' is at most some absolute constant, this is an immediate consequence of the following slightly stronger (and cleaner to state) version:

LEMMA 6.1. *There is a universal positive constant c_1 such that*

$$\Pr_{\mathbf{S},\mathbf{S}' \sim \binom{[n]}{[m]}} \left[\mathsf{score}_M(\mathbf{S}, \mathbf{S}') \geq \big(1 - \sqrt{\delta'}\big)n \right] \leq \frac{1}{4 \cdot 2^{2H(4c_1\delta')n}}. \tag{6.33}$$

6.2.4 Proof of Lemma 6.1. Let the size-t matching M be given by $M = ((v_1, v_1'), \ldots, (v_t, v_t'))$. We define sets $L := \{v_1, \ldots, v_t\}$ and $R := \{v_1', \ldots, v_t'\}$ with $v_1 < \cdots < v_t$ and $v_1' < \cdots < v_t'$.

In the proof of Lemma 6.1 it will be sometimes convenient for us to view $\mathbf{S}$ as a uniform random string from $\{0,1\}^n$ conditioned on containing exactly m ones (and $\mathbf{S}'$ as an independent random string with the same distribution). We write $\{0,1\}^n_m$ to denote the set of all such n-bit strings with exactly m ones.

The key notion for the proof of Lemma 6.1 is the following:

DEFINITION 6.2. *We say that an outcome $S \in \{0,1\}^n_m$ of the random variable $\mathbf{S}$ is* well-spaced *if it has the following property: there are at least $\delta' n/2$ many disjoint intervals $I_1, \ldots, I_{\delta' n/2} \subset [n]$, each of length exactly $1 + 2\beta$ with $\beta := 1/\delta'^{3/4}$, such that for each $j \in [\delta' n/2]$ we have that*

(i) I_j is entirely contained in L (so I_j contains $v_{i_j}, \ldots, v_{i_j+2\beta}$ for some i_j) and moreover, their corresponding indices in R ($v'_{i_j}, \ldots, v'_{i_j+2\beta}$) also form an interval (i.e., $v'_{i_j+2\beta} = v'_{i_j} + 2\beta$); and

(ii) viewing S as a bit-string from $\{0,1\}_m^n$, the subword S_{I_j} of S is $1^\beta 01^\beta$, i.e. there is a 0 exactly in the middle of interval I_j and the other 2β bits in the interval are all 1.

Given Definition 6.2, Lemma 6.1 is an immediate consequence of Claim 6.4 and Claim 6.5 using $\mathbf{Pr}[\mathbf{A}] \le \mathbf{Pr}[\mathbf{B}] + \mathbf{Pr}[\mathbf{A} \mid \overline{\mathbf{B}}]$ where $\mathbf{A}$ is the event "$\mathsf{score}_M(\mathbf{S}, \mathbf{S}') \ge (1-\sqrt{\delta'})n$" and $\mathbf{B}$ is the event "$\mathbf{S}$ is not well-spaced," and taking c_1 to be a suitably small constant relative to those constants hidden in the $\Omega(\cdot)$ of these two claims.

Claim 6.4. *We have*

$$\mathop{\mathbf{Pr}}_{\mathbf{S}\sim\{0,1\}_m^n}\left[\mathbf{S} \text{ is not well-spaced}\right] \le 2^{-\Omega(\delta' \log(1/\delta')n)}.$$

Claim 6.5. *Fix any well-spaced $S \in \{0,1\}_m^n$. Then*

$$\mathop{\mathbf{Pr}}_{\mathbf{S}'\sim\{0,1\}_m^n}\left[\mathsf{score}_M(S, \mathbf{S}') \ge \left(1-\sqrt{\delta'}\right)n\right] \le 2^{-\Omega(\delta' \log(1/\delta'))n}.$$

