Lower Bounds and Improved Algorithms for Asymmetric Streaming Edit Distance and Longest Common Subsequence

Xin Li
Department of Computer Science, Johns Hopkins University, Baltimore, MD, USA

Yu Zheng
Department of Computer Science, Johns Hopkins University, Baltimore, MD, USA

Abstract

In this paper, we study *edit distance* (ED) and *longest common subsequence* (LCS) in the asymmetric streaming model, introduced by Saks and Seshadhri [26]. As an intermediate model between the random access model and the streaming model, this model allows one to have streaming access to one string and random access to the other string. Meanwhile, ED and LCS are both fundamental problems that are often studied on large strings, thus the (asymmetric) streaming model is ideal for studying these problems.

Our first main contribution is a systematic study of space lower bounds for ED and LCS in the asymmetric streaming model. Previously, there are no explicitly stated results in this context, although some lower bounds about LCS can be inferred from the lower bounds for *longest increasing subsequence* (LIS) in [28, 16, 14]. Yet these bounds only work for large alphabet size. In this paper, we develop several new techniques to handle ED in general and LCS for small alphabet size, thus establishing strong lower bounds for both problems. In particular, our lower bound for ED provides an *exponential* separation between edit distance and Hamming distance in the asymmetric streaming model. Our lower bounds also extend to LIS and *longest non-decreasing subsequence* (LNS) in the standard streaming model. Together with previous results, our bounds provide an almost complete picture for these two problems.

As our second main contribution, we give improved algorithms for ED and LCS in the asymmetric streaming model. For ED, we improve the space complexity of the constant factor approximation algorithms in [15, 13] from $\tilde{O}(\frac{n^\delta}{\delta})$ to $O(\frac{d^\delta}{\delta}\ \mathsf{polylog}(n))$, where n is the length of each string and d is the edit distance between the two strings. For LCS, we give the first $1/2+\varepsilon$ approximation algorithm with space n^δ for any constant $\delta > 0$, over a binary alphabet. Our work leaves a plethora of intriguing open questions, including establishing lower bounds and designing algorithms for a natural generalization of LIS and LNS, which we call *longest non-decreasing subsequence with threshold* (LNST).

2012 ACM Subject Classification Theory of computation → Lower bounds and information complexity

Keywords and phrases Asymmetric Streaming Model, Edit Distance, Longest Common Subsequence, Space Lower Bound

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2021.27

Related Version *Full Version*: https://arxiv.org/abs/2103.00713

Funding *Xin Li*: Supported by NSF CAREER Award CCF-1845349.
Yu Zheng: Supported by NSF CAREER Award CCF-1845349

1 Introduction

Edit distance (ED) and longest common subsequence (LCS) are two classical problems studied in the context of measuring similarities between two strings. Edit distance is defined as the smallest number of edit operations (insertions, deletions, and substitutions) to transform one

string to the other, while longest common subsequence is defined as the longest string that appears as a subsequence in both strings. These two problems have found wide applications in areas such as bioinformatics, text and speech processing, compiler design, data analysis, image analysis and so on. In turn, these applications have led to an extensive study of both problems.

With the era of information explosion, nowadays these two problems are often studied on very large strings. For example, in bioinformatics a human genome can be represented as a string with 3 billion letters (base pairs). Such data provides a huge challenge to the algorithms for ED and LCS, as the standard algorithms for these two problems using dynamic programming need $\Theta(n^2)$ time and $\Theta(n)$ space where n is the length of each string. These bounds quickly become infeasible or too costly as n becomes large, such as in the human genome example. Especially, some less powerful computers may not even have enough memory to store the data, let alone processing it.

One appealing approach to dealing with big data is designing *streaming algorithms*, which are algorithms that process the input as a data stream. Typically, the goal is to compute or approximate the solution by using sublinear space (e.g., n^α for some constant $0 < \alpha < 1$ or even $\mathsf{polylog}(n)$) and a few (ideally one) passes of the data stream. These algorithms have become increasingly popular, and attracted a lot of research activities recently.

Designing streaming algorithms for ED and LCS, however, is not an easy task. For ED, only a couple of positive results are known. In particular, assuming that the edit distance between the two strings is bounded by some parameter k, [10] gives a randomized one pass algorithm achieving an $O(k)$ approximation of ED, using linear time and $O(\log n)$ space, in a variant of the streaming model where one can scan the two strings simultaneously in a coordinated way. In the same model [10] also give randomized one pass algorithms computing ED exactly, using space $O(k^6)$ and time $O(n+k^6)$. This was later improved to space $O(k)$ and time $O(n + k^2)$ in [11, 7]. Furthermore, [7] give a randomized one pass algorithm computing ED exactly, using space $\tilde{O}(k^8)$ and time $\tilde{O}(k^2 n)$, in the standard streaming model. We note that all of these algorithms are only interesting if k is small, e.g., $k \leq n^\alpha$ where α is some small constant, otherwise the space complexity can be as large as n. For LCS, strong lower bounds are given in [22, 28], which show that for exact computation, even constant pass randomized algorithms need space $\Omega(n)$; while any constant pass deterministic algorithm achieving a $\frac{2}{\sqrt{n}}$ approximation of LCS also needs space $\Omega(n)$, if the alphabet size is at least n.

Motivated by this situation and inspired by the work of [4], Saks and Seshadhri [26] studied the asymmetric data streaming model. This model is a relaxation of the standard streaming model, where one has streaming access to one string (say x), and random access to the other string (say y). In this model, [26] gives a deterministic one pass algorithm achieving a $1 + \varepsilon$ approximation of $n - \mathsf{LCS}$ using space $O(\sqrt{(n \log n)/\varepsilon})$, as well as a randomized one pass algorithm algorithm achieving an εn *additive* approximation of LCS using space $O(k \log^2 n/\varepsilon)$ where k is the maximum number of times any symbol appears in y. Another work by Saha [25] also gives an algorithm in this model that achieves an εn *additive* approximation of ED using space $O(\frac{\sqrt{n}}{\epsilon})$.

The asymmetric streaming model is interesting for several reasons. First, it still inherits the spirit of streaming algorithms, and is particularly suitable for a distributed setting. For example, a local, less powerful computer can use the streaming access to process the string x, while sending queries to a remote, more powerful server which has access to y. Second, because it is a relaxation of the standard streaming model, one can hope to design better algorithms for ED or to beat the strong lower bounds for LCS in this model. The latter point is indeed verified by two recent works [15, 13] (recently accepted to ICALP as a combined

paper [12]), which give a deterministic one pass algorithm achieving a $O(2^{1/\delta})$ approximation of ED, using space $\tilde{O}(n^{\delta}/\delta)$ and time $\tilde{O}_{\delta}(n^4)$ for any constant $\delta > 0$, as well as deterministic one pass algorithms achieving $1 \pm \varepsilon$ approximation of ED and LCS, using space $\tilde{O}(\frac{\sqrt{n}}{\varepsilon})$ and time $\tilde{O}_{\varepsilon}(n^2)$.

A natural question is how much we can improve these results. Towards answering this question, we study both lower bounds and upper bounds for the space complexity of ED and LCS in the asymmetric streaming model, and we obtain several new, non-trivial results.

Related work. On a different topic, there are many works that study the time complexity of ED and LCS. In particular, while [6, 1] showed that ED and LCS cannot be computed exactly in truly sub-quadratic time unless the strong Exponential time hypothesis [19] is false, a successful line of work [9, 8, 20, 5, 18, 23, 24] has led to randomized algorithms that achieve constant approximation of ED in near linear time, and randomized algorithms that provide various non-trivial approximation of LCS in linear or sub-quadratic time. Another related work is [4], where the authors proved a lower bound on the *query complexity* for computing ED in the *asymmetric query* model, where one have random access to one string but only limited number of queries to the other string.

1.1 Our Contribution

We initiate a systematic study on lower bounds for computing or approximating ED and LCS in the asymmetric streaming model. To simplify notation we always use $1+\varepsilon$ approximation for some $\varepsilon > 0$, i.e., outputting an λ with $\mathsf{OPT} \leq \lambda \leq (1+\varepsilon)\mathsf{OPT}$, where OPT is either $\mathsf{ED}(x, y)$ or $\mathsf{LCS}(x, y)$. We note that for LCS, this is equivalent to a $1/(1+\varepsilon)$ approximation in the standard notation.

Previously, there are no explicitly stated space lower bounds in this model, although as we will discuss later, some lower bounds about LCS can be inferred from the lower bounds for *longest increasing subsequence* LIS in [28, 16, 14]. As our first contribution, we prove strong lower bounds for ED in the asymmetric streaming model.

▶ **Theorem 1.** *There is a constant $c > 1$ such that for any $k, n \in \mathbb{N}$ with $n \geq ck$, given an alphabet Σ, any R-pass randomized algorithm in the asymmetric streaming model that decides if $\mathsf{ED}(x, y) \geq k$ for two strings $x, y \in \Sigma^n$ with success probability $\geq 2/3$ must use space $\Omega(\mathsf{min}(k, |\Sigma|)/R)$.*

This theorem implies the following corollary.

▶ **Corollary 2.** *Given an alphabet Σ, the following space lower bounds hold for any constant pass randomized algorithm with success probability $\geq 2/3$ in the asymmetric streaming model.*

1. *$\Omega(n)$ for computing $\mathsf{ED}(x, y)$ of two strings $x, y \in \Sigma^n$ if $|\Sigma| \geq n$.*
2. *$\Omega(\frac{1}{\varepsilon})$ for $1+\varepsilon$ approximation of $\mathsf{ED}(x, y)$ for two strings $x, y \in \Sigma^n$ if $|\Sigma| \geq 1/\varepsilon$.*

Our theorems thus provide a justification for the study of *approximating* ED in the asymmetric streaming model. Furthermore, we note that previously, unconditional lower bounds for ED in various computational models are either weak, or almost identical to the bounds for Hamming distance. For example, a simple reduction from the equality function implies the deterministic two party communication complexity (and hence also the space lower bound in the standard streaming model) for computing or even approximating ED is

$\Omega(n)$.[1] However the same bound holds for Hamming distance. Thus it has been an intriguing question to prove a rigorous, unconditional separation of the complexity of ED and Hamming distance. To the best of our knowledge the only previous example achieving this is the work of [3] and [2], which showed that the randomized two party communication complexity of achieving a $1+\varepsilon$ approximation of ED is $\Omega(\frac{\log n}{(1+\varepsilon)\log\log n})$, while the same problem for Hamming distance has an upper bound of $O(\frac{1}{\varepsilon^2})$. Thus if ε is a constant, this provides a separation of $\Omega(\frac{\log n}{\log\log n})$ vs. a constant. However, this result also has some disadvantages: (1) It only works in the randomized setting; (2) The separation becomes obsolete when ε is small, e.g., $\varepsilon = 1/\sqrt{\log n}$; and (3) The lower bound for ED is still weak and thus it does not apply to the streaming setting, as there even recoding the index needs space $\log n$.

Our result from Corollary 2, on the other hand, complements the above result in the aforementioned aspects by providing another strong separation of ED and Hamming distance. Note that even exact computation of the Hamming distance between x and y is easy in the asymmetric streaming model with one pass and space $O(\log n)$. Thus our result provides an *exponential* gap between edit distance and Hamming distance, in terms of the space complexity in the asymmetric streaming model (and also the communication model since our proof uses communication complexity), even for deterministic exact computation.

Next we turn to LCS, which can be viewed as a generalization of LIS. For example, if the alphabet $\Sigma = [n]$, then we can fix the string y to be the concatenation from 1 to n, and it's easy to see that $\mathsf{LCS}(x,y) = \mathsf{LIS}(x)$. Therefore, the lower bound of computing LIS for randomized streaming in [28] with $|\Sigma| \geq n$ also implies a similar bound for LCS in the asymmetric streaming model. However, the bound in [28] does not apply to the harder case where x is a *permutation* of y, and their lower bound where $|\Sigma| < n$ is actually for *longest non-decreasing subsequence*, which does not give a similar bound for LCS in the asymmetric streaming model. [2] Therefore, we first prove a strong lower bound for LCS in general.

