Electronic Colloquium on Computational Complexity, Report No. 197 (2025)

Optimal White-Box Adversarial Streaming Lower Bounds for Approximating LIS Length

Anna Gal
University of Texas at Austin

Gillat Kol
Princeton University

Raghuvansh R. Saxena*
Tata Institute of Fundamental Research

Huacheng Yu
Princeton University

Abstract

The space complexity of *deterministic* streaming algorithms for approximating the length of the *longest increasing subsequence* (LIS) in a string of length n has been known to be $\tilde{\Theta}(\sqrt{n})$ for almost two decades. In contrast, the space complexity of this problem for *randomized* streaming algorithms remains one of the few longstanding open problems in one-pass streaming. In fact, no better than $\Omega(\log n)$ lower bounds are known, and the best upper bounds are no better than their deterministic counterparts.

In this paper, we push the limits of our understanding of the streaming space complexity of the approximate LIS length problem by studying it in the *white-box* adversarial streaming model. This model is an intermediate model between deterministic and randomized streaming algorithms that has recently attracted attention. In the white-box model, the streaming algorithm can draw fresh randomness when processing each incoming element, but an adversary generating the stream observes all previously used randomness and adaptively chooses the subsequent elements of the stream.

We prove a tight (up to logarithmic factors) $\Omega(\sqrt{n})$ space lower bound for any white-box streaming algorithm that approximates the length of the LIS of a stream of length n to within a factor better than 1.1. Thus, for this problem, white-box algorithms offer no improvement over deterministic ones.

*We acknowledge support of the Department of Atomic Energy, Government of India, under project no. RTI4001.

ISSN 1433-8092

Contents

1 Introduction

The LIS problem. The length of the *Longest Increasing Subsequence* (LIS) is a fundamental measure of sequences that has been extensively studied, with applications in various areas, including bioinformatics, string-matching, text processing and more. See the books by Gusfield [Gus97] and Pevzner [Pev03] and the survey by Aldous and Diaconis [AD99].

Formally, given a sequence σ of length n over an ordered alphabet $[m]$, an *increasing subsequence* in σ is a subsequence $i_1 < \cdots < i_k$ such that $\sigma(i_1) < \cdots < \sigma(i_k)$. For $c \geq 1$, if the longest LIS in σ is of length ℓ, a c-approximation of the length of the LIS in σ is any value between $\frac{\ell}{c}$ and ℓ.

The streaming model. In this paper, we study the space complexity of approximating the length of the LIS in the *data stream model* of computation. In this model, the input arrives as a stream of elements, the algorithm is allowed only one or a few passes over the stream, and it should use limited memory. The streaming model has been extensively studied in recent years, driven by the need to process massive data sets arising in domains such as network traffic monitoring, database management, machine learning, social networks, and large-scale graph analysis (see the surveys by Muthukrishnan [Mut05] and Babcock et al. [BBD+02]). In these applications, the input is either stored on an external device, or obtained on the fly from some source, and the algorithm does not have sufficient working memory to store the entire data set.

LIS in the streaming model. The LIS length approximation problem is particularly motivated in the streaming model. For example, since efficient sorting is infeasible in this setting, some applications wish to estimate how close their input is to being sorted. The length of the LIS is a natural measure of this closeness.

Streaming algorithms for computing or approximating LIS length have received considerable attention. Computing the length of the LIS *exactly* via a streaming algorithm is known to require $\Omega(n)$ space, even for randomized algorithms [GJKK07, SW07]. This shows that to get sublinear space, one needs to settle for approximation. Indeed, [GJKK07] gave a *deterministic* one-pass streaming algorithm that approximates LIS length to within any multiplicative constant and uses $\mathcal{O}(\sqrt{n}\log m)$ space. A matching (up to $\log m$ factors) lower bound of $\Omega(\sqrt{n})$ was shown by [GG10, EJ15] for any alphabet size $m \geq n$.

As discussed above, the space complexity of (deterministic or randomized) streaming algorithms for computing LIS length exactly, as well as the space complexity of deterministic streaming algorithms for approximating the length of LIS, are well understood.

In contrast, as far as we know, for *randomized* streaming algorithms that $(1+\varepsilon)$-approximate the length of the LIS, the $\Omega(\log n)$ lower bound of Sun and Woodruff [SW07] is still the best known for any (arbitrarily small) constant $\varepsilon > 0$. Thus, the optimal space complexity of *randomized* streaming algorithms for approximating the length of LIS remains wide open.

The white-box adversarial streaming model. In this paper, we push the limits of our understanding of the LIS length approximation problem by considering it in the *white-box* adversarial streaming model, an intermediate model between randomized and deterministic streaming algorithms that received a lot of attention lately [ABJ+22].

In the standard data stream model, it is assumed that the entire input is fixed in advance, and a randomized streaming algorithm has to be correct on any input stream with high probability (over the algorithm's randomness). This can also be interpreted as the input is chosen by an adversary, where the adversary knows the algorithm, but does not have any information about the algorithm's internal states (memory and random coins) during the execution of the algorithm.

Recently, streaming algorithms in stronger adversarial settings received a lot of attention, where instead of fixing the entire input in advance, the adversary is allowed to choose the next element of the stream after observing parts of the algorithm's behavior up to the current step.

The *white-box adversarial streaming model*, introduced by [ABJ+22], allows the adversary to have full access to the internal states of the algorithm. More precisely, upon receiving each element, the algorithm can use fresh randomness to update its memory. However, the white-box adversary that generates the stream can observe all previously used random coins during the execution of the algorithm up to the current step (and therefore also knows the algorithm's memory content), before fixing the next element of the stream. The algorithm has to be correct with high probability over both the algorithm's and the adversary's random choices, for every white-box adversary.

The white-box model captures scenarios where algorithms lack access to a securely private source of randomness, and, as pointed out in [ABJ+22, FW23, FJW24], also some non-adversarial settings. For instance, a server may publish its initial state and users may subsequently generate their data based on this state and send it back to the server. Analogous notions of the white-box streaming model are studied in areas such as dynamic data structures, machine learning, and persistent data structures. For a broader survey of white-box models across different domains, see [ABJ+22, FW23, FJW24].

1.1 Our Result

We prove the following lower bound on the space complexity of any white-box streaming algorithm for approximating LIS length.

Theorem 1.1. *Any (randomized) white-box streaming algorithm that, with probability $\frac{2}{3}$, outputs a 1.1-approximation to the length of the LIS of a stream of length n with alphabet size $m \geq 3n$ requires $\Omega(\sqrt{n})$ space.*

The space lower bound in Theorem 1.1 is tight (up to $\log m$ factors) as a deterministic (and therefore also white-box) streaming algorithm with space complexity $O(\sqrt{n} \log m)$ is known [GJKK07]. Additionally, for alphabet size m, there are deterministic one-pass

streaming algorithms that compute the length of LIS exactly using $O(m \log m)$ space (see Section 1.2). Therefore, the space lower bound in Theorem 1.1 can only hold for large enough alphabet size m.

Our proof of Theorem 1.1 differs from the proofs of previous white-box streaming lower bounds. Known white-box lower bounds were obtained by proving one-way, two-party deterministic communication lower bounds for the corresponding communication problem, as [ABJ+22] show that such bounds imply white-box streaming lower bounds. This approach, however, cannot be used to prove Theorem 1.1 since, for every constant $\varepsilon > 0$, there is a dynamic-programming-based two-party one-way deterministic communication protocol with only $\operatorname{polylog} n$ communication that gives a $(1+\varepsilon)$-approximation of LIS length. More generally, for $k \geq 2$, there is a k-party one-way deterministic communication protocol with $k \operatorname{polylog} n$ communication [GJKK07, SW07]. This suggests that to prove Theorem 1.1 we should consider communication protocols with many parties. Unfortunately, [ABJ+22] shows that their result cannot be extended to the case of multi-party communication protocols.

Another intermediate model between deterministic and randomized streaming that attracted attention recently is the *adversarially robust* streaming model [BEJW+22]. In this model the adversary sees only part of the algorithm's internal state (see Section 1.2), and therefore any white-box algorithm is also adversarially robust. While our Theorem 1.1 shows that LIS length approximation is hard in the white-box model, the work of [BEJW+22] implies that the space complexity of LIS length approximation in the adversarially robust model is essentially the same as that of randomized algorithms. See Section 1.2 for more details. In this sense, our result proves a lower bound in the strongest studied model that does not imply a randomized lower bound.

