Near-Optimal Trace Reconstruction for Mildly Separated Strings

Anders Aamand
BARC, University of Copenhagen, Denmark

Allen Liu
Massachusetts Institute of Technology, Cambridge, MA, USA

Shyam Narayanan
Citadel Securities, Miami, FL, USA

Abstract

In the *trace reconstruction* problem our goal is to learn an unknown string $x \in \{0,1\}^n$ given independent *traces* of x. A trace is obtained by independently deleting each bit of x with some probability δ and concatenating the remaining bits. It is a major open question whether the trace reconstruction problem can be solved with a polynomial number of traces when the deletion probability δ is constant. The best known upper bound and lower bounds are respectively $\exp(\tilde{O}(n^{1/5}))$ [7] and $\tilde{\Omega}(n^{3/2})$ [6]. Our main result is that if the string x is *mildly separated*, meaning that the number of zeros between any two ones in x is at least $\operatorname{polylog} n$, and if δ is a sufficiently small constant, then the trace reconstruction problem can be solved with $O(n \log n)$ traces and in polynomial time.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms

Keywords and phrases Trace Reconstruction, deletion channel, sample complexity

Digital Object Identifier 10.4230/LIPIcs.ICALP.2025.3

Category Track A: Algorithms, Complexity and Games

Related Version *Previous Version*: `https://arxiv.org/abs/2411.18765`

Funding *Anders Aamand*: This work was supported by the DFF-International Postdoc Grant 0164-00022B and by the VILLUM Foundation grants 54451 and 16582.
Allen Liu: This work was supported in part by an NSF Graduate Research Fellowship, a Hertz Fellowship, and a Citadel GQS Fellowship.
Shyam Narayanan: Supported by the NSF TRIPODS Program, an NSF Graduate Fellowship, and a Google Fellowship.

1 Introduction

Trace reconstruction is a well-studied problem at the interface of string algorithms and learning theory. Informally, the goal of trace reconstruction is to recover an unknown string x given several independent noisy copies of the string.

Formally, fix an integer $n \geq 1$ and a deletion parameter $\delta \in (0,1)$. Let $x \in \{0,1\}^n$ be an unknown binary string with x_i representing the ith bit of x. Then, a *trace* $\tilde{x}$ of x is generated by deleting every bit x_i independently with probability δ (and retaining it otherwise), and concatenating the retained bits together. For instance, if $x = 01001$ and we delete the second and third bits, the trace would be 001 (from the first, fourth, and fifth bits of x). For a fixed string x, note that the trace follows some distribution over bitstrings, where the randomness comes from which bits are deleted. In *trace reconstruction*, we assume we are given N i.i.d. traces $\tilde{x}^{(1)}, \ldots, \tilde{x}^{(N)}$, and our goal is to recover the original string x with high probability.

52nd International Colloquium on Automata, Languages, and Programming (ICALP 2025).
Editors: Keren Censor-Hillel, Fabrizio Grandoni, Joël Ouaknine, and Gabriele Puppis
Article No. 3; pp. 3:1–3:20
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

The trace reconstruction problem has been a very well studied problem over the past two decades [23, 24, 3, 21, 20, 34, 25, 16, 30, 31, 17, 18, 19, 6, 10, 9, 7, 32]. There have also been numerous generalizations or variants of trace reconstruction studied in the literature, including coded trace reconstruction [13, 4], reconstructing mixture models [1, 2, 28], reconstructing alternatives to strings [14, 22, 29, 26, 33, 27], and approximate trace reconstruction [15, 8, 5, 11, 12].

In perhaps the most well-studied version of trace reconstruction, x is assumed to be an arbitrary n-bit string and the deletion parameter δ is assumed to be a fixed constant independent of n. In this case, the best known algorithm requires $e^{\tilde{O}(n^{1/5})}$ random traces to reconstruct x with high probability [7]. As we do not know of any polynomial-time (or even polynomial-sample) algorithms for trace reconstruction, there have been many works making distributional assumptions on the string x, such as x being a uniformly random string [20, 25, 31, 19, 32] or x being drawn from a "smoothed" distribution [10]. An alternative assumption is that the string x is parameterized, meaning that x comes from a certain "nice" class of strings that may be amenable to efficient algorithms [22, 15].

In this work, we also wish to understand parameterized classes of strings for which we can solve trace reconstruction efficiently. Indeed, we give an algorithm using polynomial traces and runtime, that works for a general class of strings that we call L-separated strings. This significantly broadens the classes of strings for which polynomial-time algorithms are known [22].

Main Result

Our main result concerns trace reconstruction of strings that are *mildly separated*. We say that a string x is L-separated if the number of zeros between any two consecutive ones is at least L. Depicting a string $x \in \{0,1\}^n$ with t ones as

$$\underbrace{0\dots0}_{a_0 \text{ times}}\ 1\ \underbrace{0\dots0}_{a_1 \text{ times}}\ 1\cdots1\ \underbrace{0\dots0}_{a_t \text{ times}},$$

it is L-separated if and only if $a_i \ge L$ for each i with $1 \le i \le t-1$. Note that we make no assumptions on a_0 or a_t. Our main result is as follows.

▶ **Theorem 1.** *There exists an algorithm that solves the trace reconstruction problem with high probability in n on any L-separated string x, provided that $L \ge C(\log n)^8$ for a universal constant C, and that the deletion probability is at most some universal constant c_0. The algorithm uses $N = O(n \log n)$ independently sampled traces of x, $\tilde{x}^{(1)}, \dots, \tilde{x}^{(N)}$, and runs in* $\text{poly}(n)$ *time.*

We note that the number of traces is nearly optimal. Even distinguishing between two strings x, x' which contain only a single one at positions $\lfloor n/2 \rfloor$ and $\lfloor n/2 \rfloor + 1$ respectively, requires $\Omega(n)$ traces to succeed with probability $1/2 + \Omega(1)$.

While trace reconstruction is known to be solvable very efficiently for random strings [19, 32], there are certain structured classes of strings that appear to be natural hard instances for existing approaches. Our algorithm can be seen as solving one basic class of hard instances. It is worth noting the work by [9] which studies the trace reconstruction problem when the deletion probability δ is sub-constant. They show that the simple Bitwise Majority Alignment (BMA) algorithm from [3] can succeed with $1/n^{o(1)}$ deletion probability as long as the original string does not contain *deserts* – which are highly repetitive blocks where some short substring is repeated many times. They then combine this with an algorithm for reconstructing repetitive blocks – but this part of their algorithm requires a significantly smaller deletion probability of $\delta \le 1/n^{1/3+\varepsilon}$. This suggests that strings containing many repetitive blocks are a natural hard instance and good test-bed for developing new algorithms and approaches. L-separated strings can be thought of as the simplest class of highly repetitive strings (where the repeating pattern is just a 0), where every repetition has length at least L.

Comparison to Related Work

Most closely related to our work is the result by Krishnamurthy et al. [22] stating that if x has at most k ones and if each pair of ones is separated by a *run* of zeros of length $\Omega(k \log n)$, then x can be recovered in polynomial time from $\text{poly}(n)$ many traces. In particular, for strings with $k = O((\log n)^7)$ ones, the required separation is milder than ours, albeit not below $\Omega(\log n)$. Our algorithm works in general assuming a $\text{polylog}\, n$ separation of the ones but with no additional requirement on the number of ones: indeed, we could even have $\frac{n}{\text{polylog}\, n}$ ones. With no sparsity assumptions, [22] would need to set $L \ge \Omega(\sqrt{n \log n})$, as a $\sqrt{n \log n}$-separated string can be $\Theta(\sqrt{n/\log n})$-sparse in the worst case. The techniques of [22] are also very different than ours. They recursively cluster the positions of the ones in the observed traces to correctly align a large fraction of the ones in the observed traces to ones in the string x. In contrast, our algorithm works quite differently and is of a more sequential nature processing the traces from left to right (or right to left). See Section 1.1 for a discussion of our algorithm.

Another paper studying strings with large runs is by Davies et al. [15]. They consider *approximate* trace reconstruction, specifically how many traces are needed to approximately reconstruct x up to edit distance εn under various assumptions on the lengths of the runs of zeros and ones in x. Among other results but most closely related to ours, they show that one can ε-approximately reconstruct x using $O((\log n)/\varepsilon^2)$ traces if the runs of zeros have length $\gg \frac{\log n}{\varepsilon}$ and if the runs of ones are all of length $\le C \log n$ or $\gg 3C \log n$ for a constant C (e.g. they could have length one as in our paper). However, for exact reconstruction, they would need to set $\varepsilon < 1/n$, which means they do not provide any nontrivial guarantees in our setting.

1.1 Technical Contributions

In this section, we give a high level overview of our techniques. Recall that we want to reconstruct a string $x \in \{0,1\}^n$ from independent traces $\tilde{x}$ where we assume that x is mildly separated. More concretely, we assume that there are numbers $a_0, \dots, a_t \gg \text{polylog}\, n$ such that x consists of a_0 zeros followed by a one, followed by a_1 zeros followed by a one and so on, with the last a_t bits of x being zero. Writing $a_{\le i} = \sum_{0 \le j \le i} a_j$, we thus have that there are t ones in x at positions $a_{\le i} + i + 1$ for $0 \le i \le t-1$.

Note that a retained bit in a trace $\tilde{x}$ naturally corresponds to a bit in x. More formally, for a trace $\tilde{x}$ of length ℓ, let $i_1 < \cdots < i_\ell$ be the ℓ positions in x where the bit was retained when generating $\tilde{x}$ so that $\tilde{x} = x_{i_1} \cdots x_{i_\ell}$. Then, the correspondence is defined by the map from $[\ell]$ to $[n]$ mapping $j \mapsto i_j$. We think of this map as the correct *alignment* of $\tilde{x}$ to x.

Our main technical contribution is an alignment algorithm (see Algorithm 1) which takes in some $m \le t$ and estimates $b_0, \dots, b_{m-1}$ of $a_0, \dots, a_{m-1}$ satisfying that for all i, $|b_i - a_i| = O(\sqrt{a_i \log n})$, and correctly aligns the one in a trace $\tilde{x}$ corresponding to the m'th one of x with probability $1 - O(\delta)$ (where the randomness is over the draw of $\tilde{x}$ – naturally, this requires that the m'th one of x was not deleted).

Moreover, we ensure that the alignment procedure with high probability, say $1-O(n^{-100})$, never aligns a one in $\tilde{x}$ too far to the right in x: if the one in $\tilde{x}$ corresponding to the m_0'th one of x is aligned to the m'th one of x, then $m \leq m_0$. We will refer to this latter property by saying that the algorithm is *never ahead* with high probability. If $m < m_0$, we say that the algorithm is *behind*. Thus, to show that the algorithm correctly aligns the m'th one, it suffices to show that the probability that the algorithm is behind is $O(\delta)$.

We first discuss how to implement this alignment procedure and then afterwards we discuss how to complete the reconstruction by using this alignment procedure.