▶ **Theorem 3.** *There is a constant $c > 1$ such that for any $k, n \in \mathbb{N}$ with $n \geq ck$, given an alphabet Σ, any R-pass randomized algorithm in the asymmetric streaming model that decides if $\mathsf{LCS}(x,y) \geq k$ for two strings $x, y \in \Sigma^n$ with success probability $\geq 2/3$ must use space $\Omega\big(\mathsf{min}(k, |\Sigma|)/R\big)$. Moreover, this holds even if x is a permutation of y when $|\Sigma| \geq n$ or $|\Sigma| \leq k$.*

Similar to the case of ED, this theorem also implies the following corollary.

▶ **Corollary 4.** *Given an alphabet Σ, the following space lower bounds hold for any constant pass randomized algorithm with success probability $\geq 2/3$ in the asymmetric streaming model.*
1. *$\Omega(n)$ for computing $\mathsf{LCS}(x,y)$ of two strings $x, y \in \Sigma^n$ if $|\Sigma| \geq n$.*
2. *$\Omega(\frac{1}{\varepsilon})$ for $1+\varepsilon$ approximation of $\mathsf{LCS}(x,y)$ for two strings $x, y \in \Sigma^n$ if $|\Sigma| \geq 1/\varepsilon$.*

We then consider *deterministic* approximation of LCS. Here, the work of [16, 14] gives a lower bound of $\Omega\left(\frac{1}{R}\sqrt{\frac{n}{\varepsilon}}\log\left(\frac{|\Sigma|}{\varepsilon n}\right)\right)$ for any R pass streaming algorithm achieving a $1+\varepsilon$ approximation of LIS, which also implies a lower bound of $\Omega\left(\frac{1}{R}\sqrt{\frac{n}{\varepsilon}}\log\left(\frac{1}{\varepsilon}\right)\right)$ for asymmetric streaming LCS when $|\Sigma| \geq n$. These bounds match the upper bound in [17] for LIS and LNS, and in [15, 13] for LCS. However, a major drawback of this bound is that it gives nothing when $|\Sigma|$ is small (e.g., $|\Sigma| \leq \varepsilon n$). For even smaller alphabet size, the bound does not even

[1] We include this bound in the appendix for completeness, as we cannot find any explicit statement in the literature.

[2] One can get a similar reduction to LCS, but now y needs to be the sorted version of x, which gives additional information about x in the asymmetric streaming model since we have random access to y.

give anything for exact computation. For example, in the case of a binary alphabet, we know that $\mathsf{LIS}(x) \leq 2$ and thus taking $\varepsilon = 1/2$ corresponds to exact computation. Yet the bound gives a negative number.

This is somewhat disappointing as in most applications of ED and LCS, the alphabet size is actually a fixed constant. These include for example the English language and the human DNA sequence (where the alphabet size is 4 for the 4 bases). Therefore, in this paper we focus on the case where the alphabet size is small, and we have the following theorem.

▶ **Theorem 5.** *Given an alphabet Σ, for any $\varepsilon > 0$ where $\frac{|\Sigma|^2}{\varepsilon} = O(n)$, any R-pass deterministic algorithm in the asymmetric streaming model that computes a $1+\varepsilon$ approximation of $\mathsf{LCS}(x, y)$ for two strings $x, y \in \Sigma^n$ must use space $\Omega\left(\frac{|\Sigma|}{\varepsilon}/R\right)$.*

Thus, even for a binary alphabet, achieving $1+\varepsilon$ approximation for small ε (e.g., $\varepsilon = 1/n$ which corresponds to exact computation) can take space as large as $\Omega(n)$ for any constant pass algorithm. Further note that by taking $|\Sigma| = \sqrt{\varepsilon n}$, we recover the $\Omega\left(\frac{\sqrt{n}}{\varepsilon}/R\right)$ bound with a much smaller alphabet.

Finally, we turn to LIS and longest non-decreasing subsequence (LNS), as well as a natural generalization of LIS and LNS which we call *longest non-decreasing subsequence with threshold* (LNST). Given a string $x \in \Sigma^n$ and a threshold $t \leq n$, $\mathsf{LNST}(x, t)$ denotes the length of the longest non-decreasing subsequence in x such that each symbol appears at most t times. It is easy to see that the case of $t = 1$ corresponds to LIS and the case of $t = n$ corresponds to LNS. Thus LNST is indeed a generalization of both LIS and LNS. It is also a special case of LCS when $|\Sigma|t \leq n$ as we can take y to be the concatenation of t copies of each symbol, in the ascending order (and possibly padding some symbols not in x). How hard is LNST? We note that in the case of $t = 1$ (LIS) and $t = n$ (LNS) a simple dynamic programming can solve the problem in one pass with space $O(|\Sigma| \log n)$, and $1+\varepsilon$ approximation can be achieved in one pass with space $\tilde{O}(\sqrt{\frac{n}{\varepsilon}})$ by [17]. Thus one can ask what is the situation for other t. Again we focus on the case of a small alphabet and have the following theorem.

▶ **Theorem 6.** *Given an alphabet Σ, for deterministic $(1+\varepsilon)$ approximation of $\mathsf{LNST}(x, t)$ for a string $x \in \Sigma^n$ in the streaming model with R passes, we have the following space lower bounds:*

1. $\Omega(\min(\sqrt{n}, |\Sigma|)/R)$ *for any constant t (this includes* LIS*), when ε is any constant.*
2. $\Omega(|\Sigma| \log(1/\varepsilon)/R)$ *for $t \geq n/|\Sigma|$ (this includes* LNS*), when $|\Sigma|^2/\varepsilon = O(n)$.*
3. $\Omega\left(\frac{\sqrt{|\Sigma|}}{\varepsilon}/R\right)$ *for $t = \Theta(1/\varepsilon)$, when $|\Sigma|/\varepsilon = O(n)$.*

Thus, case 1 and 2 show that even for any constant approximation, any constant pass streaming algorithm for LIS and LNS needs space $\Omega(|\Sigma|)$ when $|\Sigma| \leq \sqrt{n}$, matching the $O(|\Sigma| \log n)$ upper bound up to a logarithmic factor. Taking $\varepsilon = 1/\sqrt[3]{n}$ and $|\Sigma| \leq \sqrt[3]{n}$ for example, we further get a lower bound of $\Omega(|\Sigma| \log n)$ for approximating LNS using any constant pass streaming algorithm. This matches the $O(|\Sigma| \log n)$ upper bound. These results complement the bounds in [16, 14, 17] for the important case of small alphabet, and together they provide an almost complete picture for LIS and LNS. Case 3 shows that for certain choices of t and ε, the space we need for LNST can be significantly larger than those for LIS and LNS. It is an intriguing question to completely characterize the behavior of LNST for all regimes of parameters.

We also give improved algorithms for asymmetric streaming ED and LCS. For ED, [15, 13] gives a $O(2^{1/\delta})$-approximation algorithm with $\tilde{O}(n^\delta)$ space for any constant $\delta \in (0, 1)$. We further reduced the space needed from $\tilde{O}(\frac{n^\delta}{\delta})$ to $O(\frac{d^\delta}{\delta}\ \mathsf{polylog}(n))$ where $d = \mathsf{ED}(x, y)$. Specifically, we have the following theorem.

▶ **Theorem 7.** *Assume* $\mathsf{ED}(x,y) = d$*, in the asymmetric streaming model, there are one-pass deterministic algorithms in polynomial time with the following parameters:*

1. *A* $(3+\varepsilon)$*-approximation of* $\mathsf{ED}(x,y)$ *using* $O(\sqrt{d}\,\mathsf{polylog}(n))$ *space.*
2. *For any constant* $\delta \in (0,1/2)$*, a* $2^{O(\frac{1}{\delta})}$*-approximation of* $\mathsf{ED}(x,y)$ *using* $O(\frac{d^{\delta}}{\delta}\mathsf{polylog}(n))$ *space.*

For LCS over a large alphabet, the upper bounds in [15, 13] match the lower bounds implied by [16, 14]. We thus again focus on small alphabet. Note that our Theorem 5 does not give anything useful if $|\Sigma|$ is small and ε is large (e.g., both are constants). Thus a natural question is whether one can get better bounds. In particular, is the dependence on $1/\varepsilon$ linear as in our theorem, or is there a threshold beyond which the space jumps to say for example $\Omega(n)$? We note that there is a trivial one pass, $O(\log n)$ space algorithm even in the standard streaming model that gives a $|\Sigma|$ approximation of LCS (or $1/|\Sigma|$ approximation in standard notation), and no better approximation using sublinear space is known even in the asymmetric streaming model. Thus one may wonder whether this is the threshold. We show that this is not the case, by giving a one pass algorithm in the asymmetric streaming model over the binary alphabet that achieves a $2-\varepsilon$ approximation of LCS (or $1/2+\varepsilon$ approximation in standard notation), using space n^{δ} for any constant $\delta > 0$.

▶ **Theorem 8.** *For any constant* $\delta \in (0,1/2)$*, there exists a constant* $\varepsilon > 0$ *and a one-pass deterministic algorithm that outputs a* $2-\varepsilon$ *approximation of* $\mathsf{LCS}(x,y)$ *for any two strings* $x, y \in \{0,1\}^n$*, with* $\tilde{O}(n^{\delta}/\delta)$ *space and polynomial time in the asymmetric streaming model.*

Finally, as mentioned before, we now have an almost complete picture for LIS and LNS, but for the more general LNST the situation is still far from clear. Since LNST is a special case of LCS, if $|\Sigma|t = O(n)$ then the upper bound of $\tilde{O}(\frac{\sqrt{n}}{\varepsilon})$ in [15, 13] still applies and this matches our lower bound in case 3, Theorem 6 by taking $|\Sigma| = \varepsilon n$. One can then ask the natural question of whether we can get a matching upper bound for the case of small alphabet. We are not able to achieve this, but we provide a simple algorithm that can use much smaller space for certain regimes of parameters in this case.

▶ **Theorem 9.** *Given an alphabet* Σ *with* $|\Sigma| = r$*. For any* $\varepsilon > 0$ *and* $t \geq 1$*, there is a one-pass streaming algorithm that computes a* $(1+\varepsilon)$ *approximation of* $\mathsf{LNST}(x,t)$ *for any* $x \in \Sigma^n$ *with* $\tilde{O}\Big(\big(\min(t, r/\varepsilon)+1\big)^r\Big)$ *space.*

1.2 Overview of our Techniques

Here we provide an informal overview of the techniques used in this paper.

1.2.1 Lower Bounds

Our lower bounds use the general framework of communication complexity. To limit the power of random access to the string y, we always fix y to be a specific string, and consider different strings x. In turn, we divide x into several blocks and consider the two party/multi party communication complexity of $\mathsf{ED}(x,y)$ or $\mathsf{LCS}(x,y)$, where each party holds one block of x. However, we need to develop several new techniques to handle edit distance and small alphabets.

Edit distance. We start with edit distance. One difficulty here is to handle substitutions, as with substitutions edit distance becomes similar to Hamming distance, and this is exactly one of the reasons why strong complexity results separating edit distance and Hamming

distance are rare. Indeed, if we define $\mathsf{ED}(x, y)$ to be the smallest number of insertions and deletions (without substitutions) to transform x into y, then $\mathsf{ED}(x, y) = 2n - 2\mathsf{LCS}(x, y)$ and thus a lower bound for exactly computing LCS (e.g., those implied from [16, 14]) would translate directly into the same bound for exactly computing ED. On the other hand, with substitutions things become more complicated: if $\mathsf{LCS}(x, y)$ is small (e.g., $\mathsf{LCS}(x, y) \leq n/2$) then in many cases (such as examples obtained by reducing from [16, 14]) the best option to transform x into y is just replacing each symbol in x by the corresponding symbol in y if they are different, which makes $\mathsf{ED}(x, y)$ exactly the same as their Hamming distance.