1.2 Prior Work

LIS in other models. A classical dynamic-programming-based algorithm, called *patience sorting* [AD99], finds a longest increasing subsequence, and thus also computes the length of LIS exactly, in nearly linear $\mathcal{O}(n \log n)$ time using $\mathcal{O}(n \log m)$ space. Fredman [Fre75] proved that the $\Theta(n \log n)$ running time of patience sorting is optimal for computing the length of LIS *exactly*. Sublinear-time and sublinear-space algorithms for *approximating* the LIS length have also been extensively studied in various settings; see, e.g., [ANSS22, RSSS19, SS17, GJ21, KS21, CFH+21].

LIS in the streaming model. Patience Sorting can naturally be viewed as a one-pass $\mathcal{O}(n \log m)$ space streaming algorithm for computing the length of LIS exactly. More precisely, the space bound is $\mathcal{O}(L \log m)$ where L is the length of the LIS. Thus, the one-pass deterministic streaming complexity is $\mathcal{O}(\min(n, m) \log m)$.

Gopalan, Jayram, Kumar and Krauthgamer [GJKK07] and Sun and Woodruff [SW07] proved an $\Omega(n)$ space lower bound for computing the length of LIS exactly, even for randomized algorithms.

[GJKK07] gave a deterministic one-pass streaming algorithm to compute a $(1+\varepsilon)$-approximation of the length of LIS using space $\mathcal{O}(\sqrt{\frac{n}{\varepsilon}}\log m)$ for any $\varepsilon > 0$. Gál and Gopalan [GG10] and Ergun and Jowhari [EJ15] proved matching lower bounds showing that $\Omega(\sqrt{\frac{n}{\varepsilon}})$ space is necessary for deterministic constant-pass streaming algorithms for any $\varepsilon \geq 1/n$ and large enough alphabet size $m \geq n$. More precisely, the bound of [GG10] is of the form $\Omega\left(\frac{1}{R}\sqrt{\frac{n}{\varepsilon}}\log(\frac{2m}{n})\right)$ for R-pass streaming algorithms. Li and Zheng [LZ21] extended the deterministic lower bounds of [GG10, EJ15] to smaller alphabet sizes, proving lower bounds of the form $\Omega(\min(\sqrt{n}, m))$ for constant-pass streaming algorithms. In addition, Li and Zheng [LZ23] extended the above space lower bounds to the streaming model where the stream may be provided in other than the "standard" order of the input sequence σ. Saks and Seshadhri [SS13] gave a polylog n space randomized streaming algorithm that approximates LIS length within *additive* error δn for any constant $\delta > 0$.

Chakrabarty [Cha12] showed that the multiparty communication problems used in the *deterministic* space lower bounds for *approximating* the length of LIS in [GG10, EJ15] allow very efficient randomized protocols.

The adversarially robust streaming model. Another adversarial streaming model, closely related to the white-box model, is the *adversarially robust streaming model* [BEJW+22], where the adversary is allowed to observe outputs of the algorithm up to the current step before fixing the next element of the stream, but cannot see the random coins of the algorithm. It is known that space complexity in the "standard" randomized streaming model may be strictly smaller than in adversarially robust streaming [KMNS21], which in turn can be strictly smaller than white-box adversarial streaming space complexity [ABJ+22].

On the other hand, it was shown in [BEJW+22] that for a certain class of problems "standard" randomized streaming algorithms can be simulated by adversarially robust streaming algorithms without using much more space. In particular, for approximating LIS length this means that the space complexity of adversarially robust streaming is within $\mathcal{O}(\log n)$ factor of the space complexity of standard randomized streaming. Thus, proving space lower bounds in the adversarially robust streaming model for approximating the length of LIS would be equivalent to proving lower bounds for standard randomized streaming, up to $\mathcal{O}(\log n)$ factors.

In light of this, we focus our attention on the white-box adversarial model, as an intermediate model between randomized and deterministic streaming.

2 Proof Sketch

We overview our proof of Theorem 1.1 in this section. The proof has two main parts. First, we define a communication game LTMat (Less Than Matrix) and show a reduction saying that a streaming algorithm for approximating the length of the LIS in the input stream implies a protocol that solves the communication game. With this reduction, it suffices

to prove a lower bound on the length of the longest message in any protocol solving the communication game LTMat, which forms the second part of the proof.

2.1 The Communication Game LTMat

The communication game LTMat is parameterized by two parameters, an integer N and a real number η. In the communication game LTMat, there are $2N$ parties and the input of each party is an N bit vector. We interpret the input vectors of the odd-numbered parties as the rows of a matrix A and therefore call these parties $\mathsf{A}_1, \ldots, \mathsf{A}_N$, and interpret the rows of the even-numbered parties as the rows of a matrix B and call these parties $\mathsf{B}_1, \ldots, \mathsf{B}_N$. The goal of the parties is to distinguish between the case when $A = B$, which we call the no case, and the case where at least an η fraction of the entries in any row of A are strictly smaller than the corresponding entries in B, which we call the yes case.

We study one-way protocols for the game LTMat when the parties' inputs are determined by a white-box adversary. That is, first an all-powerful adversary selects an input for the party A_1, who then uses its randomness and the selected input to send a message to the party B_1. The adversary can see the sampled randomness (and therefore, the message) and then selects an input for the party B_1, who then uses this input together with fresh randomness to send a message to the party A_2. This continues till the last party B_N computes the output, which is either yes or no. The protocol succeeds in solving the game if whenever the selected input is in the no (respectively, yes) case, the output is no (resp. yes) with high probability.

The game LTMat can also be viewed in the white-box adversarial streaming setting, where the input vectors of the parties $\mathsf{A}_1, \mathsf{B}_1, \ldots, \mathsf{A}_N, \mathsf{B}_N$ are streamed coordinate by coordinate in that order. Note that, when viewed this way, the length of the resulting stream is $n = N^2$ and that the white-box adversary has more power as it can change the subsequent entries in the *same* vector after seeing how a streaming algorithm behaved in a prefix. In contrast, the white-box adversary in the communication version commits to the entire vector at once. This means that a lower bound on the length of the longest message in a white-box communication protocol implies a lower bound on the space needed by a white-box streaming algorithm. We use these two views interchangeably when we talk about our reduction.

2.2 Reducing LTMat to Approximating the Length of the LIS

We now show that a white-box streaming algorithm for estimating the length of the LIS implies a white-box streaming algorithm that solves the game LTMat. This means that a lower bound on the latter suffices to show Theorem 1.1.

For this, for any row i and any column j, we define a fixed offset $\Delta_{i,j}$ and consider the sequence formed by adding the offset $\Delta_{i,j}$ to the corresponding entry in both the matrices A and B. As the original entries in A and B are just bits, we can define the offsets so that regardless of the original entries, the offsetted entries in any column j are decreasing in both A and B, and any entry in any column exceeds any entry in any preceding column in both

A and B.

Observe that because of our choice of offsets, any increasing subsequence (when the input is viewed as a stream) has length at most $2N$. Indeed, as the columns are decreasing in both A and B, at most one entry can be selected from each column in each matrix, giving a total of at most $2N$ entries. Moreover, the only way to have a column j such that an entry from column j of A is in the LIS and an entry from column j of B is in the LIS is if it is the same entry (i, j) in both the matrices and $A_{i,j} < B_{i,j}$. Indeed, it must be the same entry as if not, the fact that the columns are decreasing would violate the LIS property. Moreover, given that it is the same entry, the entry for A comes before the entry for B in the stream, and therefore we must have $A_{i,j} < B_{i,j}$. Combining, we conclude that when the inputs come from the no case of LTMat, the length of the resulting LIS is N.

Consider now the case when the inputs come from the yes case of LTMat. By the definition of the yes case, we have that for any row i, it holds that $A_{i,j} < B_{i,j}$ for at least an η fraction of the columns j. This makes for a total of ηN^2 such pairs (i, j) which by the pigeonhole principle means that at least $\eta N/2$ such pairs would be on the same (parallel to the top-left to bottom-right) diagonal. These pairs would each contribute 2 entries to an increasing subsequence, which when combined with one entry from all the columns not participating in these pairs would give an increasing subsequence of length at least $N \cdot (1 + \eta/2)$.

As we have shown that inputs from the no case of LTMat give rise to sequences where the length of the LIS is N and inputs from the yes case of LTMat give rise to sequences where the length of the LIS is at least $N \cdot (1 + \eta/2)$, we get that an algorithm that gives a $(1 + \eta/2)$-approximation to the length of the LIS with high probability would solve the game LTMat with the same probability, finishing the proof of our reduction.