Proof. [Proof of Claim 6.4] We view the draw of $\mathbf{S}$ as a sequential process in which the outcomes of different groups of coordinates are successively revealed. We first reveal the outcome of $\mathbf{S}_{[n]\setminus L} = S_{[n]\setminus L}$, and we consider the remaining distribution over the outcome of $\mathbf{S}_L$. Let b be the number of 0's in $S_{[n]\setminus L}$. Then the remaining distribution of $\mathbf{S}_L$ is uniform random over all strings in $\{0,1\}^L$ that contains exactly $a := n - m - b$ many zeros. Given that $b \le |[n] \setminus L| = 4c_1\delta' n \le 0.01\delta' n$ (using $c_1 \le 1/400$) we have $a \in [0.99\delta' n, \delta' n]$.

After $\mathbf{S}_{[n]\setminus L}$ is drawn, we can view a draw of $\mathbf{S}_L$ from the above-described distribution as being obtained through a sequential random process, proceeding for a stages, where in the j-th stage, after locations $\boldsymbol{i}_1, \dots, \boldsymbol{i}_{j-1}$ in $[t]$ for zeros have been selected in the first $j-1$ stages, a new uniform random location $\boldsymbol{i}_j$ in $[t] \setminus \{\boldsymbol{i}_1, \dots, \boldsymbol{i}_{j-1}\}$ is selected for the j-th zero (which means that the $v_{\boldsymbol{i}_j}$-th entry of $\mathbf{S}$ is set to zero). After each stage we keep track of the number of locations $i \in [t]$ selected so far such that

1. none of $i-2\beta, \dots, i-1, i+1, \dots, i+2\beta$ was selected so far; and
2. both $v_{i-\beta}, \dots, v_{i+\beta}$ and $v'_{i-\beta}, \dots, v'_{i+\beta}$ form an interval of length $2\beta+1$.

We write $\mathbf{X}_j$ to denote this random variable after j stages. It suffices to show that $\mathbf{X}_a$, after all a stages, is at least $\delta' n/2$ with high probability.

To this end, we first notice that after the j-th stage, the number $\mathbf{X}_j$ can go down from $\mathbf{X}_{j-1}$ by at most two. On the other hand, it goes up by one when $\boldsymbol{i}_j$ is not one of the following "disallowed" locations $i \in [t]$:

1. $v_{i-\beta}, \dots, v_{i+\beta}$ or $v'_{i-\beta}, \dots, v'_{i+\beta}$ does not form an interval; the number of such $i \in [t]$ is at most $2 \cdot 2\beta \cdot (n-t)$.
2. i is within 2β of a location already picked; the number of such i is at most $(4\beta+1)a$.

As a result, the probability that $\mathbf{X}_j$ does not go up by one is at most

$$\frac{4\beta(n-t) + (4\beta+1)a}{t-(j-1)} = \frac{16c_1\delta'^{1/4}n + (4\beta+1)a}{t-(j-1)} \le 5\delta'^{1/4},$$

where we used $a \le \delta' n$. Consequently, the probability that out of the a stages in which a location is chosen, at least $\delta' n/10$ times it does not go up by one is at most

$$2^a \cdot (5\delta'^{1/4})^{\delta' n/10} \le 2^{\delta' n} \cdot (5\delta'^{1/4})^{\delta' n/10} = 2^{-\Omega(\delta' \log(1/\delta')n)},$$

when δ' is sufficiently small. If this does not happen, then $\mathbf{X}_a$ at the end is at least

$$a - (\delta' n/10) - 2 \cdot (\delta' n/10) \ge \delta' n/2$$

using $a \ge 0.99\delta' n$, so the claim is proved. ∎

Proof. [Proof of Claim 6.5] Let S be a well-spaced set in $\{0,1\}^n_m$, and $I_1, \ldots, I_{\delta' n/2}$ be the $\delta' n/2$ intervals in $[n]$ of length $2\beta+1$ each that satisfy the conditions of Definition 6.2. For each I_j, we let $i_j \in [t]$ be such that $I_j = \{v_{i_j-\beta}, \ldots, v_{i_j+\beta}\}$ (so v_{i_j} is the center of I_j). Let

$$I'_j = \left\{v'_{i_j-4/\sqrt{\delta'}}, \ldots, v'_{i_j+4/\sqrt{\delta'}}\right\}$$

for each j. Note that since δ' is at most some sufficiently small constant, we have $4/\sqrt{\delta'} < \beta$ and thus, I'_j's are mutually disjoint intervals in $[n]$ because I_j's satisfy conditions of Definition 6.2.