The alignment procedure of Algorithm 1

The main technical challenge of this paper is the analysis of Algorithm 1. Let us first describe on a high level how the algorithm works. For $0 \leq j \leq j' \leq m$, we write $b_{j:j'} = \sum_{i=j}^{j'-1} b_j$. Suppose that the trace $\tilde{x}$ consists of s_0 zeros followed by a one followed by s_1 zeros followed by a one and so on. The algorithm first attempts to align the first one in $\tilde{x}$ with a one in x by finding the minimal j_0 such that $(1-\delta) \cdot b_{0:j_0}$ is within $C \log n \sqrt{b_{0:j_0}}$ of s_0 for a sufficiently large C. Inductively, having determined j_i (that is the alignment of the i'th one of $\tilde{x}$), it looks for the minimal $j_{i+1} > j_i$ satisfying that there is a $j_i \leq j' < j_{i+1}$ such that $b_{j':j_{i+1}} \cdot (1-\delta)$ is within $C \log n \sqrt{b_{j':j_{i+1}}}$ of s_{i+1}. Intuitively, when looking at the i'th one in the trace, we want to find the *earliest* possible location in the real string (which has gaps estimated by $b_0, b_1, \dots$) that could plausibly align with the one in the trace.

It is relatively easy to check that the algorithm is never ahead with very high probability. Indeed, by concentration bounds on the number of deleted zeros and the fact that $|b_j - a_j| = O(\sqrt{a_j \log n})$ for all $j \leq m$, it always has the option of aligning the $(i+1)$'st one in $\tilde{x}$ to the correct one in x. However, it might align to an earlier one in x since it is looking for the *minimum* j_{i+1} such that an alignment is possible. For a very simple example, suppose that $a_0 = n^{\Omega(1)}$ and $a_1 = \dots = a_m = b_1 = \dots = b_m = \text{polylog}(n)$. If the first $k < m$ ones of x are deleted and the $(k+1)$'st one is retained, the algorithm will align the retained one (which corresponds to the $(k+1)$'st one of x) with the first one of x resulting in the aligning algorithm being k steps behind. Moreover, the algorithm will remain k steps behind all the way up to the m'th one of x. The probability of this happening is $\Theta(\delta^k)$. To prove that the probability of the algorithm being behind when aligning the m'th one of x is at most $1-O(\delta)$, we prove a much stronger statement which is amenable to an inductive proof, essentially stating that this is the worst that can happen: The probability of the algorithm being k steps behind at any fixed point is bounded by $(C\delta)^k$ for a constant C. In particular, we show that there is a sort of amortization – whenever there is a substring that can cause the algorithm to fall further behind with some probability (i.e. if certain bits are deleted), the substring also helps the algorithm catch back up if it is already behind.

The algorithm is not too far behind

Proving that the algorithm cannot be too far behind, i.e., is k steps behind with probability at most $O(\delta)^k$ is perhaps the most challenging technical part of our paper. We discuss some of the ideas behind proving this result.

The first step towards proving this lemma is to attempt to prove an even stronger statement: that even if the current estimates $b_0, \dots, b_m$ are totally arbitrary (perhaps not similar to $a_0, \dots, a_m$ at all), we will still not be behind. This is not too far-fetched, as for general $b_0, \dots, b_m$ we might actually start jumping ahead. For example, if b_0 is not close to a_0 and we do not delete the first 1, we will predict the first 1 in the trace to have come from a later location. This will end up being proven by induction on the length m of the string.

Now, condition on the $(m+1)^{\text{th}}$ 1 from the true string x not being deleted, and consider the probability of being k steps behind after seeing this bit. Recall that the true gaps between the bits until the $(m+1)^{\text{th}}$ 1 are $a_0, \ldots, a_m$ but the algorithm believes the gaps are $b_0, \ldots, b_m$, and the algorithm believes we have just gone through the gaps $b_0, \ldots, b_{m-k}$ so far. Let h be the smallest value where $a_h \approx b_h$ doesn't hold (here, think of $\approx$ as $|a_h - b_h|$ being much larger than $\sqrt{a_h}$, so it would be easy to distinguish between these gaps even with random deletions). Let h' be the smallest value where $b_{m-k-h'} \approx a_{m-h'}$ doesn't hold. This can be thought of in terms of reading the sequences a and b backward, from where the algorithm thinks the gaps are from the sequence b whereas the gaps actually come from a. The idea is that if we are aligned after the hth 1 (which is after the gaps a_{h-1} and b_{h-1}, the difference between a_h and b_h should cause us to move ahead, meaning that we will have to fall $k+1$ steps behind afterwards, making the inductive argument easier for us. By a symmetric argument, we shouldn't expect to have the h'th to last 1 aligned with the h' to last 1 in b. So, the point is that we should expect to fall behind both in the first h gaps and the last h' gaps. This will allow us to split the string into pieces where in each one we fall behind, and we can apply an inductive hypothesis on the length of the string. Another option is that there is never a value h where $a_h \not\approx b_h$ or h' where $a_{m-k-h'} \not\approx a_{m-h'}$. In this case, $(a_0, \ldots, a_m)$ is approximately periodic with period k and we would have to fall an entire period behind, which we show happens with very low probability

Reconstructing x using Algorithm 1

Using Algorithm 1 we can iteratively get estimates $b_0, \ldots, b_t$ with $|b_i - a_i| = O(\sqrt{a_i \log n})$. Namely, suppose that we have the estimates $b_0, \ldots, b_m$. We then run Algorithm 1 on $O(\log n)$ independent traces and with high probability, for a $1 - O(\delta)$ fraction of them, we have that the m'th and $(m+1)$'st one of x are retained in $\tilde{x}$ and correctly aligned. In particular, with probability $1 - O(\delta)$ we can identify both the m'th and $(m+1)$'st one of x in $\tilde{x}$ and taking the median over the gaps between these (and appropriately rescaling by $\frac{1}{1-\delta}$), we obtain an estimate of b_{m+1} such that $|b_{m+1} - a_{m+1}| = O(\sqrt{a_{m+1} \log n}))$. Note that the success probability of $1 - O(\delta)$ is enough to obtain the coarse estimates using the median approach but we cannot obtain a fine estimate by taking the average since with constant probability $O(\delta)$, we may have misaligned the gap completely and then our estimate can be arbitrarily off.

To obtain fine estimates, we first obtain coarse estimates, say $b_0, \ldots, b_t$, for all of the gaps. Next, we show that we can identify the m'th and $(m+1)$'st one in x in a trace $\tilde{x}$ (if they are retained) and we can detect if they were deleted not just with probability $1 - O(\delta)$ but with very high probability. The trick here is to run Algorithm 1 both from the left and from the right on $\tilde{x}$ looking for respectively the one in $\tilde{x}$ aligned to the m'th one in x and the one in $\tilde{x}$ aligned to the $(m+1)$'st one in x (which is the $(t-m)$'th one when running the algorithm from the right). If either of these runs fails to align a one in $\tilde{x}$ to respectively the m'th and $(m+1)$'st one in x or the runs disagree on their alignment, then we will almost certainly know. To see why, assuming that we are never ahead in the alignment procedure from the left, if we believe we have reached the m'th one in x, then we are truly at some m_0'th one where $m_0 \geq m$. By a symmetric argument, if we believe we have reached the $(m+1)$'st one in x after running the procedure from the right, we are truly at the m_1'th one in x, where $m_1 \leq m+1$. The key observation now is that $m_0 \leq m_1$ *if and only if* $m_0 = m$ and $m_1 = m+1$, meaning that both runs succeeding is equivalent to the one found in the left-alignment procedure being strictly earlier than the one found in the right-alignment procedure. So, if we realize that either run fails to align the ones properly, we discard the trace and repeat on a newly sampled trace.

Finally, we can ensure that the success of the runs of the alignment algorithm is independent of the deletion of zeros between the m'th and $(m+1)$'st ones in x. If a trace is not discarded, then with very high probability, the gap between the ones in $\tilde{x}$ aligned to the m'th and $(m+1)$'st ones in x (normalized by $\frac{1}{1-\delta}$) is an unbiased estimator for a_{m+1}. By taking the average of the gap over $\tilde{O}(n)$ traces, normalizing by $\frac{1}{1-\delta}$, and rounding to the nearest integer, we determine a_{m+1} exactly with very high probability. Doing so for each m, reconstructs x.

Roadmap of our paper

In Section 2, we introduce notation. In Section 3, we describe and analyse our main alignment procedure. We first prove that with high probability it is never ahead (Lemma 2). Second, in Section 3.2, we bound the probability that it is behind (Lemma 3). Finally, in Section 4, we describe our full trace reconstruction algorithm and prove Theorem 1.

2 Notation

We note a few notational conventions and definitions.

- We recall that a bitstring x is *L-separated* if the gap between any consecutive 1's in the string contains at least L 0's.
- Given an string x, we say that a *run* is a contiguous sequence of 0's in x. For $x = \underbrace{0\dots0}_{a_0 \text{ times}}\,1\,\underbrace{0\dots0}_{a_1 \text{ times}}\,1\cdots1\,\underbrace{0\dots0}_{a_t \text{ times}}$, the ith run of x is the sequence $\underbrace{0\dots0}_{a_i \text{ times}}$, and has length a_i.
- For any bitstring $x = x_1x_2\cdots x_n$, we use $\text{rev}(x) := x_nx_{n-1}\cdots x_1$ to denote the string where the bits have been reversed.
- We use $\mathbf{a} = a_0, a_1, \dots, a_{m-1}$ to denote an integer sequence of length m. For notational convenience, for any $0 \le j < j' \le m$, we write $\mathbf{a}_{j:j'}$ to denote the subsequence $a_j, a_{j+1}, \dots, a_{j'-1}$, and $a_{j:j'} := \sum_{i=j}^{j'-1} a_i$.

We will define some sufficiently large constants C_0, C_1, C_2, C_3 and a small constant c_0. We will assume the separation parameter $L = C_3 \cdot \log^8 n$, and the deletion parameter $\delta \le c_0$, where $c_0 = \frac{1}{3\cdot 10^6}$. We did not make significant effort to optimize the constant c_0 or the value 8 in $\log^8 n$, though we believe that any straightforward modifications to our analysis will not obtain bounds such as $c_0 \ge \frac{1}{2}$ or a separation of $L = O(\log n)$.

3 Main Alignment Procedure

3.1 Description and Main Lemmas

In this section, we consider a probabilistic process that models a simpler version of the trace reconstruction problem that we aim to solve. In the simpler version of the trace reconstruction problem, suppose that we never delete any 0's, but delete each 1 independently with δ probability. Let $a_0, \dots, a_{m-1} \le n$ represent the true lengths of the first m gaps (so the first 1 is at position $1 + a_0$, the second 1 is at position $2 + a_0 + a_1$, and so on). Moreover, suppose we have some current predictions $b_0, \dots, b_{m-1} \le n$ of the gaps $a_0, \dots, a_{m-1}$. The high level goal will be, given a single trace (where the trace means only 1s are deleted), to identify the mth 1 in the trace from the the original string with reasonably high probability. (Note that the mth 1 is deleted with δ probability, in which case we cannot succeed.)

In this section, we will describe and analyze the probabilistic process, and then explain how this analysis helps us solve the trace reconstruction problem in Section 4.

In the process, we fix $m \le n$ and two sequences $\mathbf{a} = a_0, \dots, a_{m-1}$ and $\mathbf{b} = b_0, \dots, b_{m'-1}$ where $\mathbf{a}$ has length m but $\mathbf{b}$ has some length m' which may or may not equal m. Moreover, we assume $L \le a_i \le n$ and $L \le b_j \le n$ for every term $a_i \in \mathbf{a}$ and $b_j \in \mathbf{b}$.