2.3 A Lower Bound for LTMat

We now show that any white-box communication protocol that solves the game LTMat will have at least one message with length $\Omega(N)$. To this end, note that while it may seem that being able to decide the input of the players after seeing the randomness used by the players so far gives the adversary a lot of power, this is hardly the case, as the definition of the game LTMat itself imposes a lot of restrictions on the inputs the adversary can give the parties.

For example, in the no case of LTMat, the adversary needs to ensure that $A = B$ which is possible only if the input of the party A_i is the same as the input of the party B_i for all $i \in [N]$. This means that, despite being able to see the randomness used by A_i, once the adversary has selected the input for the party A_i, he cannot use that ability to his advantage, as the input for the party B_i must be the same as A_i.

Moreover, the fact that the definition of LTMat treats all the rows $i \in [N]$ "separately" with nothing connecting the inputs for different rows (for $i \neq i'$) means that selecting the input for the party A_i and then looking at the player's randomness may not even help the adversary in selecting the inputs for the parties A_{i+1} and onwards. Thus, despite the apparent power a white-box adversary has in selecting the input after seeing the parties' randomness,

for the no case in our lower bound, we have to consider very simple adversaries. Specifically, the adversary we consider takes a (balanced) binary error correcting code C and samples an independent and uniformly random codeword as the inputs for all the odd parties and gives the same input to the corresponding even parties. Note that the reason we sample a uniformly random codeword instead of a uniformly random bit string is that any two distinct (balanced) codewords would have many coordinates where one of them is larger than the other, a property that we need in the yes case.

We now discuss how our adversary selects the inputs for the parties in the yes case. As the goal of the adversary is to make this case indistinguishable from the no case, the adversary must select the inputs from C in this case as well. This means that all the adversary has to do in order to satisfy the promise in the yes case is to ensure that for all i, the parties A_i and B_i always receive different codewords in C. Then, the fact that they are balanced would imply that the codeword for A_i is strictly smaller than the codeword for B_i in a large fraction of the coordinates. Additionally, all a protocol has to do in order to solve the game LTMat is to check whether the codewords for A_i and B_i are the same or not.

While the requirements of the yes case force the adversary to select different codewords for A_i and B_i, the adversary wants to ensure that the party B_i cannot detect that the codewords are different by looking at its input and the message it gets from A_i. These two requirements are incompatible for general protocols: The party A_i can simply send its input to the party B_i who can then detect that their inputs are different by comparing this input with the received message. However, for short protocols where the messages between the parties have length $o(N)$, it is not possible for A_i to send its entire input, and this protocol does not work. Still the question remains whether the adversary can select a different codeword for B_i without it being able to detect that its different.

Selecting the input of B_i. In order to show that the adversary can always select an input for B_i, we show a general lemma (see Lemma 5.3) saying that for any random variable X such that no value in the support is taken with probability larger than $\frac{1}{2}$, there exists a way to couple it with an identically distributed random variable X' such that in the joint distribution, we have $\mathsf{X} \neq \mathsf{X}'$ with probability 1. The assumption that no value is taken with probability larger than $\frac{1}{2}$ is tight, as if it is false, there exists x such that $\Pr(\mathsf{X} \neq x) = \Pr(\mathsf{X}' \neq x) < 1/2$. By a union bound, we get that $\Pr(\mathsf{X} = \mathsf{X}' = x) > 0$ contradicting the existence of a coupling for which $\mathsf{X} \neq \mathsf{X}'$ with probability 1.

To prove this lemma, we interpret all the values the random variable X can take as the vertices in a weighted graph and treat the value $\Pr(\mathsf{X} = x)$ as the desired degree of this vertex. Then our goal is assign weights to all the edges in the graph (no self-loops) such that the sum of the weights on the edges incident to any given vertex equals its desired degree. If we can achieve this, then, we can treat the weight on any edge (x, x') as the probability that $(\mathsf{X}, \mathsf{X}') = (x, x')$ and also the probability that $(\mathsf{X}, \mathsf{X}') = (x', x)$, there by defining a coupling between X and X'. As the sum of the weights on the edges incident to any vertex equals its desired degree, we get that X and X' are identically distributed with the desired distribution,

and as there are no self-loops, we get that $\mathsf{X} \neq \mathsf{X}'$ with probability 1.

We use an iterative "greedy" procedure that adds weight to the edge between two vertices with the highest desired degree, till either the weight adds up to the desired degree for some vertex, or conditioned on the weights already assigned, the probability of the random variable taking some value x reaches $\frac{1}{2}$. In the former case, we can use our procedure again with that vertex removed while in the latter case, we can connect the value whose probability is $\frac{1}{2}$ with all other vertices in a star configuration and the desired weights, finishing the proof.

With this lemma in hand, the adversary looks at the message sent by A_i to B_i and applies the lemma on the distribution of the input of A_i conditioned on this message. We show that, if this distribution assigns probability more than $\frac{1}{2}$ to some input, then it is as if A_i sent that input in its entirety which means the protocol must be long. Otherwise, we can use the lemma, to generate an input for B_i that is guaranteed to be different from the input of A_i and also guaranteed to have the right conditional probability given the message, as desired.

A hybrid argument. The above analysis shows that one can generate the input of any *one* party B_i in a way so that it is hard for it to distinguish between the yes case and the no case. To generate the input of all the parties, we use this argument N times, once for each $i \in [N]$. For this, we define $N+1$ adversaries $\Lambda_0, \ldots, \Lambda_N$, where for all i, the adversary Λ_i is such that the first i rows of the input matrices look like the no case and the remaining look like the yes case. Thus, any two consecutive adversaries only differ in one of the input rows meaning we can apply the argument above and show that any two consecutive adversaries are indistinguishable. Then, by a triangle inequality, we conclude that the adversaries Λ_0 and Λ_N, corresponding to the yes and no case are also indistinguishable as desired.

For this hybrid argument to work, the simplicity of our adversary above is helpful. Recall that our adversary in the no case treats each row independently, and thus, we can "stitch" together distributions for different copies without worrying about undesired correlations. Finally, for the triangle inequality to be useful, we need that the probability of distinguishing between any two consecutive adversaries is $o\left(\frac{1}{N}\right)$, which we show holds. In fact, we show that the probability is exponentially small in N.

3 Model and Preliminaries

For a set S, we use $\Delta(S)$ to denote the set of distributions supported on S.

3.1 The White-Box Streaming/One-Way Communication Model

In the white-box streaming model, an algorithm is defined by a set of states Σ, a designated initial state $\sigma_0 \in \Sigma$, a stream length[1] $n > 0$, an alphabet Γ and a transition function

[1] As our main result is a lower bound, letting the algorithm know the stream length only makes the lower bound stronger.

$\mathsf{m} : \Sigma \times \Gamma \times \{0,1\}^* \to \Sigma$ to modify its state. The algorithm has access to randomness which is included in the transition function as a bit-string $\in \{0,1\}^*$. Such an algorithm runs in the presence of an adversary Λ whose task is to generate an input stream $\in \Gamma^n$ for the algorithm.

The execution of an algorithm in the presence of the adversary takes place as follows: The algorithm starts in the initial state σ_0 proceeds in n steps, where at the beginning of step $i \in [n]$, the algorithm is in state σ_{i-1} and has sampled random strings $R_1, \ldots, R_{i-1}$. In step i, the adversary, that knows the algorithm and the random strings $R_1, \ldots, R_{i-1}$, generates an input $\gamma_i \in \Gamma$ based on its knowledge. The algorithm then samples a random string $R_i \in \{0,1\}^*$ and γ_i to update its state from σ_{i-1} to $\sigma_i = \mathsf{m}(\sigma_{i-1}, \gamma_i, R_i)$ and the random string R_i is added to the knowledge of the adversary. The state σ_n obtained after n steps is the output of the algorithm and number of bits $\log|\Sigma|$ needed to represent a state is the memory of the algorithm.

The white-box, one-way, communication model is similar, except that Σ is now the set of messages in the protocol, σ_0 is a dummy message, n is the number of parties, Γ is the input space for (all the) parties, and m is the function the parties use to compute their messages. Note that the identity of the party can be included in σ and does not need to be given explicitly to the message function. Also, note that when we reduce streaming problems to communication problems, we combine many symbols in the input stream to obtain the input of a single party in the communication model. This will decrease the number of parties n and blowup Γ and also limit the power of the adversary who now has to commit to more symbols at once. In particular, lower bounds proved for the communication model will translate to the streaming model as well.