Let $\mathbf{S}' \sim \{0,1\}^n_m$. We claim that, in order to have $\mathsf{score}_M(S, \mathbf{S}') \geq (1-\sqrt{\delta'})n$, it must be the case that $\mathbf{S}'$ has at least $\delta' n/4$ many zero entries in the union of I'_j. To see this, suppose that $\mathbf{S}'$ has no more than $\delta' n/4$ many zeros in the union of I'_j. Then at least $\delta' n/4$ many I'_j's have all ones in $\mathbf{S}'$. For each such j, given that $S_{I_j} = 1^\beta 0 1^\beta$, we have that either

$$\left(v_{i_j-4/\sqrt{\delta'}}, v'_{i_j-4/\sqrt{\delta'}}\right), \ldots, \left(v_{i_j-1}, v'_{i_j-1}\right) \quad \text{or} \quad \left(v_{i_j+1}, v'_{i_j+1}\right), \ldots, \left(v_{i_j+4/\sqrt{\delta'}}, v'_{i_j+4/\sqrt{\delta'}}\right)$$

are not synched. As a result, the number of pairs in M that are not synched in $(S, \mathbf{S}')$ is at least $(\delta' n/4)\cdot(4/\sqrt{\delta'}) \geq \sqrt{\delta'} n$ and thus, the score is at most $(1-\sqrt{\delta'})n$.

Finally we bound the probability of $\mathbf{S}' \sim \{0,1\}^n_m$ having at least $\delta' n/4$ many zeros in the union of I'_j. Given that the union has size

$$\frac{\delta' n}{2} \cdot \left(\frac{8}{\sqrt{\delta'}} + 1\right) < 5\sqrt{\delta'} n,$$

the probability is at most (where the summand r is the number of zeros in the union of I'_j)

$$\sum_{r=\delta' n/4}^{\delta' n} \frac{\binom{5\sqrt{\delta'}n}{r} \cdot \binom{n-5\sqrt{\delta'}n}{\delta' n - r}}{\binom{n}{\delta' n}} \leq \left(\frac{3\delta' n}{4} + 1\right) \cdot \binom{5\sqrt{\delta'}n}{\delta' n/4} \cdot \frac{\binom{n-5\sqrt{\delta'}n}{3\delta' n/4}}{\binom{n}{\delta' n}}$$

given that the terms are maximized at $r = \delta' n/4$. Using $\binom{n}{k} \leq (en/k)^k$, we have

$$\binom{5\sqrt{\delta'}n}{\delta' n/4} \leq \left(\frac{60}{\sqrt{\delta'}}\right)^{\delta' n/4}.$$

On the other hand, we have

$$\frac{\binom{n-5\sqrt{\delta'}n}{3\delta' n/4}}{\binom{n}{\delta' n}} \leq \frac{\binom{n}{3\delta' n/4}}{\binom{n}{\delta' n}} = \frac{(n-\delta' n)! \cdot (\delta' n)!}{(n-3\delta' n/4)! \cdot (3\delta' n/4)!} \leq \left(\frac{\delta' n}{n-\delta' n}\right)^{\delta' n/4} \leq (2\delta')^{\delta' n/4}.$$

As a result, the probability is at most

$$\left(\frac{3\delta' n}{4} + 1\right) \cdot \left(120\sqrt{\delta'}\right)^{\delta' n/4} = 2^{-\Omega(\delta' \log(1/\delta') n)}$$

since δ' is at most some sufficiently small constant. This finishes the proof of Claim 6.5. ∎