To get around this, we need to ensure that $\mathsf{LCS}(x, y)$ is large. We demonstrate our ideas by first describing an $\Omega(n)$ lower bound for the deterministic two party communication complexity of $\mathsf{ED}(x, y)$, using a reduction from the equality function which is well known to have an $\Omega(n)$ communication complexity bound. Towards this, fix $\Sigma = [3n] \cup \{a\}$ where a is a special symbol, and fix $y = 1 \circ 2 \circ \cdots \circ 3n$. We divide x into two parts $x = (x_1, x_2)$ such that x_1 is obtained from the string $(1, 2, 4, 5, \cdots, 3i - 2, 3i - 1, \cdots, 3n - 2, 3n - 1)$ by replacing some symbols of the form $3j - 1$ by a, while x_2 is obtained from the string $(2, 3, 5, 6, \cdots, 3i - 1, 3i, \cdots, 3n - 1, 3n)$ by replacing some symbols of the form $3j - 1$ by a. Note that the way we choose (x_1, x_2) ensures that $\mathsf{LCS}(x, y) \geq 2n$ before replacing any symbol by a.

Intuitively, we want to argue that the best way to transform x into y, is to delete a substring at the end of x_1 and a substring at the beginning of x_2, so that the resulted string becomes an increasing subsequence as long as possible. Then, we insert symbols into this string to make it match y except for those a symbols. Finally, we replace the a symbols by substitutions. If this is true then we can finish the argument as follows. Let $T_1, T_2 \subset [n]$ be two subsets with size $t = \Omega(n)$, where for any $i \in \{1, 2\}$, all symbols of the form $3j - 1$ in x_i with $j \in T_i$ are replaced by a. Now if $T_1 = T_2$ then it doesn't matter where we choose to delete the substrings in x_1 and x_2, the number of edit operations is always $3n - 2 + t$ by a direct calculation. On the other hand if $T_1 \neq T_2$ and assume for simplicity that the smallest element they differ is an element in T_2, then there is a way to save one substitution, and the the number of edit operations becomes $3n - 3 + t$.

The key part is now proving our intuition. For this, we consider all possible $r \in [3n]$ such that x_1 is transformed into $y[1 : r]$ and x_2 is transformed into $y[r + 1 : 3n]$, and compute the two edit distances respectively. To analyze the edit distance, we first show by a greedy argument that without loss of generality, we can assume that we apply deletions first, followed by insertions, and substitutions at last. This reduces the edit distance problem to the following problem: for a fixed number of deletions and insertions, what is the best way to minimize the Hamming distance (or maximize the number of agreements of symbols at the same indices) in the end. Now we break the analysis of $\mathsf{ED}(x_1, y[1 : r])$ into two cases. Case 1 is where the number of deletions (say d_d) is large. In this case, the number of insertions (say d_i) must also be large, and we argue that the number of agreements is at most $\mathsf{LCS}(x_1, y[1 : r]) + d_i$. Case 2 is where d_d is small. In this case, d_i must also be small. Now we crucially use the structure of x_1 and y, and argue that symbols in x_1 larger than $3d_i$ (or original index beyond $2d_i$) are guaranteed to be out of agreement. Thus the number of agreements is at most $\mathsf{LCS}(x_1[1 : 2d_i], y[1 : r]) + d_i$. In each case combining the bounds gives us a lower bound on the total number of operations. The situation for x_2 and $y[r + 1 : 3n]$ is completely symmetric and this proves our intuition.

In the above construction, x and y have different lengths ($|x| = 4n$ while $|y| = 3n$). We can fix this by adding a long enough string z with distinct symbols than those in $\{x, y\}$ to the end of both x and y, and then add n symbols of a at the end of z for y. We argue that

the best way to do the transformation is to transform x into y, and then insert n symbols of a. To show this, we first argue that at least one symbol in z must be kept, for otherwise the number of operations is already larger than the previous transformation. Then, using a greedy argument we show that the entire z must be kept, and thus the natural transformation is the optimal.

To extend the bound to randomized algorithms, we modify the above construction and reduce from *Set Disjointness* (DIS), which is known to have randomized communication complexity $\Omega(n)$. Given two strings $\alpha, \beta \in \{0,1\}^n$ representing the characteristic vectors of two sets $A, B \subseteq [n]$, $\mathsf{DIS}(\alpha, \beta) = 0$ if and only if $A \cap B \neq \emptyset$, or equivalently, $\exists j \in [n], \alpha_j = \beta_j = 1$. For the reduction, we first create two new strings $\alpha', \beta' \in \{0,1\}^{2n}$ which are "balanced" versions of α, β. Formally, $\forall j \in [n], \alpha'_{2j-1} = \alpha_j$ and $\alpha'_{2j} = 1 - \alpha_j$. We create β' slightly differently, i.e., $\forall j \in [n], \beta'_{2j-1} = 1 - \beta_j$ and $\beta'_{2j} = \beta_j$. Now both α' and β' have n 1's, we can use them as the characteristic vectors of the two sets T_1, T_2 in the previous construction. A similar argument now leads to the bound for randomized algorithms.

Longest common subsequence. Our lower bounds for randomized algorithms computing LCS exactly are obtained by a similar and simpler reduction from DIS: we still fix y to be an increasing sequence of length $8n$ and divide y evenly into $4n$ blocks of constant size. Now x_1 consists of the blocks with an odd index, while x_2 consists of the blocks with an even index. Thus x is a permutation of y. Next, from $\alpha, \beta \in \{0,1\}^n$ we create $\alpha', \beta' \in \{0,1\}^{2n}$ in a slightly different way and use α', β' to modify the $2n$ blocks in x_1 and x_2 respectively. If a bit is 1 then we arrange the corresponding block in the increasing order, otherwise we arrange the corresponding block in the decreasing order. A similar argument as before now gives the desired $\Omega(n)$ bound. We note that [28] has similar results for LIS by reducing from DIS. However, our reduction and analysis are different from theirs. Thus we can handle LCS, and even the harder case where x is a permutation of y.

We now turn to LCS over a small alphabet. To illustrate our ideas, let's first consider $\Sigma = \{0,1\}$ and choose $y = 0^{n/2}1^{n/2}$. It is easy to see that $\mathsf{LCS}(x, y) = \mathsf{LNST}(x, n/2)$. We now represent each string $x \in \{0,1\}^n$ as follows: at any index $i \in [n] \cup \{0\}$, we record a pair (p, q) where $p = \mathsf{min}(\text{the number of 0's in } x[1:i], n/2)$ and $q = \mathsf{min}(\text{the number of 1's in } x[i+1:n], n/2)$. Thus, if we read x from left to right, then upon reading a 0, p may increase by 1 and q does not change; while upon reading a 1, p does not change and q may decrease by 1. Hence if we use the horizontal axis to stand for p and the vertical axis to stand for q, then these points (p, q) form a polygonal chain. We call $p + q$ the *value* at point (p, q) and it is easy to see that $\mathsf{LCS}(x, y)$ must be the value of an endpoint of some chain segment.

Using the above representation, we now fix $\Sigma = \{0,1,2\}$ and choose $y = 0^{n/3}1^{n/3}2^{n/3}$, so $\mathsf{LCS}(x, y) = \mathsf{LNST}(x, n/3)$. We let $x = (x_1, x_2)$ such that $x_1 \in \{0,1\}^{n/2}$ and $x_2 \in \{1,2\}^{n/2}$. Since any common subsequence between x and y must be of the form $0^a1^b2^c$ it suffices to consider common subsequence between x_1 and $0^{n/3}1^{n/3}$, and that between x_2 and $1^{n/3}2^{n/3}$, and combine them together. Towards that, we impose the following properties on x_1, x_2: (1) The number of 0's, 1's, and 2's in each string is at most $n/3$; (2) In the polygonal chain representation of each string, the values of the endpoints strictly increase when the number of 1's increases; and (3) For any endpoint in x_1 where the number of 1's is some r, there is a corresponding endpoint in x_2 where the number of 1's is $n/3 - r$, and the values of these two endpoints sum up to a fixed number $t = \Omega(n)$. Note that property (2) implies that $\mathsf{LCS}(x, y)$ must be the sum of the values of an endpoint in x_1 where the number of 1's is some r, and an endpoint in x_2 where the number of 1's is $n/3 - r$, while property (3) implies that for any string x_1, there is a unique corresponding string x_2, and $\mathsf{LCS}(x, y) = t$ (regardless of the choice of r).

We show that under these properties, all possible strings $x = (x_1, x_2)$ form a set S with $|S| = 2^{\Omega(n)}$, and this set gives a *fooling set* for the two party communication problem of computing $\mathsf{LCS}(x, y)$. Indeed, for any $x = (x_1, x_2) \in S$, we have $\mathsf{LCS}(x, y) = t$. On the other hand, for any $(x_1, x_2) \neq (x'_1, x'_2) \in S$, the values must differ at some point for x_1 and x'_1. Hence by switching, either (x_1, x'_2) or (x'_1, x_2) will have a LCS with y that has length at least $t + 1$. Standard arguments now imply an $\Omega(n)$ communication complexity lower bound. A more careful analysis shows that we can even replace the symbol 2 by 0, thus resulting in a binary alphabet.

The above argument can be easily modified to give a $\Omega(1/\varepsilon)$ bound for $1+\varepsilon$ approximation of LCS when $\varepsilon < 1$, by taking the string length to be some $n' = \Theta(1/\varepsilon)$. To get a better bound, we combine our technique with the technique in [14] and consider the following direct sum problem: we create r copies of strings $\{x^i, i \in [r]\}$ and $\{y^i, i \in [r]\}$ where each copy uses distinct alphabets with size 2. Assume for x^i and y^i the alphabet is $\{a_i, b_i\}$, now x^i again consists of r copies of $(x^i_{j1}, x^i_{j2}), j \in [r]$, where each $x^i_{j\ell} \in \{a_i, b_i\}^{n'/2}$ for $\ell \in [2]$; while y^i consists of r copies $y^i_j = a_i^{n'/3} b_i^{n'/3} a_i^{n'/3}, j \in [r]$. The direct sum problem is to decide between the following two cases for some $t = \Omega(n')$: (1) $\exists i$ such that there are $\Omega(r)$ copies (x^i_{j1}, x^i_{j2}) in x^i with $\mathsf{LCS}((x^i_{j1} \circ x^i_{j2}), y^i_j) \geq t + 1$, and (2) $\forall i$ and $\forall j$, $\mathsf{LCS}((x^i_{j1} \circ x^i_{j2}), y^i_j) \leq t$. We do this by arranging the x^i's row by row into an $r \times 2r$ matrix (each entry is a length $n'/2$ string) and letting x be the concatenation of the *columns*. We call these strings the *contents* of the matrix, and let y be the concatenation of the y^i's. Now intuitively, case (1) and case (2) correspond to deciding whether $\mathsf{LCS}(x, y) \geq 2rt + \Omega(r)$ or $\mathsf{LCS}(x, y) \leq 2rt$, which implies a $1 + \Omega(1/t) = 1 + \varepsilon$ approximation. The lower bound follows by analyzing the $2r$-party communication complexity of this problem, where each party holds a column of the matrix.

However, unlike the constructions in [16, 14] which are relatively easy to analyze because all symbols in x (respectively y) are *distinct*, the repeated symbols in our construction make the analysis of LCS much more complicated (we can also use distinct symbols but that will only give us a bound of $\frac{\sqrt{|\Sigma|}}{\varepsilon}$ instead of $\frac{|\Sigma|}{\varepsilon}$). To ensure that the LCS is to match each (x^i_{j1}, x^i_{j2}) to the corresponding y^i_j, we use another r symbols $\{c_i, i \in [r]\}$ and add *buffers* of large size (e.g., size n') between adjacent copies of (x^i_{j1}, x^i_{j2}). We do the same thing for y^i_j correspondingly. Moreover, it turns out we need to arrange the buffers carefully to avoid unwanted issues: in each row x^i, between each copy of (x^i_{j1}, x^i_{j2}) we use a buffer of new symbol. Thus the buffers added to each row x^i are $c_1^{n'}, c_2^{n'}, \cdots, c_r^{n'}$ sequentially and this is the same for every row. That is, in each row the contents use the same alphabet $\{a_i, b_i\}$ but the buffers use different alphabets $\{c_i, i \in [r]\}$. Now we have a $r \times 3r$ matrix and we again let x be the concatenation of the *columns* while let y be the concatenation of the y^i's. Note that we are using an alphabet of size $|\Sigma| = 3r$. We use a careful analysis to argue that case (1) and case (2) now correspond to deciding whether $\mathsf{LCS}(x, y) \geq 2rn' + rt + \Omega(r)$ or $\mathsf{LCS}(x, y) \leq 2rn' + rt$, which implies a $1 + \varepsilon$ approximation. The lower bound follows by analyzing the $3r$-party communication complexity of this problem, and we show a lower bound of $\Omega(r/\varepsilon) = \Omega(|\Sigma|/\varepsilon)$ by generalizing our previous fooling set construction to the multi-party case, where we use a good error correcting code to create the $\Omega(r)$ gap.