3.2 Technical Lemmas

Lemma 3.1. *Let $\mathcal{X}$ and $\mathcal{Y}$ be finite sets. Let X, X' and Y, Y' be random variables supported on $\mathcal{X}$ and $\mathcal{Y}$ respectively. Suppose there is a function $\Lambda : \mathcal{X} \to \Delta(\mathcal{Y})$ such that for all $x \in \mathcal{X}$, we have $\mathsf{dist}(\mathsf{Y} \mid \mathsf{X} = x) = \mathsf{dist}(\mathsf{Y}' \mid \mathsf{X}' = x) = \Lambda(x)$. Let Z be a random variable independent of $(\mathsf{X}, \mathsf{X}', \mathsf{Y}, \mathsf{Y}')$. For all functions f with a finite range, we have that:*

$$\|\mathsf{dist}(\mathsf{Y}, f(\mathsf{Z}, \mathsf{X}, \mathsf{Y})) - \mathsf{dist}(\mathsf{Y}', f(\mathsf{Z}, \mathsf{X}', \mathsf{Y}'))\|_{\mathrm{TV}} \leq \|\mathsf{dist}(\mathsf{X}) - \mathsf{dist}(\mathsf{X}')\|_{\mathrm{TV}}.$$

Proof. Let $\mathcal{M}$ be the range of f. For all sets $S \subseteq \mathcal{Y} \times \mathcal{M}$, we have:

$$\begin{aligned}
&\sum_{(y,m)\in S} \Pr((\mathsf{Y}, f(\mathsf{Z}, \mathsf{X}, \mathsf{Y})) = (y,m)) - \sum_{(y,m)\in S} \Pr((\mathsf{Y}', f(\mathsf{Z}, \mathsf{X}', \mathsf{Y}')) = (y,m)) \\
&\quad = \sum_{x\in\mathcal{X}} \sum_{(y,m)\in S} \Pr((\mathsf{X}, \mathsf{Y}, f(\mathsf{Z}, \mathsf{X}, \mathsf{Y})) = (x,y,m)) \\
&\qquad\qquad - \sum_{x\in\mathcal{X}} \sum_{(y,m)\in S} \Pr((\mathsf{X}', \mathsf{Y}', f(\mathsf{Z}, \mathsf{X}', \mathsf{Y}')) = (x,y,m))
\end{aligned}$$

$$
\begin{aligned}
&= \sum_{x\in\mathcal{X}} \sum_{(y,m)\in S} \Pr((\mathsf{X},\mathsf{Y}) = (x,y)) \cdot \Pr(f(\mathsf{Z},\mathsf{X},\mathsf{Y}) = m \mid (\mathsf{X},\mathsf{Y}) = (x,y)) \\
&\qquad - \sum_{x\in\mathcal{X}} \sum_{(y,m)\in S} \Pr((\mathsf{X}',\mathsf{Y}') = (x,y)) \cdot \Pr(f(\mathsf{Z},\mathsf{X}',\mathsf{Y}') = m \mid (\mathsf{X}',\mathsf{Y}') = (x,y)) \\
&= \sum_{x\in\mathcal{X}} \sum_{(y,m)\in S} \Pr(\mathsf{X} = x) \cdot \Lambda(x)(y) \cdot \Pr(f(\mathsf{Z},x,y) = m \mid (\mathsf{X},\mathsf{Y}) = (x,y)) \\
&\qquad - \sum_{x\in\mathcal{X}} \sum_{(y,m)\in S} \Pr(\mathsf{X}' = x) \cdot \Lambda(x)(y) \cdot \Pr(f(\mathsf{Z},x,y) = m \mid (\mathsf{X}',\mathsf{Y}') = (x,y)) \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{(Definition of } \Lambda) \\
&= \sum_{x\in\mathcal{X}} \sum_{(y,m)\in S} \Pr(\mathsf{X} = x) \cdot \Lambda(x)(y) \cdot \Pr(f(\mathsf{Z},x,y) = m) \\
&\qquad - \sum_{x\in\mathcal{X}} \sum_{(y,m)\in S} \Pr(\mathsf{X}' = x) \cdot \Lambda(x)(y) \cdot \Pr(f(\mathsf{Z},x,y) = m) \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{(Independence of } \mathsf{Z} \text{ and } (\mathsf{X},\mathsf{X}',\mathsf{Y},\mathsf{Y}')) \\
&= \sum_{x\in\mathcal{X}} (\Pr(\mathsf{X} = x) - \Pr(\mathsf{X}' = x)) \cdot \sum_{(y,m)\in S} \Lambda(x)(y) \cdot \Pr(f(\mathsf{Z},x,y) = m) \\
&\leq \sum_{x\in\mathcal{X}} (\Pr(\mathsf{X} = x) - \Pr(\mathsf{X}' = x)) \cdot \mathbb{1}((\Pr(\mathsf{X} = x) \geq \Pr(\mathsf{X}' = x))) \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{(As for all } x\text{, we have } 0 \leq \textstyle\sum_{(y,m)\in S} \Lambda(x)(y) \cdot \Pr(f(\mathsf{Z},x,y) = m) \leq 1) \\
&= \|\mathsf{dist}(\mathsf{X}) - \mathsf{dist}(\mathsf{X}')\|_{\mathrm{TV}}.
\end{aligned}
$$

□

3.3 Error Correcting Codes

Lemma 3.2. *Let $n > 0$ be a large enough even integer. There exist a subset $C \subseteq \{0,1\}^n$ of size $2^{n/500}$ such that all $c \in C$ have Hamming weight[2] $n/2$ and for all $c \neq c' \in C$, the Hamming distance between c and c' is at least $2n/5$.*

Proof. Let $c_1, c_2, \ldots, c_{2^{n/500}}$ be independent and uniformly random elements of $\{0,1\}^{n/2}$. By a Chernoff bound, we have for all $i \neq i' \in \left[2^{n/500}\right]$ that:

$$\Pr(c_i \text{ and } c_{i'} \text{ have Hamming distance at most } n/5) \leq \mathrm{e}^{-n/200}.$$

By a union bound over all i, i', there exists $c_1, c_2, \ldots, c_{2^{n/500}}$ that satisfy the (fractional) Hamming distance property. For the Hamming weight property, simply concatenate all the c_i with their complement (which preserves the fractional Hamming distance). □

[2]For convenience, we ignore divisibility issues and omit floor/ceiling signs.

4 Reduction to a Communication Game

We now present a communication problem LTMat (Less Than Matrix) and show that a low-space white-box streaming algorithm that solves LIS implies a low communication white-box protocol for LTMat. The problem LTMat is parameterized by an integer N and a parameter $\eta > 0$. We write $\mathsf{LTMat}_{N,\eta}$ when we want to make these parameters explicit. The input to $\mathsf{LTMat}_{N,\eta}$ consists of two matrices $A, B \in \{0,1\}^{N \times N}$. For all $i \in [N]$, we shall use A_i and B_i to denote the i-th row of A and B respectively. The goal of $\mathsf{LTMat}_{N,\eta}$ is to distinguish between the "no" case $A = B$ and the "yes" case when for all $i \in [N]$, we have that $A_{i,j} < B_{i,j}$ for at least ηN different values of $j \in [N]$. If neither case holds, the protocol can output anything.

We consider one-way, $2N$-party communication protocols for the problem $\mathsf{LTMat}_{N,\eta}$, where there are $2N$ parties $\mathsf{A}_1, \ldots, \mathsf{A}_N$ and $\mathsf{B}_1, \ldots, \mathsf{B}_N$ and for all $i \in [N]$, the inputs for party A_i is A_i and the input for party B_i is B_i. The messages go one-way where the first message is sent from A_1 to B_1, then second message is from B_1 to A_2, and so on till the party B_N computes the output, which we treat as the last message. We now present our reduction.

Lemma 4.1. *Let $N > 0$ and $\delta, \eta \in (0,1)$. For all $S > 0$, if there exists a white-box streaming algorithm that computes a $(1 + \eta/2)$-approximation to the length of the LIS in S-space with probability δ, then there exists a one-way, $2N$-party, white-box communication protocol that solves the problem $\mathsf{LTMat}_{N,\eta}$ with probability δ where the length of every message is at most S bits.*

Proof. Fix a streaming algorithm Alg as in the lemma statement and consider the following one-way communication protocol Π. As above, the parties send messages in the order $\mathsf{A}_1, \mathsf{B}_1, \ldots, \mathsf{A}_N, \mathsf{B}_N$ where for all $i \in [N]$, the message sent by parties A_i and B_i is computed as follows: The party interprets the message received by it as the current memory state of the streaming algorithm Alg (or the initial state, if the party is A_1) and changes the entries of whatever input she gets from the white-box adversary by adding $2N \cdot (j-1) + 2 \cdot (N - i)$ to the j-th entry, for all $j \in [N]$. It then simulates the algorithm Alg on this changed input and sends the resulting memory state to the next party. In case this party is the last party B_N, then the party outputs no if Alg outputs a number at most N and outputs yes otherwise.