Now, for each $1 \le i \le m-1$, let $w_i \in \{0, 1\}$ be i.i.d. random variables, with $w_i = 1$ with $1 - \delta$ probability and $w_i = 0$ with δ probability. Also, let $w_0 = w_m = 1$ with probability 1. For each $0 \le i \le m$ with $w_i = 1$, we define a value f_i as follows. First, we set $f_0 = 0$. Next, for each index $i \ge 1$ such that $w_i = 1$, let i_0 denote the previous index with $w_{i_0} = 1$. We define f_i to be the smallest index $j' > f_{i_0}$ such that there exists $f_{i_0} \le j < j'$ with $|b_{j:j'} - a_{i_0:i}| \le C_0 \cdot \log n \cdot \sqrt{b_{j:j'}}$, where C_0 is a sufficiently large constant. (If such an index does not exist, we set $f_i = \infty$.)

Our goal will be for $f_m = m$. In general, for any i with $w_i = 1$, we would like $f_i = i$. If $f_i < i$, we say that we are $i - f_i$ steps behind at step i, and if $f_i > i$, we say that we are $f_i - i$ steps ahead at step i.

First, we note the following lemma, which states that we will never be ahead with very high probability, as long as the sequences $\mathbf{a}$ and $\mathbf{b}$ are similar enough.

▶ **Lemma 2.** *Set $C_1 = C_0/4$. Let $\mathbf{a}, \mathbf{b}$ be sequences of lengths m, m', respectively, where $m' \ge m$. Suppose that $|b_i - a_i| \le C_1 \cdot \sqrt{b_i \log n}$ for all $0 \le i < m$. Then, with probability at least $1 - \frac{1}{n^{10}}$ (over the randomness of the w_i), for all $0 \le i \le m$ with $w_i = 1$, $f_i \le i$.*

Proof. Let us consider the event that for every index $0 \le i \le m - 15 \log n$, at least one of $w_i, w_{i+1}, \dots, w_{i+15 \log n}$ equals 1. Equivalently, the string $w_0 w_1 \cdots w_m$ does not ever have $15 \log n + 1$ 0's in a row. For any fixed i, the probability of this being false is at most $\delta^{15 \log n} \le n^{-15}$, so by a union bound over all choices of i, the event holds with at most n^{-10} failure probability.

First, note that $f_0 = 0$. Now, suppose that some $i \ge 0$ satisfies $w_i = 1$ and $f_i \le i$. Suppose i' is the smallest index strictly larger than i such that $w_{i'} = 1$. Note that $i' - i \le 15 \log n + 1 \le 16 \log n$, by our assumed event. Note that if we set $j = i$ and $j' = i'$, then $j' > j \ge f_i$, since $f_i \le i$. Moreover, $|b_{j:j'} - a_{j:j'}| \le \sum_{i=j}^{j'-1} |b_i - a_i| \le C_1 \sqrt{\log n} \cdot \sum_{i=j}^{j'-1} \sqrt{b_i} \le C_1 \cdot \sqrt{\log n} \cdot \sqrt{b_{j:j'} \cdot |j' - j|} \le 4C_1 \cdot \log n \cdot \sqrt{b_{j:j'}}$, where the second to last inequality is by Cauchy-Schwarz. Thus, $j = i, j' = i'$ satisfies the requirements for $f_{i'}$, which means that $f_{i'} \le j' = i'$. Thus, if $f_i \le i$, $f_{i'} \le i'$. Since $f_0 \le 0$, this means $f_i \le i$ for all i with $w_i = 1$. ◀

The main technical result will be showing that $f_m \ge m$ with reasonably high probability, i.e., with reasonably high probability we are not behind. This result will hold for *any* choice of $\mathbf{a}, \mathbf{b}$ and does not require any similarity between these sequences. In other words, our goal is to prove the following lemma.

▶ **Lemma 3.** *Let $\mathbf{a}, \mathbf{b}$ be strings of length at most n with every $\mathbf{a}_i, \mathbf{b}_j$ between L and n, where $L = C \cdot \log^8 n$ for a sufficiently large constant C. Define $m = |\mathbf{a}|$. Then, for any $\delta \le \frac{1}{3 \cdot 10^6}$, with probability at least $1 - 200 \cdot \delta$ over the randomness of $w_1, \dots, w_{m-1}$, $f_m \ge m$.*

3.2 Proof of Lemma 3

In this section, we prove Lemma 3.

We will set a parameter $K = C_2 \log n$, where C_2 is a sufficiently large constant. For any $k \ge 0$, given the sequences $\mathbf{a} = a_0, \dots, a_{m-1}$ and $\mathbf{b} = b_0, \dots, b_{m'-1}$ (of possibly differing lengths), we define $p_k(\mathbf{a}, \mathbf{b})$ to be the probability (over the randomness of $w_1, \dots, w_{m-1}$) that

- $f_m \le m - k$.
- For any indices $0 \le i \le i' \le m$ with $w_i, w_{i'} = 1$, $f_{i'} - f_i \ge (i' - i) - K$.

Equivalently, this is the same as the probability that we fall behind at least k steps from step 0 to step m, but we never fall behind $K + 1$ or more steps (relatively) from any (possibly intermediate) steps i to i'. For any $m \ge 1$, we define $p_k(m)$ to be the supremum value of $p_k(\mathbf{a}, \mathbf{b})$ over any sequences $\mathbf{a}, \mathbf{b}$ where $\mathbf{a}$ has length at most m and every a_i and b_j is between L and n, and we also define $p_k := \sup_{m \ge 1} p_k(m)$.

Note that for any $k > K$, $p_k(\mathbf{a}, \mathbf{b}) = 0$, as $f_m = m - k$ means $f_m - f_0 < (m - 0) - K$. So, $p_k(m)$ and p_k also equal 0 for any $k > K$.

First, we note a simple proposition, that will only be useful for simplifying the argument at certain places.

▶ **Proposition 4.** *For any $m \ge 1$, $p_0(m) = 1$.*

Proof. Since $p_0(m)$ is the maximum over all $\mathbf{a}, \mathbf{b}$ where $\mathbf{a}$ has length at most m, it suffices to prove it for some $\mathbf{a}, \mathbf{b}$ of length 1. Indeed, for $m = 1$ and $a_0 = b_0 = L$, we must have that $w_0 = w_1 = 1$, so we must have $f_0 = 0$ and $f_1 = 1$. ◀

We now aim to bound the probabilities p_k for $k \le K$. We will do this via an inductive approach on the length of m, where the high-level idea is that if we fall back by k steps, there is a natural splitting point where we can say first we fell back by k_1 steps, and then by k_2 steps, for some $k_1, k_2 > 0$ with $k_1 + k_2 = k$ – see Lemmas 6 and 7. This natural splitting point will be based on the structure of the similarity of $\mathbf{a}$ and $\mathbf{b}$, and will not work if $\mathbf{a}$ and $\mathbf{b}$ share a k-periodic structure. But in the periodic case, we can give a more direct argument that we cannot fall back by k steps (i.e., a full period), even with $\frac{1}{\text{poly}(n)}$ probability – see Lemma 5. We can then compute a recursive formula for the probability of falling back k steps, by saying we need to first fall back k_1 steps and then fall back k_2 steps. In Lemma 9, we bound the terms of this recursion.

▶ **Lemma 5.** *Fix any $m \ge k \ge 1$ such that $k \le K$, and suppose that $L \ge C_3 \cdot \log^8 n$, where C_3 is a sufficiently large multiple of $C_0^2 \cdot C_2^6$. Suppose that $\mathbf{a}, \mathbf{b}$ are sequences such that for every $0 \le i < m - k$, $|b_i - a_i| \le C_0 \log n \cdot \sqrt{b_i}$ and $|b_i - a_{i+k}| \le C_0 \log n \cdot \sqrt{b_i}$. Then, the probability $p_k(\mathbf{a}, \mathbf{b}) \le (2\delta)^K$.*

Proof. We show that the probability of ever being behind by k or more is at most $(2\delta)^K$. In fact, we will show this deterministically never happens, conditioned on the event that for every index $0 \le i \le m - K \cdot k$, at least one of $w_i, w_{i+k}, w_{i+2k}, \dots, w_{i+K \cdot k}$ equals 1. Indeed, the probability of this being false for any fixed i is at most δ^K, so by a union bound over all choices of i, the event holds with at most $n \cdot \delta^K \le (2\delta)^K$ failure probability.

Now, assume the event and suppose that $f_i \le i - k$ holds for some i. More precisely, we fix i to be the smallest index such that $w_i = 1$ and $f_i \le i - k$.

First, assume that $i \ge 2K \cdot k$. Consider the values $a_{i-1}, a_{i-2}, \dots, a_{i-k}$, and let $h = \arg\max_{1 \le t \le k} a_{i-t}$. By our conditional assumption, and since $i \ge 2K \cdot k$, at least one of $w_{i-h}, w_{i-h-k}, \dots, w_{i-h-K \cdot k}$ equals 1. Say that $w_{i-h-r \cdot k} = 1$, where $0 \le r \le K$. Also, by our choice of i, we know that $f_{i-h-r \cdot k} > i - h - (r+1) \cdot k \ge 0$, and that $f_i \le i - k$. So, we have two options:

1. $i \ge 2K \cdot k$, and $f_i \le i - k$, $f_{i-h-r \cdot k} > i - h - (r+1) \cdot k \ge 0$, for some $r \le K$ and where $h = \arg\max_{1 \le t \le k} a_{i-t}$.
2. $i < 2K \cdot k$, and $f_i \le i - k$, $f_0 = 0$.

Now, let's consider the list of all indices $i_0 < i_1 < \cdots < i_s = i$ with $w_{i_0}, w_{i_1}, \dots, w_{i_s} = 1$, starting with $i_0 = i - h - r \cdot k$ if $i \ge 2K \cdot k$ and $i_0 = 0$ otherwise, and ending with $i_s = i$. By definition of the sequence f, for every $0 \le t < s$ there exists j, j' such that $f_{i_t} \le j < j' \le f_{i_{t+1}}$ and $|b_{j:j'} - a_{i_t:i_{t+1}}| \le C_0 \log n \cdot \sqrt{b_{j:j'}}$. Assuming that $L \ge (10 C_0 \log n)^2$, then $a_{i_t:i_{t+1}} \ge (10 C_0 \log n)^2$, which means $a_{i_t:i_{t+1}} \ge b_{j:j'}/2$, and thus $|b_{j:j'} - a_{i_t:i_{t+1}}| \le 2C_0 \log n \cdot \sqrt{a_{i_t:i_{t+1}}}$. So,

$$b_{f_{i_t}:f_{i_{t+1}}} \ge b_{j:j'} \ge a_{i_t:i_{t+1}} - 2C_0 \log n \sqrt{a_{i_t} : a_{i_{t+1}}}.$$

Adding the above equation over $0 \le t \le s-1$, we obtain

$$b_{f_{i_0}:f_i} \ge a_{i_0:i} - 2C_0 \log n \cdot \sum_{t=0}^{s-1} \sqrt{a_{i_t:i_{t+1}}} \ge a_{i_0:i} - 2C_0 \log n \cdot \sqrt{a_{i_0:i} \cdot s},$$

where the final line follows by Cauchy-Schwarz. Let j_0 be $i - h - (r+1) \cdot k + 1$ if $i \ge 2K \cdot k$ and $j_0 = 0$ otherwise. Then, since $s \le i_s - i_0 \le 2k \cdot K \le 4K^2$, we have

$$b_{j_0:i-k} \ge b_{f_{i_0}:f_i} \ge a_{i_0:i} - 4C_0 \cdot K \log n \cdot \sqrt{a_{i_0:i}}. \tag{1}$$

The above equation tells us that $b_{j_0:i-k} = \sum_{t=j_0}^{i-k-1} b_t$ can't be too much smaller than $a_{i_0:i} = \sum_{t=i_0}^{i-1} a_t$. We now show contrary evidence, thus establishing a contradiction.