References

[BC22] Boris Bukh and Christopher Cox. Periodic words, common subsequences and frogs. *The Annals of Applied Probability*, 32(2):1295 – 1332, 2022. 1.1.2

[BCF+19] Frank Ban, Xi Chen, Adam Freilich, Rocco A. Servedio, and Sandip Sinha. Beyond trace reconstruction: Population recovery from the deletion channel. In *60th IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, pages 745–768. IEEE Computer Society, 2019. 1, 1, 1

[BCSS19] Frank Ban, Xi Chen, Rocco A. Servedio, and Sandip Sinha. Efficient average-case population recovery in the presence of insertions and deletions. In *APPROX/RANDOM 2019*, volume 145 of *LIPIcs*, pages 44:1–44:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. 1, 1, 1

[BKKM04] Tuğkan Batu, Sampath Kannan, Sanjeev Khanna, and Andrew McGregor. Reconstructing strings from random traces. In *Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2004*, pages 910–918, 2004. 1, 1, 1

[BM14] Boris Bukh and Jie Ma. Longest common subsequences in sets of words. *SIAM J. Discret. Math.*, 28(4):2042–2049, 2014. 1.1.1, 3.2.1

[Buk22] B. Bukh. Personal communication, 2022. 1.1.2

[CDK21] Diptarka Chakraborty, Debarati Das, and Robert Krauthgamer. Approximate trace reconstruction via median string (in average-case). In *41st IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, volume 213 of *LIPIcs*, pages 11:1–11:23. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. 1, 2, 1.2

[CDL+21a] Xi Chen, Anindya De, Chin Ho Lee, Rocco A. Servedio, and Sandip Sinha. Polynomial-time trace reconstruction in the low deletion rate regime. In *12th Innovations in Theoretical Computer Science Conference*, volume 185 of *LIPIcs*, pages 20:1–20:20, 2021. 1

[CDL+21b] Xi Chen, Anindya De, Chin Ho Lee, Rocco A. Servedio, and Sandip Sinha. Polynomial-time trace reconstruction in the smoothed complexity model. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms*, pages 54–73, 2021. 1, 1

[CDL+22] Xi Chen, Anindya De, Chin Ho Lee, Rocco A. Servedio, and Sandip Sinha. Near-optimal average-case approximate trace reconstruction from few traces. In *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms (SODA 2022)*, pages 779–821, 2022. 1, 1, 1, 1, 1, 2, 1.1.1, 1.1.1

[Cha21a] Zachary Chase. New lower bounds for trace reconstruction. *Ann. Inst. H. Poincaré Probab. Statist.*, 57(2):627–643, 2021. 1, 1, 1

[Cha21b] Zachary Chase. Separating words and trace reconstruction. In *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*, pages 21–31. ACM, 2021. 1, 1

[CP21] Z. Chase and Y. Peres. Approximate trace reconstruction of random strings from a constant number of traces. Available at https://arxiv.org/abs/2107.06454, 2021. 1, 1, 1, 1, 1

[CS75] Václáv Chvatal and David Sankoff. Longest common subsequences of two random sequences. *J. Appl. Probability*, 12:306–315, 1975. 6.1, 6.1, 6.1

[DOS17] Anindya De, Ryan O'Donnell, and Rocco A. Servedio. Optimal mean-based algorithms for trace reconstruction. In *Proceedings of the 49th ACM Symposium on Theory of Computing (STOC)*, pages 1047–1056, 2017. 1, 1

[DRRS21] Sami Davies, Miklos Z. Rácz, Cyrus Rashtchian, and Benjamin G. Schiffer. Approximate trace reconstruction: Algorithms. In *IEEE International Symposium on Information Theory*, 2021. 1, 2

[GHS20] Venkatesan Guruswami, Bernhard Haeupler, and Amirbehshad Shahrasbi. Optimally resilient codes for list-decoding from insertions and deletions. In *Proccedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 524–537. ACM, 2020. 1.1.1, 1.1.1, 3.1.1, 3.2.1, 3.2.2, 3.3, 3.2.2, 3.2.2, 3.2.2, C