We claim that the protocol Π solves the problem $\mathsf{LTMat}_{N,\eta}$. For this, we show that whenever the inputs to the parties belong to the no case of $\mathsf{LTMat}_{N,\eta}$, then they simulate Alg on a sequence with LIS at most N and whenever the inputs belong to the yes case of $\mathsf{LTMat}_{N,\eta}$, then they simulate Alg on a sequence with LIS at least $N \cdot (1 + \eta/2)$. Thus, a $(1 + \eta/2)$-approximation to the length of the LIS is enough to determine the output of $\mathsf{LTMat}_{N,\eta}$.

For this, consider any execution of the protocol Π and let A and B be the matrices formed by the inputs of the $2N$ parties. If these inputs belong to the no case, *i.e.*, if $A = B$, then for all $i \in [N]$, the inputs for the parties A_i and B_i are the same. Moreover, for all $1 < i \leq N$

and all $j \in [N]$, we have that:

$$\begin{aligned} A_{i,j} + 2N \cdot (j-1) + 2 \cdot (N-i) &= A_{i,j} + 2N \cdot (j-1) + 2 \cdot (N-(i-1)) - 2 \\ &< A_{i-1,j} + 2N \cdot (j-1) + 2 \cdot (N-(i-1)). \\ &\qquad (\text{As } A, B \in \{0,1\}^{N \times N}) \end{aligned}$$

It follows that, for all $j \in [N]$, the j-th coordinate of the parties' inputs forms a non-increasing sequence which implies that the length of the LIS of the input stream to the algorithm Alg is at most N. Next, consider the case when these inputs belong to the yes case. That is, for all $i \in [N]$, we have that $A_{i,j} < B_{i,j}$ for at least ηN different values of $j \in [N]$. Thus, the number of pairs (i,j) such that $A_{i,j} < B_{i,j}$ is at least ηN^2. By the pigeonhole principle, at least $k = \eta N/2$ of these pairs have the same value of $j - i$.

Let Δ be this value of $j - i$ and let $(i_1, j_1), \ldots (i_k, j_k)$ be these values sorted in increasing order of the first coordinate. Define $j_0 = 0$ for notational convenience and consider the $N + k = N \cdot (1 + \eta/2)$ values:

$$A_{i_1,j_0+1}, \ldots, A_{i_1,j_1}, B_{i_1,j_1}, \ldots, A_{i_k,j_{k-1}+1}, \ldots, A_{i_k,j_k}, B_{i_k,j_k}, B_{i_k,j_k+1}, \ldots, B_{i_k,N}.$$

Observe that after adding our offsets, these values form a subsequence of the sequence Alg is simulated on. We finish the proof by showing that these values are increasing. Because of our choice of these pairs, this follows if we show that, for all $i \leq i' \in [N]$ and all $1 < j \leq N$, we have:

$$\begin{aligned} &\max(A_{i,j-1}, B_{i,j-1}) + 2N \cdot (j-2) + 2 \cdot (N-i) \\ &\quad = \max(A_{i,j-1}, B_{i,j-1}) + 2N \cdot (j-1) + 2 \cdot (N-i') + 2 \cdot (i'-i-N) \\ &\quad < \max(A_{i',j}, B_{i',j}) + 2N \cdot (j-1) + 2 \cdot (N-i'). \qquad (\text{As } A, B \in \{0,1\}^{N \times N}) \end{aligned}$$

□

5 The Communication Lower Bound

The goal of this section is to show the following theorem.

Theorem 5.1. *Let $N > 0$ be large enough. Any one-way, $2N$-party, white-box communication protocol that solves the problem $\mathsf{LTMat}_{N,1/5}$ with probability at least $\frac{1}{2} + \frac{1}{N}$ must have at least one message of length at least $N/1000$.*

We prove Theorem 5.1 by contradiction. Fix an $N > 0$ and suppose that there is a protocol such that every message in Π has length at most $N/1000$. Let $\mathcal{M} = \bigcup_{i \in [N/1000]} \{0,1\}^i$ be the space of all such messages and $\mathsf{C} \subseteq \{0,1\}^N$ be an error-correcting code with $2^{N/500}$ codewords where each codeword $c \in \mathsf{C}$ has Hamming weight $N/2$ and the pairwise distance

between any two codewords $c \neq c' \in \mathsf{C}$ is at least $2N/5$. Such codes are well known to exist and we also include an easy proof of their existence in Lemma 3.2.

All the white-box adversaries for the protocol Π that we consider will be such that, for all $i \in [N]$, the party A_i will get as input an independent, uniformly random element $A_i \in \mathsf{C}$. We will use A_i to denote the random variable corresponding to[3] A_i. Moreover, for all $i \in [N]$, the input for the party B_i will be sampled from a distribution that is determined by the message m that it receives from the party A_i and the input A_i sampled for the party A_i. In other words, for our purposes, a white-box adversary for the protocol Π can be defined by a vector of n functions $\Lambda = (\Lambda_i)_{i\in[N]}$ where for all $i \in [N]$, the function $\Lambda_i : \mathcal{M} \times \mathsf{C} \to \Delta(\mathsf{C})$ is a function mapping a message and an input for the party A_i to a distribution of inputs for the party B_i.

Execution in the presence of an adversary. An execution of the protocol Π in the presence of an adversary $\Lambda = (\Lambda_i)_{i\in[N]}$ as above proceeds in N steps, where for all $i \in [N]$, step i involves the party A_i sending a message to the party B_i and the party B_i sending a message to the party A_{i+1} (or computing the output, if $i = N$). In more details, for any $i \in [N]$, let $m_{\mathsf{B}_{i-1}}$ be the message sent by the party B_{i-1} to the party A_i in the previous step (or the empty message, if $i = 0$). Then, in step i, the party A_i gets an independent uniformly random element $A_i \in \mathsf{C}$. This party then uses its private randomness R_{A_i} to compute a message $m_{\mathsf{A}_i} = m_{\mathsf{A}_i}\big(R_{\mathsf{A}_i}, m_{\mathsf{B}_{i-1}}, A_i\big)$, which is sent to the party B_i. Then, the party B_i gets an input $B_i \in \mathsf{C}$ sampled from the distribution $\Lambda_i(m_{\mathsf{A}_i}, A_i)$ and uses its private randomness R_{B_i} to compute a message $m_{\mathsf{B}_i} = m_{\mathsf{B}_i}(R_{\mathsf{B}_i}, m_{\mathsf{A}_i}, B_i)$, which is sent to the party A_{i+1}, and the protocol moves to the next step.

This means that, for the protocol Π, every adversary $\Lambda = (\Lambda_i)_{i\in[N]}$ induces a (joint) distribution on the tuple $(A_i, B_i, m_{\mathsf{A}_i}, m_{\mathsf{B}_i})_{i\in[N]}$ consisting of the messages and the inputs of all the parties. We define $(\mathsf{A}_i(\Lambda), \mathsf{B}_i(\Lambda), \mathsf{m}_{\mathsf{A}_i}(\Lambda), \mathsf{m}_{\mathsf{B}_i}(\Lambda))_{i\in[N]}$ to be the random variables corresponding to this distribution. We extend this definition to the case $i = 0$ by defining those random variables to be some dummy variables that are the same for all Λ. For any random variable X, we use the notation $\mathsf{dist}(\mathsf{X})$ to denote its distribution. From the discussion above, we have that, for all adversaries $\Lambda = (\Lambda_i)_{i\in[N]}$, all $i \in [N]$, and all pairs $(m_{\mathsf{A}_i}, A_i), \big(m_{\mathsf{B}_{i-1}}, B_{i-1}\big) \in \mathcal{M} \times \mathsf{C}$, it holds that (letting $\mathfrak{U}(\mathsf{C})$ denote the uniform distribution on C):

$$\begin{aligned} \mathsf{dist}\big(\mathsf{A}_i(\Lambda) \mid \big(\mathsf{m}_{\mathsf{B}_{i-1}}(\Lambda), \mathsf{B}_{i-1}(\Lambda)\big) = \big(m_{\mathsf{B}_{i-1}}, B_{i-1}\big)\big) &= \mathfrak{U}(\mathsf{C}), \\ \mathsf{dist}(\mathsf{B}_i(\Lambda) \mid (\mathsf{m}_{\mathsf{A}_i}(\Lambda), \mathsf{A}_i(\Lambda)) = (m_{\mathsf{A}_i}, A_i)) &= \Lambda_i(m_{\mathsf{A}_i}, A_i). \end{aligned} \tag{1}$$

The adversary $\Lambda^\star$. For all $i \in [N]$, let $\Lambda^\star_i : \mathcal{M} \times \mathsf{C} \to \Delta(\mathsf{C})$ be the function that ignores its first argument and outputs a distribution that is a point mass on its second argument (equivalently, outputs the second argument with probability 1). We define the adversary

[3] This means that we use A_i to denote both the player and the random variable corresponding to the player's input.