The above technique works for $\varepsilon < 1$. For the case of $\varepsilon \geq 1$ our bound for LCS can be derived directly from our bound for LIS, which we describe next.

Longest increasing/non-decreasing subsequence. Our $\Omega(|\Sigma|)$ lower bound over small alphabet is achieved by modifying the construction in [14] and providing a better analysis. Similar as before, we consider a matrix $B \in \{0, 1\}^{\frac{r}{c} \times r}$ where c is a large constant and $r = |\Sigma|$. We now consider the r-party communication problem where each party holds one column of

B, and the problem is to decide between the following two cases for a large enough constant l: (1) for each row in B, there are at least l 0's between any two 1's, and (2) there exists a row in B which has more than αr 1's, where $\alpha \in (1/2, 1)$ is a constant. We can use a similar argument as in [14] to show that the total communication complexity of this problem is $\Omega(r^2)$ and hence at least one party needs $\Omega(r)$. The difference is that [14] sets $l = 1$ while we need to pick l to be a larger constant to handle the case $\varepsilon \geq 1$. For this we use the Lovász Local Lemma with a probabilistic argument to show the existence of a large fooling set. To reduce to LIS, we define another matrix $\tilde{B}$ such that $\tilde{B}_{i,j} = (i-1)\frac{r}{c} + j$ if $B_{i,j} = 1$ and $\tilde{B}_{i,j} = 0$ otherwise. Now let x be the concatenation of all columns of $\tilde{B}$. We show that case (2) implies $\mathsf{LIS}(x) \geq \alpha r$ and case (1) implies $\mathsf{LIS}(x) \leq (1/c + 1/l)r$. This implies a $1+\varepsilon$ approximation for any constant $\varepsilon > 0$ by setting c and l appropriately.

The construction is slightly different for LNS. This is because if we keep the 0's in $\tilde{B}$, they will already form a very long non-decreasing subsequence and we will not get any gap. Thus, we now let the matrix B have size $r \times cr$ where c can be any constant. We replace all 0's in column i with a symbol b_i for $i \in [cr]$, such that $b_1 > b_2 > \cdots > b_{cr}$. Similarly we replace all 1's in row j with a symbol a_j for $j \in [r]$, such that $a_1 < a_2 < \cdots < a_r$. Also, we let $a_1 > b_1$. We can show that the two cases now correspond to $\mathsf{LNS}(x) > \alpha cr$ and $\mathsf{LNS}(x) \leq (2 + c/l)r$.

We further prove an $\Omega(|\Sigma| \log(1/\varepsilon))$ lower bound for $1+\varepsilon$ approximation of LNS when $\varepsilon < 1$. This is similar to our previous construction for LCS, except we don't need buffers here, and we only need to record the number of some symbols. More specifically, let $l = \Theta(1/\varepsilon)$ and S be the set of all strings $x = (x_1, x_2)$ over alphabet $\{a, b\}$ with length $2l$ such that $x_1 = a^{\frac{3}{4}l+t}b^{\frac{1}{4}l-t}$ and $x_2 = a^{\frac{3}{4}l-t}b^{\frac{1}{4}l+t}$ for any $t \in [\frac{l}{4}]$. Thus S has size $\frac{l}{4} = \Omega(1/\varepsilon)$ and $\forall x \in S$, the number of a's in x is exactly $\frac{3}{2}l$. Further, for any $(x_1, x_2) \neq (x_1', x_2') \in S$, either (x_1, x_2') or (x_1', x_2) has more than $\frac{3}{2}l$ a's. We now consider the $r \times 2r$ matrix where each row i consists of $\{(x^i_{j1}, x^i_{j2}), j \in [r]\}$ such that each $x^i_{j\ell}$ has length l for $\ell \in [2]$, and for the same row i all $\{(x^i_{j1}, x^i_{j2})\}$ use the same alphabet $\{a_i, b_i\}$ while for different rows the alphabets are disjoint. To make sure the LNS of the concatenation of the columns is roughly the sum of the number of a_i's, we require that $b_r < b_{r-1} < \cdots < b_1 < a_1 < a_2 < \cdots < a_r$. Now we analyze the $2r$ party communication problem of deciding whether the concatenation of the columns has $\mathsf{LNS} \geq crl + \Omega(r)$ or $\mathsf{LNS} \leq crl$ for some constant c, which implies a $1+\varepsilon$ approximation. The lower bound is again achieved by generalizing the set S to a fooling set for the $2r$ party communication problem using an error correcting code based approach.

In Theorem 6, we give three lower bounds for LNST. The first two lower bounds are adapted from our lower bounds for LIS and LNS, while the last lower bound is adapted from our lower bound for LCS by ensuring all symbols in different rows or columns of the matrix there are different.

Improved algorithms. We defer the technique overview of our improved algorithms to Section B.

1.3 Open Problems

Our work leaves a plethora of intriguing open problems. The main one is to close the gap between our lower bounds and the upper bounds of known algorithms, especially for the case of small alphabets and large (say constant) approximation. We believe that in this case it is possible to improve both the lower bounds and the upper bounds. Another interesting problem is to completely characterize the space complexity of LNST.

2 Organization of the paper

The rest of the paper is organized as follows. In Section 3, we give a formal description of the problems we study. We then present our lower bounds for edit distance in Section 4 and lower bounds for LCS in Section 5. In the appendix, we give our lower bounds for LIS, LNS, LNST, and an algorithm for LNST in Section A. In Section B, we present our improved algorithms for asymmetric streaming edit distance and LCS. Finally in Section C, we present a proof for the strong linear lower bound for approximating ED in the standard streaming model. Due to the page limit, some of the proofs are deferred to the full version.

3 Preliminaries

We use the following conventional notations. Let $x \in \Sigma^n$ be a string of length n over alphabet Σ. By $|x|$, we mean the length of x. We denote the i-th character of x by x_i and the substring from the i-th character to the j-th character by $x[i:j]$. We denote the concatenation of two strings x and y by $x \circ y$. By $[n]$, we mean the set of positive integers no larger than n.

Edit Distance. The *edit distance* (or *Levenshtein distance*) between two strings $x, y \in \Sigma^*$, denoted by $\mathsf{ED}(x, y)$, is the smallest number of edit operations (insertion, deletion, and substitution) needed to transform one into another.

Longest Common Subsequence. We say the string $s \in \Sigma^t$ is a *subsequence* of $x \in \Sigma^n$ if there exists indices $1 \le i_1 < i_2 < \cdots < i_t \le n$ such that $s = x_{i_1} x_{i_2} \cdots x_{i_t}$. A string s is called a *common subsequence* of strings x and y if s is a subsequence of both x and y. Given two strings x and y, we denote the length of the longest common subsequence (LCS) of x and y by $\mathsf{LCS}(x, y)$.

Longest Increasing Subsequence. In the longest increasing subsequence problem, we assume there is a given total order on the alphabet set Σ. We say the string $s \in \Sigma^t$ is an *increasing subsequence* of $x \in \Sigma^n$ if there exists indices $1 \le i_1 < i_2 < \cdots < i_t \le n$ such that $s = x_{i_1} x_{i_2} \cdots x_{i_t}$ and $x_{i_1} < x_{i_2} < \cdots < x_{i_t}$. We denote the length of the longest increasing subsequence (LIS) of string x by $\mathsf{LIS}(x)$.

Longest Non-decreasing Subsequence. The longest non-decreasing subsequence is a variant of the longest increasing problem. The difference is that in a non-decreasing subsequence $s = x_{i_1} x_{i_2} \cdots x_{i_t}$, we only require $x_{i_1} \le x_{i_2} \le \cdots \le x_{i_t}$.

4 Lower Bounds for Edit Distance

We show a reduction from the Set Disjointness problem (DIS) to computing ED between two strings in the asymmetric streaming model. For this, we define the following two party communication problem between Alice and Bob.

Given an alphabet Σ and three integers n_1, n_2, n_3. Suppose Alice has a string $x_1 \in \Sigma^{n_1}$ and Bob has a string $x_2 \in \Sigma^{n_1}$. There is another fixed reference string $y \in \Sigma^{n_3}$ that is known to both Alice and Bob. Alice and Bob now tries to compute $\mathsf{ED}((x_1 \circ x_2), y)$.We call this problem $\mathsf{ED}_{cc}(y)$. We prove the following theorem.

▶ **Theorem 10.** *Suppose each input string to* DIS *has length n and let $\Sigma = [6n] \cup \{a\}$. Fix $y = (1, 2, \cdots, 6n)$. Then $R^{1/3}(\mathsf{ED}_{cc}(y)) \ge R^{1/3}(\mathsf{DIS})$.*

To prove this theorem, we first construct the strings x_1, x_2 based on the inputs $\alpha, \beta \in \{0,1\}^n$ to DIS. From α, Alice constructs the string $\alpha' \in \{0,1\}^{2n}$ such that $\forall j \in [n], \alpha'_{2j-1} = \alpha_j$ and $\alpha'_{2j} = 1 - \alpha_j$. Similarly, from β, Bob constructs the string $\beta' \in \{0,1\}^{2n}$ such that

$\forall j \in [n], \beta'_{2j-1} = 1 - \beta_j$ and $\beta'_{2j} = \beta_j$. Now Alice lets x_1 be a modification from the string $(1, 2, 4, 5, \cdots, 3i-2, 3i-1, \cdots, 6n-2, 6n-1)$ such that $\forall j \in [2n]$, if $\alpha'_j = 0$ then the symbol $3j-1$ (at index $2j$) is replaced by a. Similarly, Bob lets x_2 be a modification from the string $(2, 3, 5, 6, \cdots, 3i-1, 3i, \cdots, 6n-1, 6n)$ such that $\forall j \in [2n]$, if $\beta'_j = 0$ then the symbol $3j-1$ (at index $2j-1$) is replaced by a.

Given the construction, we have the following lemma.

▶ **Lemma 11.** *If* $\mathsf{DIS}(\alpha, \beta) = 1$ *then* $\mathsf{ED}((x_1 \circ x_2), y) \geq 7n - 2$.

To prove the lemma we observe that in a series of edit operations that transforms (x_1, x_2) to y, there exists an index $r \in [6n]$ s.t. x_1 is transformed into $[1 : r]$ and x_2 is transformed into $[r+1 : n]$. We analyze the edit distance in each part. We first have the following claim:

▷ Claim 12. For any two strings u and v, there is a sequence of optimal edit operations (insertion/deletion/substitution) that transforms u to v, where all deletions happen first, followed by all insertions, and all substitutions happen at the end of the operations.

We defer the proof of this claim to the full version. For any i, let $\Gamma_1(i)$ denote the number of a symbols up to index $2i$ in x_1. Note that $\Gamma_1(i)$ is equal to the number of 0's in $\alpha'[1 : i]$. We have the following lemma.

▶ **Lemma 13.** *For any* $p \in [n]$*, let* $r = 3p - q$ *where* $0 \leq q \leq 2$*, then* $\mathsf{ED}(x_1, [1 : r]) = 4n - p - q + \Gamma_1(p)$ *if* $q = 0, 1$ *and* $\mathsf{ED}(x_1, [1 : r]) = 4n - p + \Gamma_1(p-1)$ *if* $q = 2$.

Proof. By Claim 12 we can first consider deletions and insertions, and then compute the Hamming distance after these operations (for substitutions).

We consider the three different cases of q. Let the number of insertions be d_i and the number of deletions be d_d. Note that $d_i - d_d = r - 4n$. We define the number of agreements between two strings to be the number of positions where the two corresponding symbols are equal.

The case of $q = 0$ and $q = 1$. Here again we have two cases.