First, we compare $b_{j_0:i-k}$ to $a_{j_0+k:i}$. Indeed, for any $t < i \le m$, $|b_{t-k} - a_t| \le C_0 \log n \cdot \sqrt{b_{t-k}}$. Since every $a_i \ge (10 C_0 \log n)^2$, this also means $|b_{t-k} - a_t| \le 2C_0 \log n \cdot \sqrt{a_t}$. Adding over all $j_0 \le t < i - k$, we have

$$a_{j_0+k:i} \ge b_{j_0:i-k} - 2C_0 \log n \cdot \sum_{t=j_0+k}^{i-1} \sqrt{a_t} \ge b_{j_0:i-k} - 4C_0 \cdot K \log n \cdot \sqrt{a_{j_0+k:i}},$$

where the last inequality follows by Cauchy-Schwarz and the fact that $i - (j_0 + k) \le i - j_0 \le 2K \cdot k \le 4K^2$.

However, we do not care about $a_{j_0+k:i}$ – we really care about $a_{i_0:i}$. To bound this, first note that for any $k \le i < m$, $|a_i - b_{i-k}| \le C_0 \log n \cdot \sqrt{b_i}$ and $|a_{i-k} - a_{i-k}| \le C_0 \log n \cdot \sqrt{b_i}$. So, $|a_i - a_{i-k}| \le 4C_0 \log n \cdot \sqrt{a_i}$, assuming every $a_i \ge (10 C_0 \log n)^2$. If we additionally have that $L \ge (100 C_0 \log n \cdot K)^2$, then $|a_i - a_{i-s \cdot k}| \le 8C_0 \log n \cdot s \cdot \sqrt{a_i}$ for any $1 \le s \le K$ and $s \cdot k \le i < m$. Importantly, $\frac{a_{i-s\cdot k}}{a_i} \in [1/2, 2]$.

In the case that $i \ge 2K \cdot k$, this implies that $\sum_{t=i-h-r\cdot k}^{i-1} a_t \le 2(r+1) \cdot \sum_{t=i-k}^{i-1} a_t \le 4K \cdot \sum_{t=i-k}^{i-1} a_t$. So, because $h = \arg\max_{1 \le t \le k} a_{i-t}$, we have

$$a_{i_0} = a_{i-h-r\cdot k} \ge \frac{1}{2} \cdot a_{i-h} \ge \frac{1}{2k} \cdot \sum_{t=1}^{k} a_{i-t} \ge \frac{1}{8K^2} \cdot \sum_{t=i-h-r\cdot k}^{i-1} a_t.$$

Recalling that $i_0 = i - h - r \cdot k$ and $j_0 = i - h - (r+1) \cdot k + 1$, since $i_0 = j_0 + k - 1$,

$$\begin{aligned} a_{i_0:i} \ge \left(1 + \frac{1}{8K^2}\right) \cdot a_{j_0+k:i} &\ge \left(1 + \frac{1}{8K^2}\right) \cdot (b_{j_0:i-k} - 4C_0 \cdot K \log n \cdot \sqrt{a_{j_0+k:i}}) \\ &\ge \left(1 + \frac{1}{8K^2}\right) \cdot (b_{j_0:i-k} - 4C_0 \cdot K \log n \cdot \sqrt{a_{i_0:i}}). \end{aligned} \tag{2}$$

In the case that $i < 2K \cdot k$, we instead have $\sum_{t=0}^{i-1} a_t \le 2 \cdot \left\lceil \frac{i}{k} \right\rceil \cdot \sum_{t=0}^{k-1} a_t \le 4K \cdot \sum_{t=0}^{k-1} a_t$. So, since $i_0 = j_0 = 0$, we have that

$$a_{i_0:i} = a_{j_0+k:i} + a_{0:k} \ge \left(1 + \frac{1}{4K}\right) \cdot a_{j_0+k:i} \ge \left(1 + \frac{1}{4K}\right) \cdot (b_{j_0:i-k} - 4C_0 \cdot K \log n \cdot \sqrt{a_{j_0+k:i}}),$$

so the same bound as (2) holds (in fact, an even stronger bound holds).

So, both (1) and (2) hold, in either case. Together, they imply that

$$a_{i_0:i} \ge \left(1 + \frac{1}{8K^2}\right) \cdot \left(a_{i_0:i} - 8C_0 \cdot K \log n \cdot \sqrt{a_{i_0:i}}\right).$$

This is impossible if $a_{i_0:i}$ is a sufficiently large multiple of $(C_0 \cdot K \log n \cdot K^2)^2 = C_0^2 \cdot \log^2 n \cdot K^6$. Since $i \ge i_0 + 1$ in either case, it suffices for L to be a sufficiently large multiple of $C_0^2 \cdot \log^2 n \cdot K^6 = C_0^2 C_2^6 \cdot \log^8 n$. ◀

▶ **Lemma 6.** *Fix any $m \ge k$ such that $k \le K$, and suppose that $L \ge C_3 \cdot \log^2 n \cdot K^6$. Suppose that $\mathbf{a}, \mathbf{b}$ are sequences of length m, such that for every $0 \le i < m - k$, $|b_i - a_{i+k}| \le C_0 \log n \cdot \sqrt{b_i}$. Then, the probability*

$$p_k(\mathbf{a}, \mathbf{b}) \le (2\delta)^K + \sum_{\substack{h_1,h_2,k_2,k_2 \ge 0 \\ h_1+h_2+k_1+k_2 \ge k \\ k_1,k_2 \le K \\ (h_1,h_2,k_1,k_2) \ne (0,0,0,k),(0,0,k,0)}} \delta^{h_1+h_2} p_{k_1}(m-1) p_{k_2}(m-1).$$

Proof. Suppose that for all $0 \le i < m-k$, $|b_i - a_i| \le C_0 \log n \sqrt{b_i}$. Then, we can use Lemma 5 to bound $p_k(\mathbf{a}, \mathbf{b}) \le (2\delta)^K$. Alternatively, let $0 \le h < m - k$ be the smallest index such that $|b_h - a_h| > C_0 \log n \cdot \sqrt{b_h}$. Next, let $h_1, h_2 \ge 0$ be such that $h - h_1$ is the largest index less than h with $w_{h-h_1} = 1$, and $h+1+h_2$ is the smallest index at least $h+1$ with $w_{h+1+h_2} = 1$. Finally, let $k_1 := \max(0, (h - h_1) - f_{h-h_1})$ and $k_2 := \max(0, (m - (h + 1 + h_2)) - (f_m - f_{h+1+h_2}))$. In other words, k_1 is the number of steps we fall behind from 0 to $h - h_1$, and k_2 is the number of steps we fall behind from $h + 1 + h_2$ to m.

Note that $k_1 + k_2 \ge m - 1 - h_1 - h_2 - f_m + f_{h+1+h_2} - f_{h-h_1}$, and since each subsequent f_i is strictly increasing, this means $f_{h+1+h_2} - f_{h-h_1} \ge 1$, so $k_1 + k_2 \ge m - f_m - (h_1 + h_2) \ge k - (h_1 + h_2)$, assuming that $f_m \le m - k$. In other words, we have that h_1, h_2, k_1, k_2 are nonnegative integers such that $h_1 + h_2 + k_1 + k_2 \ge k$.

Now, let us bound the probability (over the randomness of $w_1, \dots, w_{m-1}$) of the event indicated by $p_k(\mathbf{a}, \mathbf{b})$ occurring, with the corresponding values h_1, h_2, k_1, k_2. Note that for any fixed h_1, h_2, the event of those specific values is equivalent to w_{h-h_1} and w_{h+1+h_2} being 1, and everything in between being 0. So, the probability is at most $\delta^{h_1+h_2}$. Now, conditioned on h_1, h_2, the values k_1, k_2 imply that we fall back k_1 steps from step 0 to $h - h_1$ (or we may move forward if $k_1 = 0$) and we fall back k_2 steps from step $h + 1 + h_2$ to m. Moreover, there cannot be two steps i, i' such that that we fell back K steps from i to i'. Since $h - h_1 \le h < m$ and $m - (h+1+h_2) \le m-1$, this means both $h - h_1, m - (h+1+h_2) \le m-1$. So, the overall probability of the corresponding values h_1, h_2, k_1, k_2 is at most $\delta^{h_1+h_2} \cdot p_{k_1}(m-1) \cdot p_{k_2}(m-1)$, where we are using the fact that $p_0(m) = 1$ for all m by Proposition 4.

Overall, the probability $p_k(\mathbf{a}, \mathbf{b})$ is at most

$$\sum_{\substack{h_1,h_2,k_1,k_2 \ge 0 \\ h_1+h_2+k_1+k_2 \ge k \\ k_1,k_2 \le K}} \delta^{h_1+h_2} \cdot p_{k_1}(m-1) p_{k_2}(m-1).$$

We can cap k_1, k_2 as at most K since otherwise $p_{k_1}(m-1)$ or $p_{k_2}(m-1)$ is 0. Moreover, we can give improved bounds in the cases when $h_1 = h_2 = 0$ and either $(k_1, k_2) = (0, k)$ or $(k_1, k_2) = (k, 0)$.

Note that in either case, both w_h and w_{h+1} equal 1. In the former case, we must have $f_h = h - k$ and $f_{h+1} = h + 1 - k$. Importantly, the algorithm fell back by exactly k steps from 0 to h, However, we know that for all $0 \le i \le h - 1$, $|b_i - a_i| \le C_0 \log n \cdot \sqrt{b_i}$. In that case, if we restrict ourselves to the strings $\mathbf{a}_{0:h} = a_0 a_1 \cdots a_{h-1}$ and $\mathbf{b}_{0:h} = b_0 b_1 \cdots b_{h-1}$, we are dealing with the case of Lemma 5. Hence, we can bound the overall probability of this case by $(2\delta)^K$. In the latter case, we must have $f_h = h$ and $f_{h+1} = h+1$, since we need to fall back by exactly k steps from h to m. However, this actually cannot happen, because by definition of f_h and f_{h-1}, we must have that $|b_h - a_h| \le C_0 \log n \cdot \sqrt{b_h}$, which is not true by our definition of h.