$\Lambda^\star = (\Lambda_i^\star)_{i\in[N]}$. Observe that any input generated by the adversary $\Lambda^\star$ is supported only on inputs that belong to the no case of $\mathsf{LTMat}_{N,1/5}$. Throughout this proof, we will often abbreviate the random variables corresponding to $\Lambda^\star$ by writing $\mathsf{A}_i^\star$ instead of $\mathsf{A}_i(\Lambda^\star)$, $\mathsf{m}_{\mathsf{A}_i}^\star$ instead of $\mathsf{m}_{\mathsf{A}_i}(\Lambda^\star)$, *etc.*. We will also abbreviate events of the form $\mathsf{m}_{\mathsf{A}_i}^\star = m_{\mathsf{A}_i}$ to $m_{\mathsf{A}_i}^\star$.

Bad messages. Recall that in all adversaries we consider, for all $i \in [N]$, the party A_i gets as input an independent and uniformly random element $A_i \in \mathsf{C}$. Thus, each individual input A_i has probability $\frac{1}{|\mathsf{C}|} = 2^{-N/500}$. We define a message $m_{\mathsf{A}_i} \in \mathcal{M}$ to be "bad" if conditioning on i increases the probability of some input A_i to a value more than $1/2$. Formally, the set of all such messages is:

$$\mathsf{Bad}_i = \left\{ m_{\mathsf{A}_i} \in \mathcal{M} \mid \exists A_i \in \mathsf{C} : \Pr\left(A_i^\star \mid m_{\mathsf{A}_i}^\star\right) > \tfrac{1}{2} \right\}. \tag{2}$$

The following lemma shows that messages are unlikely to be bad.

Lemma 5.2. *For all $i \in [N]$, it holds that:*

$$\Pr\left(\mathsf{m}_{\mathsf{A}_i}^\star \in \mathsf{Bad}_i\right) < 2^{-N/2000}.$$

Proof. For all $m_{\mathsf{A}_i} \in \mathsf{Bad}_i$, we have from Equation (2) that there exists $A_i \in \mathsf{C}$ such that $\Pr\left(A_i^\star \mid m_{\mathsf{A}_i}^\star\right) > \frac{1}{2}$. As $\Pr\left(A_i^\star \mid m_{\mathsf{A}_i}^\star\right) \le \frac{\Pr\left(A_i^\star\right)}{\Pr\left(m_{\mathsf{A}_i}^\star\right)} = \frac{2^{-N/500}}{\Pr\left(m_{\mathsf{A}_i}^\star\right)}$, this means that $\Pr\left(m_{\mathsf{A}_i}^\star\right) < 2^{1-N/500}$ for all $m_{\mathsf{A}_i} \in \mathsf{Bad}_i$. As $\mathsf{Bad}_i \subseteq \mathcal{M}$, we get:

$$\Pr\left(\mathsf{m}_{\mathsf{A}_i}^\star \in \mathsf{Bad}_i\right) \le |\mathsf{Bad}_i| \cdot 2^{1-N/500} < 2^{-N/2000}.$$

□

5.1 Auxiliary Distributions

Intuitively, Equation (2) says that a message m_{A_i} is not in Bad_i only if the distribution $\mathsf{dist}\left(\mathsf{A}_i^\star \mid m_{\mathsf{A}_i}^\star\right)$ is such that the probabilities of all A_i are most $1/2$. For such distributions, we show the following lemma (stated for general functions).

Lemma 5.3. *Let Ω be a finite, non-empty set and $f : \Omega \to \mathbb{R}$ be a non-negative function such that $2 \cdot \max_\omega f(\omega) \le \sum_\omega f(\omega)$. There exists a symmetric, non-negative function $g : \Omega \times \Omega \to \mathbb{R}$ such that for all $\omega \in \Omega$, we have $g(\omega, \omega) = 0$ and $\sum_{\omega'} g(\omega, \omega') = f(\omega)$.*

Proof. Define $f(\Omega) = \sum_\omega f(\omega)$ for convenience. Proof by induction on $|\Omega|$. The base case $|\Omega| = 1$ is trivial. For the case $|\Omega| = 2$, note that $2 \cdot \max_\omega f(\omega) \le \sum_\omega f(\omega)$ implies that f takes a fixed value, say c, on all of Ω. In this case, defining $g(\omega, \omega') = c \cdot \mathbb{1}(\omega \neq \omega')$ for all $\omega, \omega' \in \Omega$ satisfies the conditions of the lemma.

We show the result for $|\Omega| \ge 3$ assuming it holds for smaller values. As $|\Omega| \ge 3$, we can sort all the elements $\omega \in \Omega$ in increasing order of $f(\omega)$ and define $f(\omega_3) \le f(\omega_2) \le f(\omega_1)$

to be the three largest values in this order (breaking ties arbitrarily). Assume for now that $f(\omega_2) + f(\omega_3) \geq \frac{f(\Omega)}{2}$. In this case, define the function g according to the following matrix:

$$\begin{array}{c} \\ \omega_1 \\ \omega_2 \\ \omega_3 \\ \vdots \\ \omega \\ \vdots \end{array} \begin{array}{c} \begin{array}{ccccccc} \omega_1 & \omega_2 & \omega_3 & \dots & \omega & \dots \end{array} \\ \begin{pmatrix} 0 & \frac{f(\Omega)}{2} - f(\omega_3) & f(\omega_1) + f(\omega_3) - \frac{f(\Omega)}{2} & \dots & 0 & \dots \\ \frac{f(\Omega)}{2} - f(\omega_3) & 0 & f(\omega_2) + f(\omega_3) - \frac{f(\Omega)}{2} & \dots & 0 & \dots \\ f(\omega_1) + f(\omega_3) - \frac{f(\Omega)}{2} & f(\omega_2) + f(\omega_3) - \frac{f(\Omega)}{2} & 0 & \dots & f(\omega) & \dots \\ \vdots & \vdots & \vdots & \ddots & \ddots & \dots \\ 0 & 0 & f(\omega) & \ddots & 0 & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix} \end{array}$$

It can be verified that g satisfies the conditions of the lemma. Now, assume that $f(\omega_2) + f(\omega_3) < \frac{f(\Omega)}{2}$. In this case, we apply the induction hypothesis on the set $\Omega' = \Omega \setminus \{\omega_2\}$ and the function $f' : \Omega' \to \mathbb{R}$ that matches f everywhere except that it is not defined on ω_2 and that $f'(\omega_1) = f(\omega_1) - f(\omega_2)$. As $f(\omega_2) + f(\omega_3) < \frac{f(\Omega)}{2}$, we have that f' and Ω' satisfy the condition in the induction hypothesis and we get a non-negative function $g' : \Omega' \times \Omega' \to \mathbb{R}$ such that for all $\omega' \in \Omega'$, we have $g'(\omega', \omega') = 0$ and $\sum_{\omega''} g'(\omega', \omega'') = f'(\omega')$. Now, define the function g to match g' when neither of its arguments equals ω_2. Otherwise, we define $g(\omega, \omega_2) = g(\omega_2, \omega) = f(\omega_2) \cdot \mathbb{1}(\omega = \omega_1)$ for all $\omega \in \Omega$. Observe that this satisfies the conditions of the lemma. □

Observe that when Lemma 5.3 is applied to a function f that is a probability distribution over Ω, then the resulting function g represents a probability distribution on $\Omega \times \Omega$.