[GSZ21] Elena Grigorescu, Madhu Sudan, and Minshen Zhu. Limitations of mean-based algorithms for trace reconstruction at small distance. In *IEEE International Symposium on Information Theory*, 2021. 1, 2

[HHP18] Lisa Hartung, Nina Holden, and Yuval Peres. Trace reconstruction with varying deletion probabilities. In *Proceedings of the Fifteenth Workshop on Analytic Algorithmics and Combinatorics, ANALCO 2018, New Orleans, LA, USA, January 8-9, 2018.*, pages 54–61, 2018. 1

[HMPW08] Thomas Holenstein, Michael Mitzenmacher, Rina Panigrahy, and Udi Wieder. Trace reconstruction with constant deletion probability and related results. In *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2008*, pages 389–398, 2008. 1, 1

[HPP18] Nina Holden, Robin Pemantle, and Yuval Peres. Subpolynomial trace reconstruction for random strings and arbitrary deletion probability. In *Conference On Learning Theory, COLT 2018, Stockholm, Sweden, 6-9 July 2018*, volume 75 of *Proceedings of Machine Learning Research*, pages 1799–1840. PMLR, 2018. 1, 1

[HPPZ19] Nina Holden, Robin Pemantle, Yuval Peres, and Alex Zhai. Subpolynomial trace reconstruction for random strings and arbitrary deletion probability. *Mathematical Statistics and Learning*, 2(3/4):275–309, 2019. 1, 1, 1

[Kal73] V. V. Kalashnik. Reconstruction of a word from its fragments. *Computational Mathematics and Computer Science (Vychislitel'naya matematika i vychislitel'naya tekhnika), Kharkov*, 4:56–57, 1973. 1

[KB80] R. Kaas and J.M. Buhrman. Mean, Median and Mode in Binomial Distributions. *statistica neerlandica*, 34(1):13–18, 1980. 6.2.2, B

[KLM05] Marcos Kiwi, Martin Loebl, and Jiří Matoušek. Expected length of the longest common subsequence for large alphabets. *Advances in Mathematics*, 197(2):480–498, 2005. 6.1

[KMMP19] Akshay Krishnamurthy, Arya Mazumdar, Andrew McGregor, and Soumyabrata Pal. Trace reconstruction: Generalized and parameterized. In *27th Annual European Symposium on Algorithms, ESA 2019*, volume 144 of *LIPIcs*, pages 68:1–68:25. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. 1

[Lev01a] Vladimir Levenshtein. Efficient reconstruction of sequences. *IEEE Transactions on Information Theory*, 47(1):2–22, 2001. 1

[Lev01b] Vladimir Levenshtein. Efficient reconstruction of sequences from their subsequences or supersequences. *Journal of Combinatorial Theory Series A*, 93(2):310–332, 2001. 1

[Lue09] George S. Lueker. Improved bounds on the average length of longest common subsequences. *J. ACM*, 56(3):17:1–17:38, 2009. 1.1.2

[MPV14] Andrew McGregor, Eric Price, and Sofya Vorotnikova. Trace reconstruction revisited. In *Proceedings of the 22nd Annual European Symposium on Algorithms*, pages 689–700, 2014. 1, 1, 1

[NP17] Fedor Nazarov and Yuval Peres. Trace reconstruction with $\exp(O(n^{1/3}))$ samples. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017*, pages 1042–1046, 2017. 1, 1

[NR21] Shyam Narayanan and Michael Ren. Circular Trace Reconstruction. In *12th Innovations in Theoretical Computer Science Conference (ITCS 2021)*, pages 18:1–18:18, 2021. 1, 1, 1

[PZ17] Yuval Peres and Alex Zhai. Average-case reconstruction for the deletion channel: Subpolynomially many traces suffice. In *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 228–239. IEEE Computer Society, 2017. 1, 1