Overall, this means

$$p_k(\mathbf{a}, \mathbf{b}) \le (2\delta)^K + \sum_{\substack{h_1,h_2,k_2,k_2 \ge 0 \\ h_1+h_2+k_1+k_2 \ge k \\ k_1,k_2 \le K \\ (h_1,h_2,k_1,k_2) \ne (0,0,0,k),(0,0,k,0)}} \delta^{h_1+h_2} p_{k_1}(m-1) p_{k_2}(m-1). \blacktriangleleft$$

▶ **Lemma 7.** *Fix any $m \ge k$ such that $k \le K$, and suppose that $L \ge C_3 \cdot \log^2 n \cdot K^6$. Suppose that $\mathbf{a}, \mathbf{b}$ are sequences of length m. Then, the probability*

$$p_k(\mathbf{a}, \mathbf{b}) \le (2\delta)^K + \sum_{\substack{h_1,h_2,k_2,k_2 \ge 0 \\ h_1+h_2+k_1+k_2 \ge k \\ k_1,k_2 \le K \\ (h_1,h_2,k_1,k_2) \ne (0,0,0,k),(0,0,k,0)}} \delta^{h_1+h_2} p_{k_1}(m-1) p_{k_2}(m-1).$$

Proof. Our proof will be quite similar to that of Lemma 6, so we omit some of the identical details.

First, assume that for every $k \le i < m$, $|b_{i-k} - a_i| \le C_0 \log n \cdot \sqrt{b_{i-k}}$. Then, we can directly apply Lemma 6. Alternatively, let $k \le h < m$ be the largest index such that $|b_{h-k} - a_h| > C_0 \log n \cdot \sqrt{b_{h-k}}$. As in the proof of Lemma 6, let $h_1, h_2 \ge 0$ be such that $h - h_1$ is the largest index less than h with $w_{h-h_1} = 1$, and $h+1+h_2$ is the smallest index at least $h+1$ with $w_{h+1+h_2} = 1$. Also, let $k_1 := \max(0, (h-h_1) - f_{h-h_1})$ and $k_2 := \max(0, (m-(h+1+h_2)) - (f_m - f_{h+1+h_2}))$.

As in the proof of Lemma 6, we have $h_1 + h_2 + k_1 + k_2 \ge k$, as long as $f_m \le m - k$. We can again do the same casework on h_1, h_2, k_1, k_2, to obtain

$$\sum_{\substack{h_1,h_2,k_1,k_2 \ge 0 \\ h_1+h_2+k_1+k_2 \ge k \\ k_1,k_2 \le K}} \delta^{h_1+h_2} \cdot p_{k_1}(m-1) p_{k_2}(m-1).$$

Once again, we wish to consider the individual cases of $(h_1, h_2, k_1, k_2) = (0,0,0,k)$ or $(h_1, h_2, k_1, k_2) = (0,0,k,0)$ separately. In either case, $w_h = w_{h+1} = 1$. In the former case, must have $f_h = h$ and $f_{h+1} = h+1$. In this case, from step $h+1$ to m we fall behind k steps. In other words, we can restrict ourselves to the strings $\mathbf{a}_{h+1:m} = a_{h+1} \cdots a_{m-1}$ and $\mathbf{b}_{h+1:m} = b_{h+1} \cdots b_{m-1}$. However, we have now restricted ourselves to strings which satisfy the conditions of Lemma 6, so we can bound the probability in this case as at most

$$(2\delta)^K + \sum_{\substack{h_1,h_2,k_2,k_2 \ge 0 \\ h_1+h_2+k_1+k_2 \ge k \\ k_1,k_2 \le K \\ (h_1,h_2,k_1,k_2) \ne (0,0,0,k),(0,0,k,0)}} \delta^{h_1+h_2} p_{k_1}(m-1) p_{k_2}(m-1).$$

In the latter case, we must have $f_h = h - k$ and $f_{h+1} = h+1-k$. However, this is impossible, because $|a_h - b_{h-k}| > C_0 \log n \cdot \sqrt{b_{h-k}}$, by our definition of h.

Overall, by adding all cases together, we obtain

$$p_k(\mathbf{a}, \mathbf{b}) \le (2\delta)^K + 2 \cdot \sum_{\substack{h_1,h_2,k_2,k_2 \ge 0 \\ h_1+h_2+k_1+k_2 \ge k \\ k_1,k_2 \le K \\ (h_1,h_2,k_1,k_2) \ne (0,0,0,k),(0,0,k,0)}} \delta^{h_1+h_2} p_{k_1}(m-1) p_{k_2}(m-1). \blacktriangleleft$$

Overall, this implies that

$$p_k(m) \le (2\delta)^K + 2 \cdot \sum_{\substack{h_1,h_2,k_2,k_2 \ge 0 \\ h_1+h_2+k_1+k_2 \ge k \\ k_1,k_2 \le K \\ (h_1,h_2,k_1,k_2) \ne (0,0,0,k),(0,0,k,0)}} \delta^{h_1+h_2} p_{k_1}(m-1) p_{k_2}(m-1).$$

We now can universally bound p_k for all $0 \le k \le K$. To do so, we first recall some basic properties of the Catalan numbers.

▶ **Fact 8.** *For $n \ge 0$, the Catalan numbers $\mathfrak{C}_n$* [1] *are defined as $\mathfrak{C}_n = \binom{2n}{n}/(n+1)$. They satisfy the following list of properties.*

1. *$\mathfrak{C}_0 = 1$ and for all $n \ge 0$, $\mathfrak{C}_{n+1} = \sum_{i=0}^{n} \mathfrak{C}_i \mathfrak{C}_{n-i}$.*
2. *For all $n \ge 1$, $2 \le \frac{\mathfrak{C}_{n+1}}{\mathfrak{C}_n} \le 4$.*
3. *For all $n \ge 0$, $\mathfrak{C}_n \le 4^n$.*

▶ **Lemma 9.** *Assume $\delta \le \frac{1}{3 \cdot 100^3}$, and define $\mathfrak{D}_k := 100^{2k-1}\mathfrak{C}_k$ for $k \ge 1$ and $\mathfrak{D}_0 = 1$. Then, for all $0 \le k \le K$, $p_k \le \mathfrak{D}_k \cdot \delta^k$.*

Proof. We prove the statement by induction on m. For $m = 1$, note that $w_0 = w_1 = 1$ with probability 1, so $f_1 \ge 1$. Indeed, either $f_1 = 1$ or $f_1 = \infty$. So, $p_0(m) \le 1$ and $p_k(m) = 0$ for all $k \ge 1$.

Now, suppose that the induction hypothesis holds for $m-1$: we now prove the statement for m. First, note that $p_0(m) = 1 = \mathfrak{D}_0 \cdot \delta^0$. Next, for $k \ge 1$,

$$\begin{aligned} p_k(m) &\le (2\delta)^K + 2 \cdot \sum_{\substack{h_1,h_2,k_2,k_2 \ge 0 \\ h_1+h_2+k_1+k_2 \ge k \\ k_1,k_2 \le K \\ (h_1,h_2,k_1,k_2) \ne (0,0,0,k),(0,0,k,0)}} \delta^{h_1+h_2} \cdot \mathfrak{D}_{k_1}\delta^{k_1} \cdot \mathfrak{D}_{k_2}\delta^{k_2} \\ &\le (2\delta)^k + 2 \cdot \sum_{\substack{h_1,h_2,k_2,k_2 \ge 0 \\ h_1+h_2+k_1+k_2 \ge k \\ (h_1,h_2,k_1,k_2) \ne (0,0,0,k),(0,0,k,0)}} \mathfrak{D}_{k_1}\mathfrak{D}_{k_2} \cdot \delta^{h_1+h_2+k_1+k_2}. \end{aligned} \tag{3}$$

We now bound the summation in the above expression. First, we focus on the terms where one of k_1 or k_2 is 0. If $k_1 = k_2 = 0$, the summation becomes $\sum_{h_1+h_2 \ge k} \delta^{h_1+h_2}$. If we fix $h_3 = h_1 + h_2$, for each $h_3 \ge k$ there are $h_3 + 1$ choices of $h_1 + h_2$, which means the summation is $\sum_{h_3 \ge k} \delta^{h_3}(h_3+1)$. For $\delta \le \frac{1}{3 \cdot 100^3}$, each term is at most half the previous term, so this is at most $2(k+1) \cdot \delta^k$. Next, for $k_1 = 0, k_2 > 0$, if we fix $h_3 = h_1 + h_2$, the summation is $\sum_{h_3+k_2 \ge k, (h_3,k_2) \ne (0,k)} (h_3+1)\mathfrak{D}_{k_2}\delta^{h_3+k_2}$, since there are $h_3 + 1$ choices of $(h_1, h_2) : h_1 + h_2 = h_3$. We have a symmetric summation for $k_1 > 0, k_2 = 0$. Finally, if we focus on the terms with $k_1, k_2 \ge 1$, by writing $h_3 = h_1 + h_2$ and $k_3 = k_1 + k_2$, for any fixed h_3, k_3, the sum of $\mathfrak{D}_{k_1}\mathfrak{D}_{k_2}$ is at most $100^{2k_1+2k_2-2} \cdot \mathfrak{C}_{k_3+1} \le 100^{-3} \cdot \mathfrak{D}_{k_3+1}$, and there are $h_3 + 1$ choices for (h_1, h_2). So, the summation is at most $100^{-3} \cdot \sum_{h_3+k_3 \ge k}(h_3 + 1)\mathfrak{D}_{k_3+1}\delta^{h_3+k_3} \le \frac{4}{100} \cdot \sum_{h_3+k_3 \ge k}(h_3+1)\mathfrak{D}_{k_3}\delta^{h_3+k_3}$, where the last inequality holds because $\frac{\mathfrak{D}_{k_3+1}}{\mathfrak{D}_{k_3}} \le 100^2 \cdot \frac{\mathfrak{C}_{k_3+1}}{\mathfrak{C}_{k_3}} \le 4 \cdot 100^2$.

[1] We use $\mathfrak{C}_n$ rather than the more standard C_n to avoid confusion with the constants $C_0, C_1, \ldots$ we have defined.

Overall, replacing indices accordingly, we can write (3) as at most

$$(2\delta)^k + 2 \cdot \left(2(k+1) \cdot \delta^k + 2 \cdot \sum_{\substack{a,b \ge 0 \\ a+b \ge k \\ (a,b) \ne (0,k)}} (a+1)\mathfrak{D}_b \delta^{a+b} + \frac{4}{100} \cdot \sum_{\substack{a,b \ge 0 \\ a+b \ge k}} (a+1)\mathfrak{D}_b \delta^{a+b} \right)$$

$$\le (2\delta)^k + 2 \cdot \left(2(k+1) \cdot \delta^k + 3 \cdot \sum_{\substack{a,b \ge 0 \\ a+b \ge k \\ (a,b) \ne (0,k)}} (a+1)\mathfrak{D}_b \delta^{a+b} + \frac{4}{100} \mathfrak{D}_k \delta^k \right).$$

We can now focus on the middle summation term. If we first consider all terms with $b = 0$, the sum equals $\sum_{a \ge k}(a+1)\delta^a = (k+1)\delta^k + (k+2)\delta^{k+1} + \cdots \le 2(k+1)\delta^k$, as long as $\delta \le \frac{1}{3 \cdot 100^3}$. For the remaining terms, we fix $d = a + b$ and consider the sum. If $d = k$, the sum equals $\delta^k \cdot (2\mathfrak{D}_{k-1} + 3\mathfrak{D}_{k-2} + \cdots + k\mathfrak{D}_1)$. Since $\mathfrak{D}_{n+1} \ge 100^2 \mathfrak{D}_n$ for all $n \ge 1$, this is at most $\delta^k \cdot 4\mathfrak{D}_{k-1}$. For $d > k$, the sum equals $\delta^d \cdot (\mathfrak{D}_d + 2\mathfrak{D}_{d-1} + \cdots + d\mathfrak{D}_1) \le 2\delta^d \cdot \mathfrak{D}_d$. Since $\mathfrak{D}_d \le 4 \cdot 100^2 \cdot \mathfrak{D}_{d+1}$, as long as $\delta \le \frac{1}{3 \cdot 100^3}$, the terms $2\delta^d \cdot \mathfrak{D}_d$ decrease by a factor greater than 2 each time d increases. So the sum over all $d > k$ is at most $4\delta^{k+1} \cdot \mathfrak{D}_{k+1}$. Overall, the summation in the middle term is at most $2(k+1)\delta^k + 4\mathfrak{D}_{k-1} \cdot \delta^k + 4\mathfrak{D}_{k+1} \cdot \delta^{k+1}$.