Definition 5.4 (Auxiliary distribution)**.** *Let Ω be a finite, non-empty set and f be a probability distribution on Ω such that $\max_\omega f(\omega) \leq 1/2$. Let g be the probability distribution (on the set $\Omega \times \Omega$) obtained by applying Lemma 5.3 on f. For all $\omega \in \Omega$, we will use $f_{\mathsf{aux}(\omega)}$ to denote the distribution g conditioned on the first element being ω.*

5.2 White-Box Adversaries

We now define the other white-box adversaries we consider. For this, we define functions $(\Lambda_{i,i'})_{0 \leq i < i' \leq N}$ where each such function is a map from $\mathcal{M} \times \mathsf{C}$ to $\Delta(\mathsf{C})$. Let $0 \leq i < i' \leq N$. We define the function $\Lambda_{i,i'}$ to be such that for all $m_{\mathsf{A}_{i'}} \in \mathcal{M}$ and all $A_{i'} \in \mathsf{C}$, it outputs the distribution $\mathsf{dist}\left(\mathsf{A}^\star_{i'} \mid m^\star_{\mathsf{A}_{i'}}\right)_{\mathsf{aux}(A_{i'})}$ if $m_{\mathsf{A}_{i'}} \notin \mathsf{Bad}_{i'}$ and otherwise, it outputs a uniformly random element in $\mathsf{C} \setminus \{A_{i'}\}$. Observe that the fact that $m_{\mathsf{A}_{i'}} \notin \mathsf{Bad}_{i'}$ means that the former is well-defined. Using these, for all $0 \leq i \leq N$, we define the adversary $\Lambda_i = \left((\Lambda^\star_{i'})_{i' \in [i]}, (\Lambda_{i,i'})_{i < i' \leq N}\right)_{i \in [N]}$. Observe that $\Lambda_N = \Lambda^\star$ and that the following holds:

Observation 5.5. *Let $0 \leq i_1, i_2 < i' \leq N$. Then, it holds that $\Lambda_{i_1,i'} = \Lambda_{i_2,i'}$.*

Lemma 5.6. *For all $i < i' \in [N]$, we have that:*

$$\|\mathsf{dist}\big(\mathsf{m}_{\mathsf{B}_{i'}}(\Lambda_i), \mathsf{B}_{i'}(\Lambda_i)\big) - \mathsf{dist}\big(\mathsf{m}_{\mathsf{B}_{i'}}(\Lambda_{i-1}), \mathsf{B}_{i'}(\Lambda_{i-1})\big)\|_{\mathrm{TV}} \\ \leq \|\mathsf{dist}\big(\mathsf{m}_{\mathsf{B}_{i'-1}}(\Lambda_i), \mathsf{B}_{i'-1}(\Lambda_i)\big) - \mathsf{dist}\big(\mathsf{m}_{\mathsf{B}_{i'-1}}(\Lambda_{i-1}), \mathsf{B}_{i'-1}(\Lambda_{i-1})\big)\|_{\mathrm{TV}}.$$

Proof. To show the lemma, we show that the term

$$\|\mathsf{dist}\big(\mathsf{m}_{\mathsf{A}_{i'}}(\Lambda_i), \mathsf{A}_{i'}(\Lambda_i)\big) - \mathsf{dist}\big(\mathsf{m}_{\mathsf{A}_{i'}}(\Lambda_{i-1}), \mathsf{A}_{i'}(\Lambda_{i-1})\big)\|_{\mathrm{TV}},$$

upper bounds the expression on the left and lower bounds the expression on the right. For the former, we apply Lemma 3.1 with $\mathsf{X} = \big(\mathsf{m}_{\mathsf{A}_{i'}}(\Lambda_i), \mathsf{A}_{i'}(\Lambda_i)\big)$, $\mathsf{X}' = \big(\mathsf{m}_{\mathsf{A}_{i'}}(\Lambda_{i-1}), \mathsf{A}_{i'}(\Lambda_{i-1})\big)$ and $\mathsf{Y} = \mathsf{B}_{i'}(\Lambda_i)$, $\mathsf{Y}' = \mathsf{B}_{i'}(\Lambda_{i-1})$ and $\mathsf{Z} = \mathsf{R}_{\mathsf{B}_{i'}}$, the private randomness of the party $\mathsf{B}_{i'}$. Observe that the condition in Lemma 3.1 is satisfied as for all $\big(m_{\mathsf{A}_{i'}}, A_{i'}\big) \in \mathcal{M} \times \mathsf{C}$, we have $\mathsf{dist}\big(\mathsf{B}_{i'}(\Lambda_i) \mid \big(\mathsf{m}_{\mathsf{A}_{i'}}(\Lambda_i), \mathsf{A}_{i'}(\Lambda_i)\big) = \big(m_{\mathsf{A}_{i'}}, A_{i'}\big)\big) = \Lambda_{i,i'}\big(m_{\mathsf{A}_{i'}}, A_{i'}\big)$ and we also have $\mathsf{dist}\big(\mathsf{B}_{i'}(\Lambda_{i-1}) \mid \big(\mathsf{m}_{\mathsf{A}_{i'}}(\Lambda_{i-1}), \mathsf{A}_{i'}(\Lambda_{i-1})\big) = \big(m_{\mathsf{A}_{i'}}, A_{i'}\big)\big) = \Lambda_{i-1,i'}\big(m_{\mathsf{A}_{i'}}, A_{i'}\big)$ from Equation (1). These are the same by Observation 5.5.

For the latter, we use $\mathsf{X} = \big(\mathsf{m}_{\mathsf{B}_{i'-1}}(\Lambda_i), \mathsf{B}_{i'-1}(\Lambda_i)\big)$, $\mathsf{X}' = \big(\mathsf{m}_{\mathsf{B}_{i'-1}}(\Lambda_{i-1}), \mathsf{B}_{i'-1}(\Lambda_{i-1})\big)$ and $\mathsf{Y} = \mathsf{A}_{i'}(\Lambda_i)$, $\mathsf{Y}' = \mathsf{A}_{i'}(\Lambda_{i-1})$ and $\mathsf{Z} = \mathsf{R}_{\mathsf{A}_{i'}}$, the private randomness of the party $\mathsf{A}_{i'}$ when applying Lemma 3.1. Observe that the condition in Lemma 3.1 is satisfied as for all $\big(m_{\mathsf{A}_{i'}}, A_{i'}\big) \in \mathcal{M} \times \mathsf{C}$, we have that both distributions $\mathsf{dist}\big(\mathsf{A}_{i'}(\Lambda_i) \mid \big(\mathsf{m}_{\mathsf{B}_{i'-1}}(\Lambda_i), \mathsf{B}_{i'-1}(\Lambda_i)\big) = \big(m_{\mathsf{B}_{i'-1}}, B_{i'-1}\big)\big)$ and $\mathsf{dist}\big(\mathsf{A}_{i'}(\Lambda_{i-1}) \mid \big(\mathsf{m}_{\mathsf{B}_{i'-1}}(\Lambda_{i-1}), \mathsf{B}_{i'-1}(\Lambda_{i-1})\big) = \big(m_{\mathsf{B}_{i'-1}}, B_{i'-1}\big)\big)$ are just the uniform distribution over C, and hence identical. □

Lemma 5.7. *For all $i, i' \in [N]$, we have that:*

$$\|\mathsf{dist}\big(\mathsf{m}_{\mathsf{B}_{i'}}(\Lambda_i), \mathsf{B}_{i'}(\Lambda_i)\big) - \mathsf{dist}\big(\mathsf{m}_{\mathsf{B}_{i'}}(\Lambda_{i-1}), \mathsf{B}_{i'}(\Lambda_{i-1})\big)\|_{\mathrm{TV}} \leq \begin{cases} 0, & \text{if } i' < i \\ 2^{-N/2000}, & \text{otherwise} \end{cases}.$$

Proof. Fix an arbitrary $i \in [N]$ and observe that the case $i' < i$ is trivial. Moreover, due to Lemma 5.6, it suffices to consider the case $i' = i$. For this, we will show that:

$$\|\mathsf{dist}(\mathsf{m}_{\mathsf{A}_i}(\Lambda_i), \mathsf{B}_i(\Lambda_i)) - \mathsf{dist}(\mathsf{m}_{\mathsf{A}_i}(\Lambda_{i-1}), \mathsf{B}_i(\Lambda_{i-1}))\|_{\mathrm{TV}} \leq 2^{-N/2000}.$$

This suffices as then we use Lemma 3.1 with $\mathsf{X} = (\mathsf{m}_{\mathsf{A}_i}(\Lambda_i), \mathsf{B}_i(\Lambda_i))$, $\mathsf{X}' = (\mathsf{m}_{\mathsf{A}_i}(\Lambda_{i-1}), \mathsf{B}_i(\Lambda_{i-1}))$ and $\mathsf{Y} = \mathsf{B}_i(\Lambda_i)$, $\mathsf{Y}' = \mathsf{B}_i(\Lambda_{i-1})$ and $\mathsf{Z} = \mathsf{R}_{\mathsf{B}_i}$, the private randomness of the party B_i. The condition in Lemma 3.1 is satisfied as Y and Y' are just the second coordinates of X and X' respectively and Lemma 3.1 then finishes the proof. To see why the foregoing equation holds, note from the definition of Λ_i and Equation (1) that $\mathsf{dist}(\mathsf{m}_{\mathsf{A}_i}(\Lambda_i), \mathsf{B}_i(\Lambda_i)) = \mathsf{dist}(\mathsf{m}_{\mathsf{A}_i}(\Lambda_i), \mathsf{A}_i(\Lambda_i))$. This means that it suffices to show that:

$$\|\mathsf{dist}(\mathsf{m}_{\mathsf{A}_i}(\Lambda_i), \mathsf{A}_i(\Lambda_i)) - \mathsf{dist}(\mathsf{m}_{\mathsf{A}_i}(\Lambda_{i-1}), \mathsf{B}_i(\Lambda_{i-1}))\|_{\mathrm{TV}} \leq 2^{-N/2000}.$$