[SB21] Jin Sima and Jehoshua Bruck. Trace reconstruction with bounded edit distance. In *IEEE International Symposium on Information Theory*, 2021. Manuscript, available at https://arxiv.org/abs/2102.05372. 1, 2

[SDDF18] Sundara Rajan Srinivasavaradhan, Michelle Du, Suhas Diggavi, and Christina Fragouli. On maximum likelihood reconstruction over multiple deletion channels. In *IEEE International Symposium on Information Theory, ISIT 2018*, pages 436–440, 2018. 1, 2

[Tur66] L. Richard Turner. Inverse of the vandermonde matrix with applications. NASA technical note D-3547 available at available at http://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/19660023042.pdf., 1966. 3.2

[vL82] J. H. van Lint. *Introduction to Coding Theory*. Springer Science+Business Media, 1982. 2.1

[Wik22a] Wikipedia contributors. Chvátal-Sankoff constants. Wikipedia, The Free Encyclopedia, accessed June 23, 2022. https://en.wikipedia.org/wiki/Chvátal-Sankoff_constants. 1.1.2

[Wik22b] Wikipedia contributors. Subadditivity. Wikipedia, The Free Encyclopedia, accessed June 23, 2022. https://en.wikipedia.org/wiki/Subadditivity. 1.1.2

A An upper bound on average-case zero-trace reconstruction

We recall from Section 1.1.2 that in the asymptotic limit, the best possible performance of any zero-trace average-case reconstruction algorithm is given by

$$c_2 = \lim_{n\to\infty} \max_{z\in\{0,1\}^n} \frac{\mathbf{E}_{\boldsymbol{x}\sim\{0,1\}^n}[|\mathsf{LCS}(\boldsymbol{x},\boldsymbol{z})|]}{n},$$

and from Section 5 that this quantity equals $\lim_{n\to\infty} \frac{L_{0,\mathrm{avg}}(n)}{n}$.

Via an involved analysis, Bukh and Cox show that $\mathbf{E}_{\boldsymbol{x}\sim\{0,1\}^n}[|\mathsf{LCS}(\boldsymbol{x},w)|] \geq 0.82118$ where w is the n-bit string $(01101110100101100100011011010)^{n/28}$, and hence $c_2 \geq 0.82118$. We give an upper bound on c_2:

Claim A.1. $c_2 \leq 0.88999$.

Proof. Fix $z \in \{0,1\}^n$ to be the optimal string that maximizes $\mathbf{E}_{\boldsymbol{x}\sim\{0,1\}^n}[|\mathsf{LCS}(\boldsymbol{x},\boldsymbol{z})|]$. The claimed bound on c_2 follows from

$$\Pr_{\boldsymbol{x}\sim\{0,1\}^n}[z \text{ has a matching of size } 0.88999n \text{ with } \boldsymbol{x}] \leq o(1), \tag{A.1}$$

which we establish below by showing that

$$\sum_{S\subseteq[n],|S|=0.88999n} \Pr_{\boldsymbol{x}\sim\{0,1\}^n}[z_S \text{ matches entirely into } \boldsymbol{x}] \leq o(1). \tag{A.2}$$

Via a union bound, Equation (A.2) in turn follows from showing that for any t-bit string y, where $t := 0.88999n$, we have

$$\Pr_{\boldsymbol{x}\sim\{0,1\}^n}[y \text{ matches entirely into } \boldsymbol{x}] = \frac{o(1)}{\binom{n}{0.88999n}}. \tag{A.3}$$

Fix any t-bit string y and any n-bit string x. The "greedy strategy" for (attempting to) entirely match y into x is the approach which maintains two pointers p_y (into the coordinates of y) and p_x and scans across x by successively incrementing p_x, matching each coordinate of y and incrementing p_y whenever it is possible to do so. We recall the following well-known fact:

CLAIM A.2. (GREEDY MATCHING IS OPTIMAL FOR ENTIRELY MATCHING ONE STRING INTO ANOTHER) *There is some matching that entirely matches y into x if and only if the greedy strategy succeeds in entirely matching y into x.*

We return to establishing Equation (A.3). By Claim A.2, y matches entirely into $\boldsymbol{x}$ if and only if the greedy strategy matches y entirely into $\boldsymbol{x}$. We may view a uniform $\boldsymbol{x} \sim \{0,1\}^n$ as being generated by successively tossing coins for the successive bits of $\boldsymbol{x}$; from this perspective it is clear that $\mathbf{Pr}_{\boldsymbol{x}\sim\{0,1\}^n}[$the greedy strategy successfully matches y entirely into $\boldsymbol{x}]$ is precisely the probability that a sequence of n fair coin tosses has at least t "heads" (the i-th coin toss coming up "heads" corresponds to the i-th bit $\boldsymbol{x}_i$ matching the bit of y currently pointed to by p_y). By Fact 2.1, this probability is at most

$$\frac{2^{H(0.11001)n}}{2^n}, \tag{A.4}$$

so again using Fact 2.1 and $2H(0.11001) < 1$ we get that $(A.4) = \frac{o(1)}{\binom{n}{0.88999n}}$ as required. ∎

B A simple upper bound on average-case one-trace reconstruction in the small deletion rate regime

In this section we give a simple upper bound on the best possible expected LCS that any one-trace algorithm can achieve in the average-case small-deletion-rate regime. The argument, which is based on a union bound over all possible matchings of a given size, is significantly simpler than the proof of Theorem 6.2, but it yields a result that is quantitatively weaker by a $\Theta(\log(1/\delta))$ factor.

THEOREM B.1. (WEAK AVERAGE-CASE UPPER BOUND ON ANY ALGORITHM, SMALL DELETION RATE) *Let $\delta = \delta(n)$ be any $\omega(1/n)$ deletion rate. There is an absolute constant $c > 0$ such that for sufficiently large n we have $L_{1,\mathrm{avg}}(\delta, n) \le (1 - c\delta/\log(1/\delta))n$.*

Proof. As in the beginning of the proof of Theorem 6.2, by recalling the well-known fact [KB80] that the median of the $\mathrm{Bin}(n,\delta)$ distribution belongs to $\{\lfloor n\delta \rfloor, \lceil n\delta \rceil\}$, since $\delta = \omega(1/n)$ we have that with probability $\Omega(1)$ the length $|\boldsymbol{y}|$ of a random trace $\boldsymbol{y}$ drawn from the δ-deletion channel is at least $(1 - \Omega(\delta))n =: (1 - \delta')n$. Hence to upper bound $L_{1,\mathrm{avg}}(\delta, n)$ as claimed, it suffices to show the following: for any one-trace algorithm A that is given as input a uniform random trace $\boldsymbol{y}$, of length exactly $(1-\delta')n$, from a uniform random source string $\boldsymbol{x} \sim \{0,1\}^n$, we have

$$\mathop{\mathbf{Pr}}_{\boldsymbol{x}\sim\{0,1\}^n}\left[A \text{ outputs a hypothesis string } z \text{ with } |\mathsf{LCS}(\boldsymbol{x}, z)| \ge \left(1 - \frac{c\delta'}{\log(1/\delta')}\right) n\right] \le 0.9. \tag{B.5}$$

We first recall from Corollary 2.1 that given a trace $\boldsymbol{y}$ of length $(1-\delta')n$ from a uniform $\boldsymbol{x} \sim \{0,1\}^n$, the $\delta' n$ bits of $\boldsymbol{x}_{\mathbf{D}}$ that are missing from $\boldsymbol{y}$ are independent and uniform random. Next, we note that any candidate matching μ of size $(1-\tau)n$ between a source string $x \in \{0,1\}^n$ and a hypothesis string $z \in \{0,1\}^n$ is completely specified by two subsets $S = \{i_1 < \cdots < i_{\tau n}\} \subset [n]$ and $S' = \{j_1 < \cdots < j_{\tau n}\} \subset [n]$ of size τn, where S (S', respectively) is the set of positions in x (positions in z, respectively) that do not participate in the matching.