Overall, this means (3) is at most

$$2^k \delta^k + 16(k+1) \cdot \delta^k + 24\mathfrak{D}_{k-1} \cdot \delta^k + 24\mathfrak{D}_{k+1} \cdot \delta^{k+1} + \frac{8}{100} \mathfrak{D}_k \delta^k. \tag{4}$$

Now, note that $\frac{\mathfrak{D}_{k-1}}{\mathfrak{D}_k} \le \frac{1}{100}$ for all $k \ge 1$, even for $k = 1$. Moreover, $\frac{\mathfrak{D}_{k+1}}{\mathfrak{D}_k} \le 100^2 \cdot \frac{\mathfrak{C}_{k+1}}{\mathfrak{C}_k} \le 4 \cdot 100^2$. Thus, (4) is at most

$$\delta^k \cdot \left(2^k + 16(k+1) + \frac{32}{100} \cdot \mathfrak{D}_k + 96 \cdot 100^2 \cdot \mathfrak{D}_k \cdot \delta \right)$$

Assuming that $\delta \le \frac{1}{3 \cdot 100^3}$, this is at most $\delta^k \cdot \left(2^k + 16(k+1) + 0.64 \cdot \mathfrak{D}_k\right)$, which can be verified to be at most $\delta^k \cdot \mathfrak{D}_k$ for all $k \ge 1$, by just using the fact that $\mathfrak{D}_k \ge 100^k$ for all $k \ge 1$. This completes the inductive step. ◀

We are now ready to prove Lemma 3.

Proof of Lemma 3. If $f_m < m$, this means that either the event $p_1(\mathbf{a}, \mathbf{b})$ occurs, or there exist indices $i < i'$ with $w_i = w_{i'} = 1$ but we fall behind at least $K + 1$ steps from step i to step i'.

Assuming $\delta \le \frac{1}{3 \cdot 10^3}$, the probability of $p_1(\mathbf{a}, \mathbf{b})$ is at most 100δ. Alternatively, if there exist $i < i'$ with $w_i = w_{i'} = 1$ but we fall behind at least $K + 1$ steps from step i to step i', there must exist such an i, i' with minimal $i' - i$ (breaking ties arbitrarily). This could be because $w_{i+1} = w_{i+2} = \cdots = w_{i+r} = 0$ for some $r \ge K/2$. However, the probability of there being $r \ge K/2$ consecutive indices $w_{i+1} = w_{i+2} = \cdots = w_{i+r} = 0$ is at most $n \cdot \delta^{K/2} \le \delta$.

The final option is that, if we look at the first index $i + r > i$ with $w_{i+r} = 0$, $r \le K/2$. This means that from step $i + r$ to i', we must fall behind at least $K/2$ steps, and there could not have been any intermediate steps where we fell behind more than K steps. Hence, if we restrict ourselves to the strings $\mathbf{a}_{i+r:i'}$ and $\mathbf{b}_{f_{i+r}, f_{i'}}$, the event indicated by $p_k(\mathbf{a}_{i+r:i'}, \mathbf{b}_{f_{i+r}:})$ must occur, since conditioned on f_{i+r} and the fact that $w_{i+r} = w_{i'} = 1$, the value $f_{i'}$ only depends on $\mathbf{a}_{i+r:i'}$, $\mathbf{b}$ starting from position f_{i+r}, and $w_{i+r+1}, \ldots, w_{i'-1}$.

In other words, there exists some contiguous subsequences $\mathbf{a}'$ and $\mathbf{b}'$ of $\mathbf{a}$ and $\mathbf{b}$, respectively, such that the event of $p_{K/2}(\mathbf{a}', \mathbf{b}')$ occurs. For any fixed $\mathbf{a}', \mathbf{b}'$, the probability is at most $(4 \cdot 100^2 \cdot \delta)^{K/2}$. Since there are at most n^2 possible contiguous subsequences for each of $\mathbf{a}'$ and $\mathbf{b}'$, the overall probability is at most $n^4 \cdot (4 \cdot 100^2 \cdot \delta)^{K/2} \le 50\delta$, assuming that $\delta \le \frac{1}{3 \cdot 10^6}$ and $K = C_2 \log n$ where C_2 is sufficiently large.

Overall, the probability of falling behind is at most $100\delta + \delta + 50\delta \le 200\delta$. ◀

4 Full algorithm/analysis

Let us depict the true string $x \in \{0,1\}^n$ as $\underbrace{0\dots0}_{a_0 \text{ times}} 1 \underbrace{0\dots0}_{a_1 \text{ times}} 1 \cdots 1 \underbrace{0\dots0}_{a_t \text{ times}}$, i.e., there are $t-1$ ones, and the string starts and ends with a run of 0's. This assumption can be made WLOG by padding the string with L 0's at the front and the end. For any L-separated string, doing this padding maintains the L-separated property, and we can easily simulate the padded trace by adding $\text{Bin}(L, 1-\delta)$ 0's at the front and $\text{Bin}(L, 1-\delta)$ 0's at the back. Once we reconstruct the padded string, we remove the padding to get x.

We assume we know the value of t. Indeed, the number of 1's in a single trace $\tilde{x}$ is distributed as $\text{Bin}(t, 1-\delta)$. So, by averaging the number of 1's over $O(n \log n)$ random traces and dividing by $1-\delta$, we get an estimate of $t-1$ that is accurate within 0.1 with $1 - \frac{1}{n^{10}}$ probability. Thus, by rounding, we know t exactly with $1 - \frac{1}{n^{10}}$ probability.

The main goal is now to learn the lengths $a_0, a_1, \dots, a_t$. If we learn these exactly just using the traces, this completes the proof. Our algorithm runs in two phases: a coarse estimation phase and a fine estimation phase. In the coarse estimation phase, we sequentially learn each a_i up to error $O(\sqrt{a_i} \log n)$. In the fine estimation phase, we learn each a_i exactly, given the coarse estimates.

4.1 Coarse estimation

Fix some $0 \le m \le t$, and suppose that for all $i < m$, we have estimates b_i satisfying $|b_i - (1-\delta)a_i| \le 10\sqrt{a_i}$. (If $m = 0$, then we have no estimates yet.) Our goal will be to provide an estimate b_m such that $|b_m - (1-\delta)a_m| \le 10\sqrt{a_m}$.

Consider a trace $\tilde{x}$ of x. Let $w_0 = w_{t+1} = 1$ and for each $1 \le i \le t$, let w_i be the indicator that the ith 1 is retained. Next, for each $0 \le i \le t$, let $\tilde{a}_i \sim \text{Bin}(a_i, 1-\delta)$ represent the number of 0s in the ith run that were not deleted. Note that with at least 0.99 probability, $|\tilde{a}_i - (1-\delta)a_i| \le 10\sqrt{\log n \cdot a_i}$ for all i. Since $|b_i - (1-\delta)a_i| \le 10\sqrt{a_i}$ for all $i < m$, this implies that $|\tilde{a}_i - b_i| \le 20\sqrt{\log n \cdot b_i}$ for all $i < m$.

Now, even though we have no knowledge of $\tilde{a}_i$ or a_i, we can still simulate the probabilistic process of Section 3. Let $0 = i_0 < i_1 < \cdots < i_h = t+1$ be the list of all indices $i : 0 \le i \le t+1$ with $w_i = 1$. While we do not know the values $\tilde{a}_i$, for every pair of consecutive indices i_q, i_{q+1}, the value $\tilde{a}_{i_q:i_{q+1}}$ is exactly the number of 0's between the qth and $(q+1)$st 1 in the trace $\tilde{x}$ (where we say that the 0th 1 is at position 0 and the $(t+1)$st 1 is at position $|\tilde{x}|+1$). In other words, if r_q represents the position of the qth 1, then $\tilde{a}_{i_q:i_{q+1}} = r_{q+1} - r_q - 1$. Hence, because computing each $f_{i_{q+1}}$ only requires knowledge of $\mathbf{b}$ and the value of $\tilde{a}_{i_q:i_{q+1}}$, and since $f_{i_0} = f_0 = 0$, the algorithm can in fact compute $g_q := f_{i_q}$ for all $0 \le q \le h$, using the same process as described in Section 3, even if the values i_q are not known.

Algorithm 1 simulates this process, assuming knowledge of m, $b_0, \dots, b_{m-1}$, a single trace $\tilde{x}$, and t. In Algorithm 1, we use the variable val to represent $g_q = f_{i_q}$, i.e., the current prediction of the position i_q. In other words, $\text{val} - i_q$ equals the number of steps ahead (or $i_q - \text{val}$ equals the number of steps behind) we are.

Algorithm 1 Locate the mth and $(m+1)$st 1 in x, in the trace $\tilde{x}$, and return the position and length of the gap.

```
procedure ALIGN($\tilde{x}, t, m, b_0, \dots, b_{m-1}$)
    Let $r_q$ be the position of the $q$th 1 in $\tilde{x}$, for each $1 \le q \le t-1$.
    $r_0 \leftarrow 0$, $r_t \leftarrow |\tilde{x}| + 1$.
    val $\leftarrow 0$, $q \leftarrow 0$
    while val $< m$ do
        Find the smallest $j'$ such that $\exists j, j'$ with val $\le j < j'$ and $|(r_{q+1} - r_q - 1) - b_{j:j'}| \le C_0 \log n \cdot \sqrt{b_{j:j'}}$.
        if no such $j, j'$ exist then
            Return FAIL
        val $\leftarrow j'$
        $q \leftarrow q + 1$
    if val $= m$ then
        Return $(q, r_{q+1} - r_q - 1)$.
    else
        Return FAIL
```

▶ **Lemma 10.** *Fix $b_0, \dots, b_{m-1}$ such that $|b_i - (1-\delta)a_i| \le 10\sqrt{a_i}$ for all $0 \le i \le m-1$. With probability at least 0.98 over the randomness of $\tilde{x}$, we have that Algorithm 1 returns q such that the qth 1 in $\tilde{x}$ corresponds to the mth 1 in x. Moreover, conditioned on this event holding, the distribution $r_{q+1} - r_q - 1$ exactly follows* $\mathrm{Bin}(a_m, 1-\delta)$.

Proof. Let us first condition on the values $\tilde{a}_0, \dots, \tilde{a}_{m-1}$, assuming that $|\tilde{a}_i - (1-\delta)a_i| \le 10\sqrt{\log n \cdot a_i}$ for all $0 \le i \le m-1$. As discussed earlier, this occurs with at least 0.99 probability, and implies that $|\tilde{a}_i - b_i| \le 20\sqrt{\log n \cdot b_i}$ for all $i < m$.