Case (a): $d_d \geq 4n - 2p$. In this case, notice that the LCS after the operations between x_1 and y is at most the original $\mathsf{LCS}(x_1, y) = 2p - \Gamma_1(p)$. With d_i insertions, the number of agreements can be at most $\mathsf{LCS}(x_1, y) + d_i = 2p - \Gamma_1(p) + d_i$, thus the Hamming distance at the end is at least $r - 2p + \Gamma_1(p) - d_i$. Therefore, in this case the number of edit operations is at least $d_i + d_d + r - 2p + \Gamma_1(p) - d_i \geq 4n - p - q + \Gamma_1(p)$, and the equality is achieved when $d_d = 4n - 2p$.

Case (b): $d_d < 4n - 2p$. In this case, notice that all original symbols in x_1 larger than $3d_i$ (or beyond index $2d_i$ before the insertions) are guaranteed to be out of agreement. Thus the only possible original symbols in x_1 that are in agreement with y after the operations are the symbols with original index at most $2d_i$. Note that the LCS between $x_1[1 : 2d_i]$ and y is $2d_i - \Gamma_1(d_i)$. Thus with d_i insertions the number of agreements is at most $3d_i - \Gamma_1(d_i)$, and the Hamming distance at the end is at least $r - 3d_i + \Gamma_1(d_i)$.
Therefore the number of edit operations is at least $d_i + d_d + r - 3d_i + \Gamma_1(d_i) = r - d_i + (d_d - d_i) + \Gamma_1(d_i) = 4n - d_i + \Gamma_1(d_i)$. Now notice that $d_i = d_d + r - 4n < p$ and the quantity $d_i - \Gamma_1(d_i)$ is non-decreasing as d_i increases. Thus the number of edit operations is at least $4n - p + \Gamma_1(p) \geq 4n - p - q + \Gamma_1(p)$.

The other case of q is similar, as follows.

The case of $q = 2$. Here again we have two cases.

Case (a): $d_d \geq 4n - 2p + 1$. In this case, notice that the LCS after the operations between x_1 and y is at most the original $\mathsf{LCS}(x_1, y) = 2(p-1) - \Gamma_1(p-1) + 1 = 2p - 1 - \Gamma_1(p-1)$. With d_i insertions, the number of agreements can be at most $\mathsf{LCS}(x_1, y) + d_i = 2p - 1 - \Gamma_1(p-1) + d_i$, thus the Hamming distance at the end is at least $r - 2p + 1 + \Gamma_1(p-1) - d_i$. Therefore, in this case the number of edit operations is at least $d_i + d_d + r - 2p + 1 + \Gamma_1(p-1) - d_i \geq 4n - p + \Gamma_1(p-1)$, and the equality is achieved when $d_d = 4n - 2p + 1$.

Case (b): $d_d \leq 4n - 2p$. In this case, notice that all original symbols in x_1 larger than $3d_i$ (or beyond index $2d_i$ before the insertions) are guaranteed to be out of agreement. Thus the only possible original symbols in x_1 that are in agreement with y after the operations are the symbols with original index at most $2d_i$. Note that the LCS between $x[1 : 2d_i]$ and y is $2d_i - \Gamma_1(d_i)$. Thus with d_i insertions the number of agreements is at most $3d_i - \Gamma_1(d_i)$, and the Hamming distance at the end is at least $r - 3d_i + \Gamma_1(d_i)$.

Therefore the number of edit operations is at least $d_i + d_d + r - 3d_i + \Gamma_1(d_i) = r - d_i + (d_d - d_i) + \Gamma_1(d_i) = 4n - d_i + \Gamma_1(d_i)$. Now notice that $d_i = d_d + r - 4n < p - 1$ and the quantity $d_i - \Gamma_1(d_i)$ is non-decreasing as d_i increases. Thus the number of edit operations is at least $4n - (p-1) + \Gamma_1(p-1) > 4n - p + \Gamma_1(p-1)$. ◀

We can now prove a similar lemma for x_2. For any i, let $\Gamma_2(i)$ denote the number of a symbols from index $2i + 1$ to $4n$ in x_2. Note that $\Gamma_2(i)$ is equal to the number of 0's in $\beta'[i + 1 : 2n]$.

▶ **Lemma 14.** *Let $r = 3p + q$ where $0 \leq q \leq 2$, then $\mathsf{ED}(x_2, [r + 1 : 6n]) = 2n + p - q + \Gamma_2(p)$ if $q = 0, 1$ and $\mathsf{ED}(x_2, [r + 1 : 6n]) = 2n + p + \Gamma_2(p + 1)$ if $q = 2$.*

Proof. We can reduce to Lemma 13. To do this, use $6n + 1$ to minus every symbol in x_2 and in $[r + 1 : 6n]$, while keeping all the a symbols unchanged. Now, reading both strings from right to left, x_2 becomes the string $\overline{x_2} = 1, 2, \cdots, 3i - 2, 3i - 1, \cdots, 6n - 2, 6n - 1$ with some symbols of the form $3j - 1$ replaced by a's. Similarly $[r + 1 : 6n]$ becomes $[1 : 6n - r]$ where $6n - r = 3(2n - p) - q$.

If we regard $\overline{x_2}$ as x_1 as in Lemma 13 and define $\Gamma_1(i)$ as in that lemma, we can see that $\Gamma_1(i) = \Gamma_2(2n - i)$.

Now the lemma basically follows from Lemma 13. In the case of $q = 0, 1$, we have

$$\mathsf{ED}(x_2, [r+1 : 6n]) = \mathsf{ED}(\overline{x'}, [1 : 6n-r]) = 4n-(2n-p)-q+\Gamma_1(2n-p) = 2n+p-q+\Gamma_2(p).$$

In the case of $q = 2$, we have

$$\mathsf{ED}(x_2, [r+1 : 6n]) = \mathsf{ED}(\overline{x'}, [1 : 6n-r]) = 4n-(2n-p)+\Gamma_1(2n-p-1) = 2n+p+\Gamma_2(p+1).$$ ◀

We can now prove Lemma 11.

Proof of Lemma 11. We show that for any $r \in [6n]$, $\mathsf{ED}(x_1, [1 : r]) + \mathsf{ED}(x_2, [r + 1 : 6n]) \geq 7n - 2$. First we have the following claim.

▷ Claim 15. If $\mathsf{DIS}(\alpha, \beta) = 1$, then for any $i \in [2n]$, we have $\Gamma_1(i) + \Gamma_2(i) \geq n$.

To see this, note that when i is even, we have $\Gamma_1(i) = i/2$ and $\Gamma_1(i) = n - i/2$ so $\Gamma_1(i) + \Gamma_2(i) = n$. Now consider the case of i being odd and let $i = 2j - 1$ for some $j \in [2n]$. We know $\Gamma_1(i - 1) = (i - 1)/2 = j - 1$ and $\Gamma_2(i + 1) = n - (i + 1)/2 = n - j$, so we only need to look at $x_1[2i - 1, 2i]$ and $x_2[2i + 1, 2i + 2]$ and count the number of symbols a's in them. If the number of a's is at least 1, then we are done.

The only possible situation where the number of a's is 0 is that $\alpha'_i = \beta'_{i+1} = 1$ which means $\alpha_j = \beta_j = 1$ and this contradicts the fact that $\mathsf{DIS}(\alpha, \beta) = 1$.

We now have the following cases.

Case (a): $r = 3p$. In this case, by Lemma 13 and Lemma 14 we have $\mathsf{ED}(x_1, [1 : r]) = 4n - p + \Gamma_1(p)$ and $\mathsf{ED}(x_2, [r + 1 : 6n]) = 2n + p + \Gamma_2(p)$. Thus we have $\mathsf{ED}(x_1, [1 : r]) + \mathsf{ED}(x_2, [r + 1 : 6n]) = 6n + n = 7n$.

Case (b): $r = 3p - 1 = 3(p - 1) + 2$. In this case, by Lemma 13 and Lemma 14 we have $\mathsf{ED}(x_1, [1 : r]) = 4n - p - 1 + \Gamma_1(p)$ and $\mathsf{ED}(x_2, [r + 1 : 6n]) = 2n + (p - 1) + \Gamma_2(p)$, thus we have $\mathsf{ED}(x_1, [1 : r]) + \mathsf{ED}(x_2, [r + 1 : 6n]) = 6n - 2 + n = 7n - 2$.

Case (c): $r = 3p - 2 = 3(p - 1) + 1$. In this case, by Lemma 13 and Lemma 14 we have $\mathsf{ED}(x_1, [1 : r]) = 4n - p + \Gamma_1(p - 1)$ and $\mathsf{ED}(x_2, [r + 1 : 6n]) = 2n + (p - 1) - 1 + \Gamma_2(p - 1)$, thus we have $\mathsf{ED}(x_1, [1 : r]) + \mathsf{ED}(x_2, [r + 1 : 6n]) = 6n - 2 + n = 7n - 2$. ◀

We now prove Theorem 10.

Proof of Theorem 10. We begin by upper bounding $\mathsf{ED}((x_1 \circ x_2), y)$ when $\mathsf{DIS}(\alpha, \beta) = 0$.

▷ Claim 16. If $\mathsf{DIS}(\alpha, \beta) = 0$ then $\mathsf{ED}((x_1 \circ x_2), y) \leq 7n - 3$.

To see this, note that if $\mathsf{DIS}(\alpha, \beta) = 0$ then there exists a $j \in [n]$ such that $\alpha_j = \beta_j = 1$. Thus $\alpha'_{2j-1} = 1$, $\beta'_{2j-1} = 0$ and $\alpha'_{2j} = 0$, $\beta'_{2j} = 1$. Note that the number of 0's in $\alpha'[1 : 2j - 1]$ is $j - 1$ and thus $\Gamma_1(2j - 1) = j - 1$. Similarly the number of 0's in $\beta'[2j : 2n]$ is $n - j$ and thus $\Gamma_2(2j - 1) = n - j$. To transform (x_1, x_2) to y, we choose $r = 6j - 2$, transform x_1 to $y[1 : r]$, and transform x_2 to $y[r + 1 : 6n]$.

By Lemma 13 and Lemma 14 we have $\mathsf{ED}(x_1, [1 : r]) = 4n - 2j + \Gamma_1(2j - 1)$ and $\mathsf{ED}(x_2, [r + 1 : 6n]) = 2n + (2j - 1) - 1 + \Gamma_2(2j - 1)$. .Thus we have $\mathsf{ED}(x_1, [1 : r]) + \mathsf{ED}(x_2, [r + 1 : 6n]) = 6n - 2 + \Gamma_1(2j - 1) + \Gamma_2(2j - 1) = 6n - 2 + n - 1 = 7n - 3$. Therefore $\mathsf{ED}((x_1, x_2), y) \leq 7n - 3$.

Therefore, in the case of $\mathsf{DIS}(\alpha, \beta) = 1$, we have $\mathsf{ED}((x_1 \circ x_2), y) \geq 7n - 2$ while in the case of $\mathsf{DIS}(\alpha, \beta) = 0$, we have $\mathsf{ED}((x_1 \circ x_2), y) \leq 7n - 3$. Thus any protocol that solves $\mathsf{ED}_{cc}(y)$ can also solve DIS, hence the theorem follows. ◀

In the proof of Theorem 10, the two strings $x = (x_1 \circ x_2)$ and y have different lengths, however we can extend it to the case where the two strings have the same length and prove the following theorem.

▶ **Theorem 17.** *Suppose each input string to* DIS *has length* n *and let* $\Sigma = [16n] \cup \{a\}$. *Fix* $\tilde{y} = (1, 2, \cdots, 16n, a^{2n})$, *let* $\tilde{x}_1 \in \Sigma^{4n}$ *and* $\tilde{x}_2 \in \Sigma^{14n}$. *Define* $\mathsf{ED}_{cc}(\tilde{y})$ *as the two party communication problem of computing* $\mathsf{ED}((\tilde{x}_1 \circ \tilde{x}_2), \tilde{y})$. *Then* $R^{1/3}(\mathsf{ED}_{cc}(\tilde{y})) \geq R^{1/3}(\mathsf{DIS})$.

We defer the proof of Theorem 17 to the full version. From Theorem 17 we immediately have the following theorem.

▶ **Theorem 18.** *Any* R*-pass randomized algorithm in the asymmetric streaming model that computes* $\mathsf{ED}(x, y)$ *exactly between two strings* x, y *of length* n *with success probability at least* $2/3$ *must use space at least* $\Omega(n/R)$.