Fix any hypothesis string $z \in \{0,1\}^n$ (here z may depend on the trace $\boldsymbol{y} \sim \{0,1\}^{(1-\delta')n}$ that algorithm A receives as input). Consider a fixed candidate matching μ of size $(1-\tau)n$ between z and $\boldsymbol{x}$, defined by two fixed sets S, S' as described above. For $\tau < \delta'/2$ (which will be the case given our final parameter setting for τ), even if all τn positions in S are contained in the deleted locations $\mathbf{D}$, there are at least $(\delta - \tau)n \ge (\delta'/2)n$ bits in $\boldsymbol{x}_{\mathbf{D}}$ that are not present in $\boldsymbol{y}$ but are matched to some bits of z by the candidate matching μ. As mentioned above, these bits are independently uniform random, and so the probability that μ successfully matches all of those (at least) $(\delta'/2)n$ bits with the right outcomes of their partners in z is at most $2^{-(\delta'/2)n}$. It follows that $\mathbf{Pr}_{\boldsymbol{x}}[$the candidate matching μ is a valid matching between z and $\boldsymbol{x}] \le 2^{-(\delta'/2)n}$. Hence we have

$$\begin{aligned}&\mathop{\mathbf{Pr}}_{\boldsymbol{x}\sim\{0,1\}^n}[\text{there exists some matching of size } (1-\tau)n \text{ between } \boldsymbol{x} \text{ and } z]\\ &\le \binom{n}{\tau n}^2 \cdot 2^{-(\delta'/2)n} \le 2^{(2H(\tau)-\delta'/2)n} \le 2^{-(\delta'/4)n} \le 0.9,\end{aligned}$$

where the first inequality is by a union bound over all $\binom{n}{\tau n}^2$ many candidate matchings of size $(1-\tau)n$, the second is Fact 2.1, the third holds by choosing $\tau = c\delta'/\log(1/\delta')$ for a suitable absolute constant c, and the fourth (with room to spare) is because δ', like δ, is $\omega(1/n)$. ∎

C No constant-size $(2/3+\varepsilon)n$-LCS cover for any constant $\varepsilon > 0$

Claim C.1. *For any positive constant ε, any $(2/3+\varepsilon)n$-LCS cover $S \subseteq \{0,1\}^n$ must have size $\Omega(\log n)$.*

Proof. Let ε be a positive constant and let $\varepsilon' = 6\varepsilon$. Let $S \subseteq \{0,1\}^n$ be a $(2/3+\varepsilon)n$-LCS cover for strings of length n. As explained in Section 3.2.2, by arguments given in the proof of Theorem 1.4 of [GHS20], for any $x \in \{0,1\}^n$ (and hence in particular for each string $x \in S$), there can be at most $1200/\varepsilon'^3$ many strings $a \in C_{n,\varepsilon'}$ that have $|\mathsf{LCS}(x,a)| \geq (2/3+\varepsilon'/6)n = (2/3+\varepsilon)n$. Say a string $a \in C_{n,\varepsilon'}$ is *covered* if there is some string $x \in S$ such that $|\mathsf{LCS}(x,a)| \geq (2/3+\varepsilon)n$; it follows that at most $|S| \cdot (1200/\varepsilon'^3)$ strings in $C_{n,\varepsilon'}$ are covered. Given that every string in $C_{n,\varepsilon'}$ is covered (by the assumption that S is a $(2/3+\varepsilon)n$-LCS cover), we have

$$|S| \cdot \frac{1200}{\varepsilon'^3} \geq \left|C_{n,\varepsilon'}\right| = \frac{\log n}{\log(1/\varepsilon'^4)},$$

from which the $\Omega(\log n)$ lower bound on $|S|$ follows. ∎