Let us also condition on $w_m = 1$. By Lemma 2 and Lemma 3, the probability that $f_m = m$, for $\delta = \frac{1}{3 \cdot 10^6}$, is at least 0.99. This is conditioned on $w_m = 1$ and the values $\tilde{a}_1, \dots, \tilde{a}_{m-1}$ (assuming $|\tilde{a}_i - b_i| \le 20\sqrt{\log n \cdot b_i}$). This means that with at least 0.99 probability, the algorithm finds the position q with $i_q = m$. Since f_m only depends on $\mathbf{b}$, $\tilde{\mathbf{a}}_{0:m}$ and $w_1, \dots, w_m$, with probability at least $0.99 \cdot (1-\delta) \cdot 0.99$ over the randomness of $w_1, \dots, w_m$ and $\tilde{a}_1, \dots, \tilde{a}_{m-1}$, we have that $w_m = 1$ and $i_q = m$. This is independent of w_{m+1}, so with probability at least $0.99^2 \cdot (1-\delta)^2 \ge 0.98$ probability, we additionally have that $w_{m+1} = 1$.

The event that $i_q = m$ means that r_q is the position in $\tilde{x}$ of the mth 1 in the true string x. Moreover, since neither the mth nor $(m+1)$th 1 was deleted, r_{q+1} is the position in $\tilde{x}$ of the $(m+1)$th 1 in the true string x. So, $r_{q+1} - r_q - 1$ is in fact the length of the gap between the mth and $(m+1)$th 1 after deletion, which means it has length $\tilde{a}_m \sim \mathrm{Bin}(a_m, 1-\delta)$, since $\tilde{a}_m$ is independent of the events that decide whether $w_m = w_{m+1} = 1$ and $i_q = m$. ◀

Given this, we can crudely estimate every gap, in order. Namely, assuming that that we have estimates $b_0, \dots, b_{m-1}$ (where $0 \le m \le t$), we can run the ALIGN procedure on $O(\log n)$ independent traces. By a Chernoff bound, with $\frac{1}{n^{15}}$ failure probability, at least 0.9 fraction of the traces will have the desired property of Lemma 10, so will output some (q, b) where $b \sim \mathrm{Bin}(a_m, 1-\delta)$. Since $\mathrm{Bin}(a_m, 1-\delta)$ is in the range $a_m(1-\delta) \pm 10\sqrt{a_m}$ with at least 0.99 probability, at least 0.75 fraction of the outputs (q, b) will satisfy $|b - (1-\delta)a_m| \le 10\sqrt{a_m}$, with $\frac{1}{n^{15}}$ failure probability. Thus, by defining b_m to be the median value of b across the randomly drawn traces, we have that $|b_m - (1-\delta)a_m| \le 10\sqrt{a_m}$ with at least $1 - \frac{1}{n^{10}}$ probability.

By running this procedure iteratively to provide estimates $b_0, b_1, \ldots, b_t$, we obtain Algorithm 2. The analysis in the above paragraph implies the following result.

▶ **Theorem 11** (Crude Approximation). *Algorithm 2 uses $O(n \log n)$ traces and polynomial time, and learns estimates $b_0, b_1, \ldots, b_t$ such that with at least $1 - \frac{1}{n^9}$ probability, $|b_m - (1-\delta)a_m| \le 10\sqrt{a_m}$ for all $0 \le m \le t$.*

Algorithm 2 Crude Estimation of all gaps.

```
procedure Crude
    Use $O(n \log n)$ traces to compute $t$, where $t$ equals the number of 1s in $x$.
    for $m = 0$ to $t$ do
        for $i = 1$ to $O(\log n)$ do
            Draw trace $\tilde{x}^{(i)}$.
            $(q^{(i)}, b^{(i)}) \leftarrow$ Align$(\tilde{x}^{(i)}, t, m, b_0, \ldots, b_{m-1})$
        Let $b_m$ be the median of $b^{(1)}, \ldots, b^{(O(\log n))}$ ▷ Some of the outputs $(q^{(i)}, b^{(i)})$ may
    be FAIL, but we can let $b^{(i)}$ be an arbitrary real number if Align failed on $\tilde{x}^{(i)}$, so that
    the median is well-defined.
    Return $(b_0, b_1, \ldots, b_t)$
```

4.2 Fine estimation

In this section, we show how to exactly compute each a_m with high probability, given the crude estimates $b_0, b_1, \ldots, b_{t-1}$. This will again be done using an alignment procedure, but this time running the alignment both "forward and backward".

Namely, given a trace $\tilde{x}$, we will try to identify the mth and $(m+1)$st 1's from the original string, but we try to identify the mth 1 by running Align on $\tilde{x}$ and the $(m+1)$st 1 by running Align on the reverse string $\text{rev}(\tilde{x}) := \tilde{x}_{|\tilde{x}|} \cdots \tilde{x}_2 \tilde{x}_1$. The idea is: assuming that we never go ahead in the alignment procedure, if we find some index q in the forward alignment procedure with $g_q = f_{i_q} = m$, then the true position i_q must be at least m. Likewise, if we do the alignment procedure in reverse until we believe we have found the $(t-m)$th 1 from the back (equivalently, the $(m+1)$th 1 from the front), the true position must be at most $m+1$.

So, the true positions of the index found in the forward alignment procedure can only be earlier than that of the index from the backward alignment procedure, if the true positions were exactly m and $m+1$, respectively. Thus, by comparing the indices, we can effectively verify that the positions are correct, with negligible failure probability (rather than with $1 - O(\delta)$ failure probability). This is the key towards obtaining the fine estimate of a_m, rather than just a coarse estimate that may be off by $O(\sqrt{a_m})$.

Algorithm 3 formally describes the fine alignment procedure, using $N = O(n \log n)$ traces, assuming we have already done the coarse estimation to find $b_0, b_1, \ldots, b_t$.

▶ **Lemma 12.** *Suppose that $|b_i - (1-\delta)a_i| \le 10\sqrt{a_i}$ for all $1 \le 0 \le t$. Fix indices $0 \le m \le t$ and $1 \le i \le N$, and for simplicity of notation, let $\tilde{x} := \tilde{x}^{(i)}$. Let $\tilde{m}$ be the number of 1's in $\tilde{x}$. Then, the probability that $q_f + q_b = \tilde{m}$, but either the forward or backward iterations finds an index in $\tilde{x}$ which does not correspond to the mth 1 or $(m+1)$th 1, respectively, from x, is at most $2n^{-10}$. Moreover, if the forward and backward iterations find indices in $\tilde{x}$ corresponding to the mth 1 and $(m+1)$th 1, respectively, then $q_f + q_b = \tilde{m}$. Finally, the probability of finding both corresponding indices is at least 0.98.*

Algorithm 3 Fine Estimation of all gaps.

```
procedure FINE($t, b_0, \dots, b_t$)
    Draw $N = O(n \log n)$ traces $\tilde{x}^{(1)}, \dots, \tilde{x}^{(N)}$.
    for $m = 0$ to $t$ do
        Initialize $b^{(1)}, b^{(2)}, \dots, b^{(N)} \leftarrow$ NULL.
        for $i = 1$ to $N$ do
            $\tilde{m} \leftarrow$ number of 1's in $\tilde{x}$.
            $(q_{\mathrm{f}}, b_{\mathrm{f}}) \leftarrow$ ALIGN($\tilde{x}^{(i)}, t, m, b_0, b_1 \dots, b_t$).
            $(q_{\mathrm{b}}, b_{\mathrm{b}}) \leftarrow$ ALIGN(rev($\tilde{x}^{(i)}$)$, t, t-m, b_t, b_{t-1}, \dots, b_0$).
            if $q_{\mathrm{f}} + q_{\mathrm{b}} = \tilde{m}$ then
                $b^{(i)} \leftarrow b_{\mathrm{f}}$
            Set $a_m$ to be $\frac{1}{1-\delta}$ times the average of all non-null $b^{(i)}$'s, rounded to the nearest
    integer.
    Return $(a_0, a_1, \dots, a_t)$
```

Proof. First, let us consider the forward alignment procedure. We know that val tracks f_{i_q} when looking at the qth 1 of $\tilde{x}$ (from left to right). So, if we do not return **FAIL**, then $f_{i_{q_{\mathrm{f}}}} = m$. If $i_{q_{\mathrm{f}}} < m$, this implies there is an index $i = i_{q_{\mathrm{f}}}$ where $f_i > i$. The probability of this is at most n^{-10}, by Lemma 2. Otherwise, $i_{q_{\mathrm{f}}} \geq m$, meaning that the q_{f}th 1 in $\tilde{x}$ is after (or equal to) the mth 1 in x.

Likewise, if we consider the backward alignment procedure, if we do not return **FAIL**, then except for an event with probability at most n^{-10}, the q_{b}th 1 in rev($\tilde{x}$) is ahead of (or equal to) the $(t-m)$th 1 in rev(x). Equivalently, the $(\tilde{m}+1-q_{\mathrm{b}})$th 1 in $\tilde{x}$ (reading from left to right) is before (or equal to) the $(m+1)$th 1 in x (reading from left to right).

So, barring a $2 \cdot n^{-10}$ probability event, the only way that the q_{f}th 1 in $\tilde{x}$ is strictly before the $(\tilde{m}+1-q_{\mathrm{b}})$th 1 in $\tilde{x}$ is if the q_{f}th 1 in $\tilde{x}$ is precisely the mth 1 in x and $(\tilde{m}+1-q_{\mathrm{b}})$th 1 in $\tilde{x}$ is precisely the $(m+1)$th 1 in x. However, if $q_{\mathrm{f}} + q_{\mathrm{b}} = \tilde{m}$, then in fact the q_{f}th 1 is before the $(\tilde{m}+1-q_{\mathrm{b}})$th 1 in $\tilde{x}$ (reading from left to right). This proves the first statement.

Next, if we in fact found the corresponding indices, they are consecutive 1's in x, which means they must be consecutive 1's in $\tilde{x}$. So, if we found the q_{f}th 1 from the left, and the q_{b}th 1 from the right, we must have $q_{\mathrm{f}} + q_{\mathrm{b}} = \tilde{m}$.

Finally, the event of finding both corresponding indices is equivalent to $f_m = m$ in the forward iteration and $f_{t-m} = t-m$ in the backward iteration. Conditioned on the corresponding 1's *not* being deleted, each of these occur with at least 0.98 probability, by Lemmas 2 and 3. So, the overall probability is at least 0.9. ◀

We are now ready to prove Theorem 1. Indeed, given the accuracy of the crude estimation procedure, it suffices to check that for each m, we compute a_m correctly, with at least $1-n^{-5}$ probability.

▶ **Theorem 13** (Fine Estimation). *Assume that t, the number of ones in x, is computed correctly, and for all $0 \leq m \leq t$, $|b_m - (1-\delta)a_m| \leq 10\sqrt{a_m}$.*

Then, for any fixed $m : 0 \leq m \leq t$, with at least $1-n^{-5}$ probability, we compute the gap a_m correctly.

Proof. For any fixed iteration $i : 1 \leq i \leq N$, if both the forward and backward procedures correctly identify the mth and $(m+1)$th 1's from the left, respectively, then $q_{\mathrm{f}} + q_{\mathrm{b}} = \tilde{m}$ by Lemma 12. In this case, we will compute an actual value $b^{(i)} = b_{\mathrm{f}}$. Moreover, as discussed in the proof of Lemma 10, the event that the forward procedure correctly identifies the right

1 only depends on $\mathbf{b}$, $\hat{a}_0, \dots, \hat{a}_{m-1}$, and the events of whether the first m 1's are deleted. Thus, the event that the backward procedure correctly identifies the right 1 only depends on $\mathbf{b}$, $\hat{a}_{m+1}, \dots, \hat{a}_t$, and the events of whether the $(m+1)$th 1 until the tth 1 are deleted.