We can generalize the theorem to the case of deciding if $\mathsf{ED}(x, y)$ is at least a given number k. We have the following theorem. The proof is deferred to the full version.

▶ **Theorem 19** (Restatement of Theorem 1)**.** *There is a constant* $c > 1$ *such that for any* $k, n \in \mathbb{N}$ *with* $n \geq ck$, *given an alphabet* Σ, *any* R*-pass randomized algorithm in the asymmetric streaming model that decides if* $\mathsf{ED}(x, y) \geq k$ *between two strings* $x, y \in \Sigma^n$ *with success probability at least* $2/3$ *must use space at least* $\Omega(\mathsf{min}(k, |\Sigma|)/R)$.

For $0 < \varepsilon < 1$, by taking $k = 1/\varepsilon$ we also get the following corollary:

▶ **Corollary 20.** *Given an alphabet Σ, for any $0 < \varepsilon < 1$, any R-pass randomized algorithm in the asymmetric streaming model that achieves a $1+\varepsilon$ approximation of $\mathsf{ED}(x, y)$ between two strings $x, y \in \Sigma^n$ with success probability at least $2/3$ must use space at least $\Omega(\mathsf{min}(1/\varepsilon, |\Sigma|)/R)$.*

5 Lower Bounds for LCS

In this section, we study the space lower bounds for asymmetric streaming LCS.

5.1 Exact computation

5.1.1 Binary alphabet, deterministic algorithm

In this section, without loss of generality, we assume n can be diveded by 60 and let $l = \frac{n}{30} - 1$. We assume the alphabet is $\Sigma = \{a, b\}$. Consider strings x of the form

$$x = b^{10}a^{s_1}b^{10}a^{s_2}b^{10}\cdots b^{10}a^{s_l}b^{10}. \tag{1}$$

That is, x contains l blocks of consecutive a symbols. Between each block of a symbols, we insert 10 b's and we also add 10 b's to the front, and the end of x. $s_1, \ldots, s_l$ are l integers such that

$$\sum_{i=1}^{l} s_i = \frac{n}{6} + 5, \tag{2}$$

$$1 \le s_i \le 9,\ \forall i \in [l]. \tag{3}$$

Thus, the length of x is $\sum_{i=1}^{l} \frac{n}{6} + 5 + 10(l+1) = \frac{n}{2} + 5$ and it contains exactly $\frac{n}{3}$ b's.

Let S be the set of all $x \in \{a, b\}^{\frac{n}{2}+5}$ of form 1 that satisfying equations 2, 3. For each string $x \in S$, we can define a string $f(x) \in \{a, b\}^{\frac{n}{2}-5}$ as following. Assume $x = b^{10}a^{s_1}b^{10}a^{s_2}b^{10}\cdots b^{10}a^{s_l}b^{10}$, we set $f(x) = a^{s_1}b^{10}a^{s_2}b^{10}\cdots b^{10}a^{s_l}b^{10}$. That is, $f(x)$ simply removed the first 10 b's of x. We denote $\bar{S} = \{f(x) | x \in S\}$.

▷ Claim 21. $|S| = |\bar{S}| = 2^{\Omega(n)}$.

Proof. Notice that for $x^1, x^2 \in S$, if $x^1 \neq x^2$, then $f(x^1) \neq f(x^2)$. We have $|S| = |\bar{S}|$.

The size of S equals to the number of choices of l integers $s_1, s_2, \ldots, s_l$ that satisfies 2 and 3. For an lower bound of $|S|$, we can pick $\frac{n}{60}$ of the integers to be 9, and set the remaining to be 1 or 2. Thus the number of such choices is at least $\binom{l}{\frac{n}{60}} = \binom{\frac{n}{30}-1}{\frac{n}{60}} = 2^{\Omega(n)}$. ◁

Given the construction of set S, we have the following lemma. We defer the proof to the full version.

▶ **Lemma 22.** *Let $y = a^{n/3}b^{n/3}a^{n/3}$. For every $x \in S$,*

$$\mathsf{LCS}(x \circ f(x), y) = \frac{n}{2} + 5.$$

For any two distinct $x^1, x^2 \in S$,

$$\max\{\mathsf{LCS}(x^1 \circ f(x^2), y), \mathsf{LCS}(x^2 \circ f(x^1), y)\} > \frac{n}{2} + 5.$$

The above lemma implies the following lower bound.

▶ **Lemma 23.** *In the asymmetric streaming model, any deterministic protocol that computes* $\mathsf{LCS}(x, y)$ *for any* $x, y \in \{0,1\}^n$*, in* R *passes of* x *needs* $\Omega(n/R)$ *space.*

Proof. Consider a two party game where player 1 holds a string $x^1 \in S$ and player 2 holds a string $x^2 \in S$. The goal is to verify whether $x^1 = x^2$. It is known that the total communication complexity of testing the equality of two elements from set S is $\Omega(\log |S|)$, see [21] for example. We can reduce this to computing the length of LCS. To see this, we first compute $\mathsf{LCS}(x^1 \circ f(x^2), y)$ and $\mathsf{LCS}(x^2 \circ f(x^1), y)$ with $y = a^{n/3}b^{n/3}a^{n/3}$. By lemma 22, if both $\mathsf{LCS}(x^1 \circ f(x^2), y) = \mathsf{LCS}(x^2 \circ f(x), y) = \frac{n}{2} + 5$, we know $x^1 = x^2$, otherwise, $x^1 \neq x^2$. Here, y is known to both parties.

The above reduction shows the total communication complexity of this game is $\Omega(n)$ since $|S| = 2^{\Omega(n)}$. If we only allow R rounds of communication, the size of the longest message sent by the players is $\Omega(n/R)$. Thus, in the asymmetric model, any protocol that computes $\mathsf{LCS}(x, y)$ in R passes of x needs $\Omega(n/R)$ space. ◀

5.1.2 $\Omega(n)$ size alphabet, randomized algorithm

If the alphabet set Σ has size $\Omega(n)$, then we show there is a space lower bound of $\Omega(n)$ for asymmetric streaming algorithms that computes $\mathsf{LCS}(x, y)$ for $x, y \in \Sigma^n$. We have the following theorem.

▶ **Theorem 24** (Restatement of Theorem 3)**.** *There is a constant* $c > 1$ *such that for any* $k, n \in \mathbb{N}$ *with* $n > ck$*, given an alphabet* Σ*, any* R*-pass randomized algorithm in the asymmetric streaming model that decides if* $\mathsf{LCS}(x, y) \geq k$ *between two strings* $x, y \in \Sigma^n$ *with success probability at least* $2/3$ *must use at least* $\Omega\big(\mathsf{min}(k, |\Sigma|)/R\big)$ *space.*

The proof relies on a reduction from set-disjointness problem. We defer the proof of Theorem 3 to the full version.

5.2 Approximation

For deterministic approximation of LCS in the asymmetric streaming model, we have the following lower bound.

▶ **Theorem 25** (Restatement of Theorem 5)**.** *Assume* $\varepsilon > 0$*, and* $\frac{|\Sigma|^2}{\varepsilon} \leq n$ *. In the asymmetric streaming model, any deterministic protocol that computes an* $1+\varepsilon$ *approximation of* $\mathsf{LCS}(x, y)$ *for any* $x, y \in \Sigma^n$*, with constant number of passes of* x *needs* $\Omega(\frac{|\Sigma|}{\varepsilon})$ *space.*

This lower bound is achieved by combining our construction in Section 5.1.1 with the techniques from [14]. We defer the proof to the full version.

References

1 Amir Abboud, Arturs Backurs, and Virginia Vassilevska Williams. Tight hardness results for lcs and other sequence similarity measures. In *Foundations of Computer Science (FOCS), 2015 IEEE 56th Annual Symposium on*. IEEE, 2015.

2 Alexandr Andoni, T.S. Jayram, and Mihai Patrascu. Lower bounds for edit distance and product metrics via poincare type inequalities. In *Proceedings of the twenty first annual ACM-SIAM symposium on Discrete algorithms*, pages 184–192, 2010.

3 Alexandr Andoni and Robert Krauthgamer. The computational hardness of estimating edit distance. *SIAM Journal on Discrete Mathematics*, 39(6):2398–2429, 2010.

4 Alexandr Andoni, Robert Krauthgamer, and Krzysztof Onak. Polylogarithmic approximation for edit distance and the asymmetric query complexity. In *Foundations of Computer Science (FOCS), 2010 IEEE 51st Annual Symposium on*. IEEE, 2010.

5 Alexandr Andoni and Negev Shekel Nosatzki. Edit distance in near-linear time: it's a constant factor. In *Proceedings of the 61st Annual Symposium on Foundations of Computer Science (FOCS)*, 2020.

6 Arturs Backurs and Piotr Indyk. Edit distance cannot be computed in strongly subquadratic time (unless seth is false). In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing (STOC)*. IEEE, 2015.

7 Djamal Belazzougui and Qin Zhang. Edit distance: Sketching, streaming, and document exchange. In *Proceedings of the 57th IEEE Annual Symposium on Foundations of Computer Science*, pages 51–60. IEEE, 2016.

8 Joshua Brakensiek and Aviad Rubinstein. Constant-factor approximation of near-linear edit distance in near-linear time. In *Proceedings of the 52nd annual ACM symposium on Theory of computing (STOC)*, 2020.

9 Diptarka Chakraborty, Debarati Das, Elazar Goldenberg, Michal Koucky, and Michael Saks. Approximating edit distance within constant factor in truly sub-quadratic time. In *Foundations of Computer Science (FOCS), 2018 IEEE 59th Annual Symposium on*. IEEE, 2019.

10 Diptarka Chakraborty, Elazar Goldenberg, and Michal Koucký. Low distortion embedding from edit to hamming distance using coupling. In *Proceedings of the 48th IEEE Annual Annual ACM SIGACT Symposium on Theory of Computing*. ACM, 2016.

11 Diptarka Chakraborty, Elazar Goldenberg, and Michal Koucký. Streaming algorithms for computing edit distance without exploiting suffix trees. *arXiv preprint*, 2016. `arXiv:1607.03718`.

12 Kuan Cheng, Alireza Farhadi, MohammadTaghi Hajiaghayi, Zhengzhong Jin, Xin Li, Aviad Rubinstein, Saeed Seddighin, and Yu Zheng. Streaming and small space approximation algorithms for edit distance and longest common subsequence. In *International Colloquium on Automata, Languages, and Programming*. Springer, 2021.

13 Kuan Cheng, Zhengzhong Jin, Xin Li, and Yu Zheng. Space efficient deterministic approximation of string measures. *arXiv preprint*, 2020. `arXiv:2002.08498`.

14 Funda Ergun and Hossein Jowhari. On distance to monotonicity and longest increasing subsequence of a data stream. In *Proceedings of the nineteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 730–736, 2008.

15 Alireza Farhadi, MohammadTaghi Hajiaghayi, Aviad Rubinstein, and Saeed Seddighin. Streaming with oracle: New streaming algorithms for edit distance and lcs. *arXiv preprint*, 2020. `arXiv:2002.11342`.

16 Anna Gál and Parikshit Gopalan. Lower bounds on streaming algorithms for approximating the length of the longest increasing subsequence. *SIAM Journal on Computing*, 39(8):3463–3479, 2010.

17 Parikshit Gopalan, TS Jayram, Robert Krauthgamer, and Ravi Kumar. Estimating the sortedness of a data stream. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 318–327. Society for Industrial and Applied Mathematics, 2007.

18 MohammadTaghi Hajiaghayi, Masoud Seddighin, Saeed Seddighin, and Xiaorui Sun. Approximating lcs in linear time: beating the $\sqrt{n}$ barrier. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1181–1200. Society for Industrial and Applied Mathematics, 2019.

19 Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity. *Journal of Computer and System Sciences*, 63(4):512–530, 2001.

20 Michal Kouckỳ and Michael E Saks. Constant factor approximations to edit distance on far input pairs in nearly linear time. In *Proceedings of the 52nd annual ACM symposium on Theory of computing (STOC)*, 2020.

21 Eyal Kushilevitz and Noam Nisan. *Communication Complexity*. Cambridge Press, 1997.

22 David Liben-Nowell, Erik Vee, and An Zhu. Finding longest increasing and common subsequences in streaming data. In *COCOON*, 2005.

23 Aviad Rubinstein, Saeed Seddighin, Zhao Song, and Xiaorui Sun. Approximation algorithms for lcs and lis with truly improved running times. In *Foundations of Computer Science (FOCS), 2019 IEEE 60th Annual Symposium on*. IEEE, 2019.

24 Aviad Rubinstein and Zhao Song. Reducing approximate longest common subsequence to approximate edit distance. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1591–1600. SIAM, 2020.

25 Barna Saha. Fast & space-efficient approximations of language edit distance and RNA folding: An amnesic dynamic programming approach. In *FOCS*, 2017.

26 Michael Saks and C Seshadhri. Space efficient streaming algorithms for the distance to monotonicity and asymmetric edit distance. In *Proceedings of the twenty-fourth annual ACM-SIAM symposium on Discrete algorithms*, pages 1698–1709. SIAM, 2013.

27 Leonard J Schulman and David Zuckerman. Asymptotically good codes correcting insertions, deletions, and transpositions. *IEEE transactions on information theory*, 45(7):2552–2557, 1999.

28 Xiaoming Sun and David P Woodruff. The communication and streaming complexity of computing the longest common and increasing subsequences. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 336–345, 2007.

A Lower Bounds for LIS and LNS

In this section, we introduce our space lower bound for LIS and LNS. We first introduce some definition and related results regarding the multi-party communication model.

We will consider the one-way t-party communication model where t players $P_1, P_2, \ldots, P_t$ each holds input $x_1, x_2, \ldots, x_t$ respectively. The goal is to compute the function $f(x_1, x_2, \ldots, x_t)$. In the one-way communication model, each player speaks in turn and player P_i can only send message to player P_{i+1}. We sometimes consider multiple round of communication. In an R round protocol, during round $r \leq R$, each player speaks in turn P_i sends message to P_{i+1}. At the end of round $r < R$, player P_t sends a message to P_1. At the end of round R, player P_t must output the answer of the protocol.

We define the *total communication complexity* of f in the t-party one-way communication model, denoted by $CC_t^{tot}(f)$, as the minimum number of bits required to be sent by the players in every deterministic communication protocol that always outputs a correct answer. We define $CC_t^{max}(f)$, the maximum communication complexity of f, as the maximum number of bits required to be sent by some player in protocol P, where P ranges over all deterministic protocol that outputs a correct answer. We have $CC_t^{max}(f) \geq \frac{1}{tR} CC_t^{tot}(f)$ where R is the number of rounds.

Let X be a subset of U^t where U is some finite universe and t is an integer. Define the **span** of X by $\mathsf{Span}(X) = \{y \in U^t | \forall\ i \in [t],\ \exists\ x \in X \text{ s. t. } y_i = x_i\}$. The notion k-fooling set introduced in [14] is defined as following.

▶ **Definition 26** (k-fooling set). *Let $f : U^t \to \{0,1\}$ where U is some finite universe. Let $S \subseteq U^t$. For some integer k, we say S is a k-fooling set for f iff $f(x) = 0$ for each $x \in S$ and for each subset S' of S with cardinality k, the span of S' contains a member y such that $f(y) = 1$.*

We have the following.

▶ **Lemma 27** (Fact 4.1 from [14]). *Let S be a k-fooling set for f, we have $CC_t^{tot}(f) \geq \log(\frac{|S|}{k-1})$.*

We now present our lower bounds. Consider the following problem. Let $s \in \{0,1\}^t$ be a binary string of length t. For each integer $l \geq 1$, we can define a function $h^{(l)}$ whose domain is a subset of $\{0,1\}^t$. Let $\alpha \in (1/2, 1)$ be some constant. We have following definition

$$h^{(l)}(a) = \begin{cases} 1, & \text{if there are at least } l \text{ zeros between any two nonzero positions in } s. \\ 0, & \text{if } s \text{ contains at least } \alpha t \text{ nonzeros.} \end{cases} \tag{4}$$

We leave $h^{(l)}$ undefined otherwise. Let $B \in \{0,1\}^{s \times t}$ be a matrix and denote the i-th row of B by $R_i(B)$. We can define $g^{(l)}$ as the direct sum of s copies of $h^{(l)}$. Let

$$g^{(l)}(B) = h^{(l)}(R_1(B)) \vee h^{(l)}(R_2(B)) \vee \cdots \vee h^{(l)}(R_s(B)). \tag{5}$$

That is, $g^{(l)}(B) = 1$ if and only if there is some $i \in [s]$ such that $h^{(l)}(R_i(B)) = 1$.

In the following, we consider computing $h^{(l)}$ and $g^{(l)}$ in the t-party one-way communication model. When computing $h^{(l)}(a)$, player P_i holds the i-th element of $a \in \{0,1\}^t$ for $i \in [t]$.In this setting, when computing $g^{(l)}(B)$, player P_i holds the i-th column of matrix B for $i \in [t]$. In the following, we use $CC_t^{tot}(h^{(l)})$ to denote the total communication complexity of $h^{(l)}$ and respectively use $CC_t^{tot}(g^{(l)})$ to denote the total communication complexity of $g^{(l)}$. We also consider multiple rounds of communication and we denote the number of rounds by R. For a more detailed discussion of the multiparty communication model, we refer readers to the full version of this paper.

We can show the following lemma using Lovás Local Lemma.

▶ **Lemma 28.** *For any constant $l \geq 1$, there exists a constant k (depending on l), such that there is a k-fooling set for function $h^{(l)}$ of size c^t for some constant $c > 1$.*

We note that Lemma 4.2 of [14] proved a same result for the case $l = 1$. We defer the proof to the full version.

The following lemma is essentially the same as Lemma 4.3 in [14].

▶ **Lemma 29.** *Let $F \subseteq \{0,1\}^t$ be a k-fooling set for $h^{(l)}$. Then the set of all matrix $B \in \{0,1\}^{s \times t}$ such that $R_i(B) \in F$ is a k^s-fooling set for $g^{(l)}$.*

Combining Lemma 28 and Lemma 29, we have the following.

▶ **Lemma 30.** $CC_t^{max}(g^{(l)}) = \Omega(s/R)$.

Proof. By Lemma 28 and Lemma 29, there is a k^s-fooling set for function $g^{(l)}$ of size c^{ts} for some large enough constant k and some constant $c > 1$. By Lemma 27, in the t-party one-way communication model, $CC_t^{tot}(g^{(l)}) = \Omega(\log \frac{c^{ts}}{k^s - 1}) = \Omega(ts)$. Thus, we have $CC_t^{max}(g^{(l)}) \geq \frac{1}{tR} CC_t^{tot}(g^{(l)}) = \Omega(s/R)$. ◀

A.1 Lower bound for streaming LIS over small alphabet

With Lemma 30, we can show the following lower bound.

▶ **Lemma 31.** *For $x \in \Sigma^n$ with $|\Sigma| = O(\sqrt{n})$ and any constant $\varepsilon > 0$, any deterministic algorithm that makes R passes of x and outputs a $(1+\varepsilon)$-approximation of $\mathsf{LIS}(x)$ requires $\Omega(|\Sigma|/R)$ space.*

Proof sketch of Lemma 31. We assume the alphabet set $\Sigma = \{0, 1, \ldots, 2r\}$ which has size $|\Sigma| = 2r + 1$. Let c be a large constant and assume r can be divided by c for similicity. We set $s = \frac{r}{c}$ and $t = r$. Consider a matrix B of size $s \times t$. We denote the element on i-th row

and j-th column by $B_{i,j}$. ALso, we require that $B_{i,j}$ is either $(i-1)\frac{r}{c}+j$ or 0. For each row of B, say $R_i(B)$, either there are at least l 0's between any two nonzeros or it has more than αr nonzeros. We let $\tilde{B} \in \{0,1\}^{s\times r}$ be a binary matrix such that $\tilde{B}_{i,j}=1$ if $B_{i,j}\neq 0$ and $\tilde{B}_{i,j}=0$ if $B_{i,j}=0$ for $(i,j)\in[s]\times[r]$.

Without loss of generality, we can view any row or any column in B as a string. More specifically, let $R_i(B)=B_{i,1}B_{i,2}\dots B_{i,r}$ for $i\in[s]$, and $C_i(B)=B_{1,i}B_{2,i}\dots B_{s,i}$ for $i\in[r]$. We let $\sigma(B)=C_1(B)\circ C_2(B)\circ\cdots\circ C_r(B)$. Thus, $\sigma(B)$ is a string of length sr. For convenience, we denote $\sigma=\sigma(B)$. Here, we required the length of $\sigma=r^2/c\leq n$. If $|\sigma|<n$, we can pad σ with 0 symbols to make it has length n. This will not affect the length of the longest increasing subsequence of σ.

We can show that if there is some row of B containing more than αt nonzeros, then $\mathsf{LIS}(\sigma)\geq\alpha r$. If not, then $\mathsf{LIS}(\sigma(B))\leq(\frac{1}{r}+\frac{1}{c})r$.

Thus, if $g^{(l)}(\tilde{B})=0$, we have $\mathsf{LIS}(\sigma(B))\geq\alpha r$. And if $g^{(l)}(\tilde{B})=1$, $\mathsf{LIS}(\sigma(B))\leq(\frac{1}{c}+\frac{1}{l})r$. Here, c and l can be any large constant up to our choice and $\alpha\in(1/2,1)$ is fixed. For any $\varepsilon>0$, we can choose c and l such that $(1+\varepsilon)(\frac{1}{c}+\frac{1}{l})\leq\alpha$. This gives us a reduction from computing $g^{(l)}(\tilde{B})$ to compute a $(1+\varepsilon)$-approximation of $\mathsf{LIS}(\sigma(B))$.

In the t-party game for computing $g^{(l)}(\tilde{B})$, each player holds one column of $\tilde{B}$. Thus, player P_i also holds $C_i(B)$ since $C_i(B)$ is determined by $C_i(\tilde{B})$. If the t players can compute a $(1+\varepsilon)$ approximation of $\sigma(B)$ in the one-way communication model, we can distinguish the case of $g^{(l)}(\tilde{B})=0$ and $g^{(l)}(\tilde{B})=1$. Thus, any R passes deterministic streaming algorihtm that approximate LIS within a $1+\varepsilon$ factor requires at least $CC_t^{max}(g^{(l)})$. By Lemma 30, $CC_t^{max}(g^{(l)})=\Omega(s/R)=\Omega(|\Sigma|/R)$. ◀

A.2 Longest Non-decreasing Subsequence

We can proof a similar space lower bound for approximating the length of longest non-decreasing subsequence in the streaming model. We have the following two lemmas. The proof is deferred to the full version.

▶ **Lemma 32.** *For $x\in\Sigma^n$ with $|\Sigma|=O(\sqrt{n})$ and any constant $\varepsilon>0$, any deterministic algorithm that makes R passes of x and outputs a $(1+\varepsilon)$-approximation of $\mathsf{LNS}(x)$ requires $\Omega(|\Sigma|/R)$ space.*

▶ **Lemma 33.** *Let $x\in\Sigma^n$ and $\varepsilon>0$ such that $|\Sigma|^2/\varepsilon=O(n)$. Then any deterministic algorithm that makes constant pass of x and outputs a $(1+\varepsilon)$ approximation of $\mathsf{LNS}(x)$ takes $\Omega(r\log\frac{1}{\varepsilon})$ space.*

A.3 Longest Non-decreasing Subsequence with Threshold

We also consider a variant of LNS problem we call longest non-decreasing subsequence with threshold (LNST). In this problem, we are given a sequence $x\in\Sigma^n$ and a threshold $t\in[n]$, the longest non-decreasing subsequence with threshold t is the longest non-decreasing subsequence of x such that each symbol appeared in it is repeated at most t times. We denote the length of such a subsequence by $\mathsf{LNST}(x,t)$.

By combining the techniques from the previous sections, we can show Theorem 6. We also presented upper bound for LNST in Theorem 9 by giving a simple algorithm. We omit the algorithm and the formal proofs here.

B Algorithms for Edit Distance and LCS

In this section, we give an informal description of our improved algorithms for edit distance and LCS in the asymmetric streaming model. The formal proofs are deferred to the full version.