Thus, the forward and backward procedure correctly identifying the right 1's is independent of $\hat{a}_m \sim \text{Bin}(a_m, 1-\delta)$. Moreover, in this case, b_f is precisely $\hat{a}_m$, since q_f is the position in $\tilde{x}$ corresponding to the mth 1 in x, and neither the mth nor $(m+1)$th 1 can be deleted if both of these 1's are identified.

So, if the forward and backward procedures identifying the right 1's for trace $\tilde{x}^{(i)}$, the conditional distribution of $b^{(i)}$ is $\text{Bin}(a_m, 1-\delta)$. However, we really want to look at the distribution conditioned on the event $q_\text{f} + q_\text{b} = \tilde{m}$. Indeed, by Lemma 12, this event is equivalent to either the forward and backward procedures identifying the right 1's, or some other event which occurs with at most $2n^{-10}$ probability. Because $b^{(i)}$ is clearly between 0 and n, and since the probability of both 1's being correctly identified is at least 0.9 by Lemma 12, the expectation of $b^{(i)}$, conditioned on not being **NULL**, is $a_m(1-\delta) \pm O(n^{-10} \cdot n) = a_m(1-\delta) \pm O(n^{-9})$.

By a Chernoff bound, the number of $1 \le i \le N$ with $b^{(i)} \neq \textbf{NULL}$ is at least $0.5 \cdot N$ with at least $1 - n^{-10}$ probability, since in expectation it is at least $0.9N$. Then, by another Chernoff bound, the empirical average of all such $b^{(i)}$ is within 0.1 of its expectation with $1 - n^{-10}$ probability, which is $a_m(1-\delta) \pm O(n^{-9})$. Thus, taking the empirical average and dividing by $1-\delta$, with at most $O(n^{-10})$ failure probability, $\frac{1}{1-\delta}$ times the average of all non-null $b^{(i)}$'s is within 0.2 of a_m, and thus rounds to a_m. ◀

5 Conclusion and Open Questions

In this paper, we established that the trace reconstruction problem can be solved with a polynomial number of traces, as long as any two ones in the initial string are separated by at least $\operatorname{polylog} n$ zeros and the deletion probability is at most a sufficiently small constant. It is an interesting open question to handle more general deserts such as $(01)_n = 010101\dots01$ interspersed with mildly separated zeros and ones. Indeed, we believe that this is an important step towards solving the general trace reconstruction problem with deletion probability $\delta = n^{-o(1)}$. With this deletion probability, the Bitwise Majority Alignment (BMA) algorithm from [3] succeeds in reconstructing x as long as x does not contain any such highly repetitive contiguous substrings. If one can provide a separate algorithm for such strings, one could then imagine x being partitioned into contiguous substrings that can be reconstructed by respectively BMA and the highly repetitive algorithm in an alternating fashion. Additional work is required to determine how to switch between the two algorithms.

References

1 Frank Ban, Xi Chen, Adam Freilich, Rocco A. Servedio, and Sandip Sinha. Beyond trace reconstruction: Population recovery from the deletion channel. In *Foundations of Computer Science (FOCS)*, pages 745–768, 2019. doi:10.1109/FOCS.2019.00050.

2 Frank Ban, Xi Chen, Rocco A. Servedio, and Sandip Sinha. Efficient average-case population recovery in the presence of insertions and deletions. In *Approximation, Randomization, and Combinatorial Optimization: Algorithms and Techniques*, pages 44:1–44:18, 2019. doi:10.4230/LIPIcs.APPROX-RANDOM.2019.44.

3 Tugkan Batu, Sampath Kannan, Sanjeev Khanna, and Andrew McGregor. Reconstructing strings from random traces. In *Symposium on Discrete Algorithms (SODA)*, pages 910–918, 2004. URL: http://dl.acm.org/citation.cfm?id=982792.982929.

4 Joshua Brakensiek, Ray Li, and Bruce Spang. Coded trace reconstruction in a constant number of traces. In *Foundations of Computer Science (FOCS)*, 2020. arXiv:1908.03996.

5 Diptarka Chakraborty, Debarati Das, and Robert Krauthgamer. Approximate trace reconstruction via median string (in average-case). In *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. doi:10.4230/LIPICS.FSTTCS.2021.11.

6 Zachary Chase. New lower bounds for trace reconstruction. *Ann. Inst. H. Poincaré Probab. Statist.*, 57(2), 2021. URL: http://arxiv.org/abs/1905.03031.

7 Zachary Chase. Separating words and trace reconstruction. In *Symposium on Theory of Computing (STOC)*, 2021.

8 Zachary Chase and Yuval Peres. Approximate trace reconstruction of random strings from a constant number of traces. *CoRR*, abs/2107.06454, 2021.

9 Xi Chen, Anindya De, Chin Ho Lee, Rocco A. Servedio, and Sandip Sinha. Polynomial-time trace reconstruction in the low deletion rate regime. In *Innovations in Theoretical Computer Science (ITCS)*, 2021. arXiv:2012.02844.

10 Xi Chen, Anindya De, Chin Ho Lee, Rocco A. Servedio, and Sandip Sinha. Polynomial-time trace reconstruction in the smoothed complexity model. In *Symposium on Discrete Algorithms (SODA)*, 2021. arXiv:2008.12386.

11 Xi Chen, Anindya De, Chin Ho Lee, Rocco A. Servedio, and Sandip Sinha. Near-optimal average-case approximate trace reconstruction from few traces. In *Symposium on Discrete Algorithms (SODA)*, 2022. arXiv:2107.11530.

12 Xi Chen, Anindya De, Chin Ho Lee, Rocco A. Servedio, and Sandip Sinha. Approximate trace reconstruction from a single trace. In *Symposium on Discrete Algorithms (SODA)*, 2023. doi:10.48550/arXiv.2211.03292.

13 Mahdi Cheraghchi, Ryan Gabrys, Olgica Milenkovic, and João Ribeiro. Coded trace reconstruction. *IEEE Trans. Inf. Theory*, 66(10):6084–6103, 2020. doi:10.1109/TIT.2020.2996377.

14 Sami Davies, Miklos Racz, and Cyrus Rashtchian. Reconstructing trees from traces. In *Conference On Learning Theory (COLT)*, pages 961–978, 2019. URL: http://proceedings.mlr.press/v99/davies19a.html.

15 Sami Davies, Miklós Z. Rácz, Benjamin G. Schiffer, and Cyrus Rashtchian. Approximate trace reconstruction: Algorithms. In *International Symposium on Information Theory (ISIT)*, pages 2525–2530. IEEE, 2021. doi:10.1109/ISIT45174.2021.9517926.

16 Anindya De, Ryan O'Donnell, and Rocco A. Servedio. Optimal mean-based algorithms for trace reconstruction. *Annals of Applied Probability*, 29(2):851–874, 2019. doi:10.1214/18-AAP1394.

17 Lisa Hartung, Nina Holden, and Yuval Peres. Trace reconstruction with varying deletion probabilities. In *Analytic Algorithmics and Combinatorics (ANALCO)*, pages 54–61, 2018. doi:10.1137/1.9781611975062.6.

18 Nina Holden and Russell Lyons. Lower bounds for trace reconstruction. *Annals of Applied Probability*, 30(2):503–525, 2020. doi:10.1214/19-AAP1506.

19 Nina Holden, Robin Pemantle, and Yuval Peres. Subpolynomial trace reconstruction for random strings and arbitrary deletion probability. In *Conference On Learning Theory (COLT)*, pages 1799–1840, 2018. URL: http://proceedings.mlr.press/v75/holden18a.html.

20 Thomas Holenstein, Michael Mitzenmacher, Rina Panigrahy, and Udi Wieder. Trace reconstruction with constant deletion probability and related results. In *Symposium on Discrete Algorithms (SODA)*, pages 389–398, 2008. doi:10.1145/1347082.1347125.

21 Sampath Kannan and Andrew McGregor. More on reconstructing strings from random traces: insertions and deletions. In *International Symposium on Information Theory (ISIT)*, pages 297–301, 2005. doi:10.1109/ISIT.2005.1523342.

22 Akshay Krishnamurthy, Arya Mazumdar, Andrew McGregor, and Soumyabrata Pal. Trace reconstruction: Generalized and parameterized. *IEEE Trans. Inf. Theory*, 67(6):3233–3250, 2021. doi:10.1109/TIT.2021.3066010.

23 Vladimir I. Levenshtein. Efficient reconstruction of sequences. *IEEE Trans. Information Theory*, 47(1):2–22, 2001. doi:10.1109/18.904499.

24 Vladimir I. Levenshtein. Efficient reconstruction of sequences from their subsequences or supersequences. *J. Comb. Theory, Ser. A*, 93(2):310–332, 2001. doi:10.1006/jcta.2000.3081.

25 Andrew McGregor, Eric Price, and Sofya Vorotnikova. Trace reconstruction revisited. In *European Symposium on Algorithms (ESA)*, pages 689–700, 2014. doi:10.1007/978-3-662-44777-2_57.

26 Andrew McGregor and Rik Sengupta. Graph reconstruction from random subgraphs. In *International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 229, pages 96:1–96:18, 2022. doi:10.4230/LIPICS.ICALP.2022.96.

27 Andrew McGregor and Rik Sengupta. Graph reconstruction from noisy random subgraphs. *CoRR*, abs/2405.04261, 2024. doi:10.48550/arXiv.2405.04261.

28 Shyam Narayanan. Improved algorithms for population recovery from the deletion channel. In *Symposium on Discrete Algorithms (SODA)*, pages 1259–1278. SIAM, 2021. doi:10.1137/1.9781611976465.77.

29 Shyam Narayanan and Michael Ren. Circular trace reconstruction. In *Innovations in Theoretical Computer Science (ITCS)*, 2021. arXiv:2009.01346.

30 Fedor Nazarov and Yuval Peres. Trace reconstruction with $\exp(\mathrm{o}(\mathrm{n}^{1/3}))$ samples. In *Symposium on Theory of Computing (STOC)*, pages 1042–1046, 2017. doi:10.1145/3055399.3055494.

31 Yuval Peres and Alex Zhai. Average-case reconstruction for the deletion channel: Subpolynomially many traces suffice. In *Foundations of Computer Science (FOCS)*, pages 228–239, 2017. doi:10.1109/FOCS.2017.29.

32 Ittai Rubinstein. Average-case to (shifted) worst-case reduction for the trace reconstruction problem. In *International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 261 of *LIPIcs*, pages 102:1–102:20, 2023. URL: https://arxiv.org/abs/2207.11489.

33 Alec Sun and William Yue. The trace reconstruction problem for spider graphs. *Discrete Mathematics*, 346(1):113115, 2023. doi:10.1016/J.DISC.2022.113115.

34 Krishnamurthy Viswanathan and Ram Swaminathan. Improved string reconstruction over insertion-deletion channels. In *Symposium on Discrete Algorithms (SODA)*, pages 399–408, 2008. doi:10.1145/1347082.1347126.