Algorithm for edit distance. Our algorithm for edit distance builds on and improves the algorithm in [15, 13]. The key idea of that algorithm is to use triangle inequality. Given a constant δ, the algorithm first divides x evenly into $b = n^\delta$ blocks. Then for each block x^i of x, the algorithm recursively finds an α-approximation of the closest substring to x^i in y. That is, the algorithm finds a substring $y[l_i : r_i]$ and a value d_i such that for any substring $y[l : r]$ of y, $\mathsf{ED}(x^i, y[l_i : r_i]) \le d_i \le \alpha\mathsf{ED}(x^i, y[l : r])$. Let $\tilde{y}$ be the concatenation of $y[l_i : r_i]$ from $i = 1$ to b. Then using triangle inequality, [15] showed that $\mathsf{ED}(y, \tilde{y}) + \sum_{i=1}^{b} d_i$ is a $2\alpha + 1$ approximation of $\mathsf{ED}(x, y)$. The $\tilde{O}(n^\delta)$ space is achieved by recursively applying this idea, which results in a $O(2^{1/\delta})$ approximation.

To further reduce the space complexity, our key observation is that, instead of dividing x into blocks of equal length, we can divide it according to the positions of the edit operations that transform x to y. More specifically, assume we are given a value k with $\mathsf{ED}(x, y) \le k \le c\mathsf{ED}(x, y)$ for some constant c, we show how to design an approximation algorithm using space $\tilde{O}(\sqrt{k})$. Towards this, we can divide x and y each into $\sqrt{k}$ blocks $x = x^1 \circ \cdots \circ x^{\sqrt{k}}$ and $y = y^1 \circ \cdots \circ y^{\sqrt{k}}$ such that $\mathsf{ED}(x^i, y^i) \le \frac{\mathsf{ED}(x,y)}{\sqrt{k}} \le \sqrt{k}$ for any $i \in [\sqrt{k}]$. However, such a partition of x and y is not known to us. Instead, we start from the first position of x and find the largest index l_1 such that $\mathsf{ED}(x[1 : l_1], y[p_1, q_1]) \le \sqrt{k}$ for some substring $y[p_1 : q_1]$ of y. To do this, we start with $l = \sqrt{k}$ and try all substrings of y with length in $[l - \sqrt{k}, l + \sqrt{k}]$. If there is some substring of y within edit distance $\sqrt{k}$ to $x[1 : l]$, we set $l_1 = l$ and store all the edit operations that transform $y[p_1 : q_1]$ to $x[1 : l_1]$ where $y[p_1 : q_1]$ is the substring closest to $x[1 : l_1]$ in edit distance. We continue doing this with $l = l + 1$ until we can not find a substring of y within edit distance $\sqrt{k}$ to $x[1 : l]$.

One problem here is that l can be much larger than $\sqrt{k}$ and we cannot store $x[1 : l]$ with $\tilde{O}(\sqrt{k})$ space. However, since we have stored some substring $y[p_1 : q_1]$ (we only need to store the two indices p_1, q_1) and the at most $\sqrt{k}$ edit operations that transform $y[p_1 : q_1]$ to $x[1 : l - 1]$, we can still query every bit of $x[1 : l]$ using $\tilde{O}(\sqrt{k})$ space.

After we find the largest possible index l_1, we store l_1, (p_1, q_1) and $d_1 = \mathsf{ED}(x[1 : l_1], y[p_1 : q_1])$. We then start from the $(l_1 + 1)$-th position of x and do the same thing again to find the largest l_2 such that there is a substring of y within edit distance $\sqrt{k}$ to $x[l_1 + 1 : l_1 + l_2]$. We continue doing this until we have processed the entire string x. Assume this gives us T pairs of indices (p_i, q_i) and integers l_i, d_i from $i = 1$ to T, we can use $O(T \log n)$ space to store them. We show by induction that $x^1 \circ \cdots \circ x^i$ is a substring of $x[1 : \sum_{j=1}^{i} l_j]$ for $i \in [T - 1]$. Recall that $x = x^1 \circ \cdots \circ x^{\sqrt{k}}$ and each $l_i > 0, i \in [T - 1]$. Thus, the process must end within $\sqrt{k}$ steps and we have $T \le \sqrt{k}$. Then, let $\tilde{y}$ be the concatenation of $y[p_i : q_i]$ from $i = 1$ to T. Using techniques developed in [15], we can show $\mathsf{ED}(y, \tilde{y}) + \sum_{i=1}^{T} d_i$ is a 3 approximation of $\mathsf{ED}(x, y)$. For any small constant $\varepsilon > 0$, we can compute a $1 + \varepsilon$ approximation of $\mathsf{ED}(y, \tilde{y})$ with $\mathsf{polylog}(n)$ space using the algorithm in [13]. This gives us a $3 + \varepsilon$ approximation algorithm with $O(\sqrt{\mathsf{ED}(x, y)}\ \mathsf{polylog}(n))$ space.

Similar to [15], we can use recursion to further reduce the space. Let δ be a small constant and a value $k = \Theta(\mathsf{ED}(x, y))$ be given as before. There is a way to partition x and y each into k^δ blocks such that $\mathsf{ED}(x^i, y^i) \le \frac{\mathsf{ED}(x,y)}{k^\delta} \le k^{1-\delta}$. Now similarly, we want to find the largest index l^0 such that there is a substring of y within edit distance $k^{1-\delta}$ to $x[1 : l^0]$. However naively this would require $\Theta(k^{1-\delta})$ space to compute the edit distance. Thus again we turn to approximation.

We introduce a recursive algorithm called FindLongestSubstring. It takes two additional parameters as inputs: an integer u and a parameter s for the amount of space we can use. It outputs a three tuple: an index l, a pair of indices (p, q) and an integer d. Let l^0 be the largest index such that there is a substring of y within edit distance u to $x[1 : l^0]$.

We show the following two properties of FindLongestSubstring: (1) $l \geq l^0$, and (2) for any substring $y[p^* : q^*]$, $\mathsf{ED}(x[1 : l], y[p : q]) \leq d \leq c(u, s)\mathsf{ED}(x[1 : l], y[p^* : q^*])$. Here, $c(u, s)$ is a function of (u, s) that measures the approximation factor. If $u \leq s$, FindLongestSubstring outputs $l = l^0$ and the substring of y that is closest to $x[1 : l]$ using $O(s \log n)$ space by doing exact computation. In this case we set $c(u, s) = 1$. Otherwise, it calls FindLongestSubstring itself up to s times with parameters u/s and s. This gives us $T \leq s$ outputs $\{l_i, (p_i, q_i), d_i\}$ for $i \in [T]$. Let $\tilde{y}$ be the concatenation of $y[p_i : q_i]$ for $i = 1$ to T. We find the pair of indices (p, q) such that $y[p : q]$ is the substring that minimizes $\mathsf{ED}(\tilde{y}, y[p : q])$. We output $l = \sum_{j=1}^{T} l_j$, (p, q), and $d = \mathsf{ED}(\tilde{y}, y[p : q]) + \sum_{i=1}^{T} d_i$. We then use induction to show property (1) and (2) hold for these outputs, where $c(u, s) = 2(c(u/s, s) + 1)$ if $u > s$ and $c(u, s) = 1$ if $u \leq s$. Thus we have $c(u, s) = 2^{O(\log_s u)}$.

This gives an $O(k^\delta/\delta \ \mathsf{polylog}(n))$ space algorithm as follows. We run algorithm FindLongestSubstring with $u = k^{1-\delta}$ and $s = k^\delta$ to find T tuples: $\{l_i, (p_i, q_i), d_i\}$. Again, let $\tilde{y}$ be the concatenation of $y[p_i : q_i]$ from $i = 1$ to T. Similar to the $O(\sqrt{k}\ \mathsf{polylog}(n))$ space algorithm, we can show $T \leq k^\delta$ and $\mathsf{ED}(y, \tilde{y}) + \sum_{i=1}^{T} d_i$ is a $2c(k^{1-\delta}, k^\delta) + 1 = 2^{O(1/\delta)}$ approximation of $\mathsf{ED}(x, y)$. Since the depth of recursion is at most $1/\delta$ and each level of recursion needs $O(k^\delta \ \mathsf{polylog}(n))$ space, FindLongestSubstring uses $O(k^\delta/\delta \ \mathsf{polylog}(n))$ space.

The two algorithms above both require a given value k. To remove this constraint, our observation is that the two previous algorithms actually only need the number k to satisfy the following relaxed condition: there is a partition of x into k^δ blocks such that for each block x^i, there is a substring of y within edit distance $k^{1-\delta}$ to x^i. Thus, when such a k is not given, we can do the following. We first set k to be a large constant k_0. While the algorithm reads x from left to right, let T' be the number of $\{l_i, (p_i, q_i), d_i\}$ we have stored so far. Each time we run FindLongestSubstring at this level, we increase T' by 1. If the current k satisfies the relaxed condition, then by a similar argument as before T' should never exceed k^δ. Thus whenever $T' = k^\delta$, we increase k by a $2^{1/\delta}$ factor. Assume that k is updated m times in total and after the i-th update, k becomes k_i. We show that $k_m = O(\mathsf{ED}(x, y))$ (but k_m may be much smaller than $\mathsf{ED}(x, y)$). To see this, suppose $k_j > 2^{1/\delta}\mathsf{ED}(x, y)$ for some $j \leq m$. Let t_j be the position of x where k_{j-1} is updated to k_j. We know it is possible to divide $x[t_j : n]$ into $\mathsf{ED}(x, y)^\delta$ blocks such that for each part, there is a substring of y within edit distance $\mathsf{ED}(x, y)^{1-\delta} \leq k_j^{1-\delta}$ to it. By property (1) and a similar argument as before, we will run FindLongestSubstring at most $\mathsf{ED}(x, y)^\delta$ times until we reach the end of x. Since $k_j^\delta - k_{j-1}^\delta > \mathsf{ED}(x, y)^\delta$, T' must be always smaller than k_j^δ and hence k_j will not be updated. Therefore we must have $j = m$. This shows $k_{m-1} \leq 2^{1/\delta}\mathsf{ED}(x, y)$ and $k_m \leq 2^{2/\delta}\mathsf{ED}(x, y)$. Running FindLongestSubstring with $k \leq k_m$ takes $O(k_m^\delta/\delta \ \mathsf{polylog}(n)) = O(\mathsf{ED}(x, y)^\delta/\delta \ \mathsf{polylog}(n))$ space and the number of intermediate results ((p_i, q_i) and d_i's) is $O(k_m^\delta) = O(\mathsf{ED}(x, y)^\delta)$. This gives us a $2^{O(1/\delta)}$ approximation algorithm with space complexity $O(\mathsf{ED}(x, y)^\delta/\delta \ \mathsf{polylog}(n))$.

Algorithm for LCS. We show that the reduction from LCS to ED discovered in [24] can work in the asymmetric streaming model with a slight modification. Combined with our algorithm for ED, this gives a n^δ space algorithm for LCS that achieves a $1/2 + \varepsilon$ approximation for binary strings. We defer the detailed analysis and proof to the full version.

C Lower Bound for ED in the Standard Streaming Model

▶ **Theorem 34.** *There exists a constant $\varepsilon > 0$ such that for strings $x, y \in \{0,1\}^n$, any deterministic R pass streaming algorithm achieving an εn additive approximation of $\mathsf{ED}(x,y)$ needs $\Omega(n/R)$ space.*

Proof. Consider an asymptotically good insertion-deletion code $C \subseteq \{0,1\}^n$ over a binary alphabet (See [27] for example). Assume C has rate α and distance β. Both α and β are some constants larger than 0, and we have $|C| = 2^{\alpha n}$. Also, for any $x, y \in C$ with $x \neq y$, we have $\mathsf{ED}(x,y) \geq \beta n$. Let $\varepsilon = \beta/2$ and consider the two party communication problem where player 1 holds $x \in C$ and player 2 holds $y \in C$. The goal is to decide whether $x = y$. Any deterministic protocol has communication complexity at least $\log |C| = \Omega(n)$. Note that any algorithm that approximates $\mathsf{ED}(x,y)$ within an εn additive error can decide whether $x = y$. Thus the theorem follows. ◀

We note that the same bound holds for Hamming distance by the same argument.