Next, note that the behavior of the adversaries Λ_i, Λ_{i-1}, and $\Lambda^\star$ is the same for parties up to and including A_i. This means that $\mathsf{dist}(\mathsf{m}_{\mathsf{A}_i}(\Lambda_i), \mathsf{A}_i(\Lambda_i)) = \mathsf{dist}(\mathsf{m}_{\mathsf{A}_i}(\Lambda_{i-1}), \mathsf{A}_i(\Lambda_{i-1})) = \mathsf{dist}\big(\mathsf{m}^\star_{\mathsf{A}_i}, \mathsf{A}^\star_i\big)$ and it suffices to show that:

$$\|\mathsf{dist}(\mathsf{m}_{\mathsf{A}_i}(\Lambda_{i-1}), \mathsf{A}_i(\Lambda_{i-1})) - \mathsf{dist}(\mathsf{m}_{\mathsf{A}_i}(\Lambda_{i-1}), \mathsf{B}_i(\Lambda_{i-1}))\|_{\mathrm{TV}} \leq 2^{-N/2000}. \tag{3}$$

To show Equation (3), note that for all $(m_{\mathsf{A}_i}, B_i) \in \mathcal{M} \times \mathsf{C}$ such that $m_{\mathsf{A}_i} \notin \mathsf{Bad}_i$, we have that:

$$\begin{aligned}
&\Pr((\mathsf{m}_{\mathsf{A}_i}(\Lambda_{i-1}), \mathsf{B}_i(\Lambda_{i-1})) = (m_{\mathsf{A}_i}, B_i)) \\
&\quad= \sum_{A_i \in \mathsf{C}} \Pr((\mathsf{m}_{\mathsf{A}_i}(\Lambda_{i-1}), \mathsf{A}_i(\Lambda_{i-1}), \mathsf{B}_i(\Lambda_{i-1})) = (m_{\mathsf{A}_i}, A_i, B_i)) \\
&\quad= \sum_{A_i \in \mathsf{C}} \Pr((\mathsf{m}_{\mathsf{A}_i}(\Lambda_{i-1}), \mathsf{A}_i(\Lambda_{i-1})) = (m_{\mathsf{A}_i}, A_i)) \cdot \Lambda_{i-1,i}(m_{\mathsf{A}_i}, A_i)(B_i) && \text{(Equation (1))} \\
&\quad= \sum_{A_i \in \mathsf{C}} \Pr((\mathsf{m}_{\mathsf{A}_i}(\Lambda_{i-1}), \mathsf{A}_i(\Lambda_{i-1})) = (m_{\mathsf{A}_i}, A_i)) \cdot \mathsf{dist}\big(\mathsf{A}^\star_i \mid m^\star_{\mathsf{A}_i}\big)_{\mathsf{aux}(A_i)}(B_i) \\
&\qquad\qquad && \text{(Definition of } \Lambda_{i-1,i} \text{ and as } m_{\mathsf{A}_i} \notin \mathsf{Bad}_i) \\
&\quad= \sum_{A_i \in \mathsf{C}} \Pr\big(m^\star_{\mathsf{A}_i}, A^\star_i\big) \cdot \mathsf{dist}\big(\mathsf{A}^\star_i \mid m^\star_{\mathsf{A}_i}\big)_{\mathsf{aux}(A_i)}(B_i) \\
&\qquad\qquad && \text{(As } \mathsf{dist}(\mathsf{m}_{\mathsf{A}_i}(\Lambda_{i-1}), \mathsf{A}_i(\Lambda_{i-1})) = \mathsf{dist}\big(\mathsf{m}^\star_{\mathsf{A}_i}, \mathsf{A}^\star_i\big)) \\
&\quad= \Pr\big(m^\star_{\mathsf{A}_i}\big) \cdot \sum_{A_i \in \mathsf{C}} \Pr\big(A^\star_i \mid m^\star_{\mathsf{A}_i}\big) \cdot \mathsf{dist}\big(\mathsf{A}^\star_i \mid m^\star_{\mathsf{A}_i}\big)_{\mathsf{aux}(A_i)}(B_i) \\
&\quad= \Pr\big(m^\star_{\mathsf{A}_i}\big) \cdot \Pr\big(\mathsf{A}^\star_i = B_i \mid m^\star_{\mathsf{A}_i}\big) && \text{(Lemma 5.3 and Definition 5.4)} \\
&\quad= \Pr\big(\big(\mathsf{m}^\star_{\mathsf{A}_i}, \mathsf{A}^\star_i\big) = (m_{\mathsf{A}_i}, B_i)\big) \\
&\quad= \Pr((\mathsf{m}_{\mathsf{A}_i}(\Lambda_{i-1}), \mathsf{A}_i(\Lambda_{i-1})) = (m_{\mathsf{A}_i}, B_i)). \\
&\qquad\qquad && \text{(As } \mathsf{dist}(\mathsf{m}_{\mathsf{A}_i}(\Lambda_{i-1}), \mathsf{A}_i(\Lambda_{i-1})) = \mathsf{dist}\big(\mathsf{m}^\star_{\mathsf{A}_i}, \mathsf{A}^\star_i\big))
\end{aligned}$$

With this equation, Equation (3) now follows as we get:

$$\begin{aligned}
&\|\mathsf{dist}(\mathsf{m}_{\mathsf{A}_i}(\Lambda_{i-1}), \mathsf{A}_i(\Lambda_{i-1})) - \mathsf{dist}(\mathsf{m}_{\mathsf{A}_i}(\Lambda_{i-1}), \mathsf{B}_i(\Lambda_{i-1}))\|_{\mathrm{TV}} \\
&\qquad\leq \Pr(\mathsf{m}_{\mathsf{A}_i}(\Lambda_{i-1}) \in \mathsf{Bad}_i) \\
&\qquad= \Pr\big(\mathsf{m}^\star_{\mathsf{A}_i} \in \mathsf{Bad}_i\big) && \text{(As } \mathsf{dist}(\mathsf{m}_{\mathsf{A}_i}(\Lambda_{i-1}), \mathsf{A}_i(\Lambda_{i-1})) = \mathsf{dist}\big(\mathsf{m}^\star_{\mathsf{A}_i}, \mathsf{A}^\star_i\big)) \\
&\qquad= 2^{-N/2000}. && \text{(Lemma 5.2)}
\end{aligned}$$

□

5.3 Finishing the Proof

Observe that the adversary $\Lambda_N = \Lambda^\star$ always satisfies $A_i = B_i$ for all $i \in [N]$ and thus, always generates inputs in the no case of $\mathsf{LTMat}_{N,1/5}$. We now show that the adversary Λ_0 always

generates inputs in the yes case of $\mathsf{LTMat}_{N,1/5}$. As $A_i, B_i \in \mathsf{C}$ and C only has codewords with Hamming weight $N/2$ and the pairwise distance between any two codewords $\in \mathsf{C}$ is at least $2N/5$, it suffices to show that the adversary Λ_0 always satisfies $A_i \neq B_i$. This is because of the definition of Λ_0 and Lemma 5.3. However, applying Lemma 5.7 with $i' = N$ and using the triangle inequality, we get that:

$$\begin{aligned}\|\mathsf{dist}(\mathsf{m}_{\mathsf{B}_N}(\Lambda_0)) - \mathsf{dist}(\mathsf{m}_{\mathsf{B}_N}(\Lambda_N))\|_{\mathrm{TV}} &\leq \|\mathsf{dist}(\mathsf{m}_{\mathsf{B}_N}(\Lambda_0), \mathsf{B}_N(\Lambda_0)) - \mathsf{dist}(\mathsf{m}_{\mathsf{B}_N}(\Lambda_N), \mathsf{B}_N(\Lambda_N))\|_{\mathrm{TV}} \\ &\leq 2^{-N/10000}.\end{aligned}$$

As N is large enough, it follows that either the probability that the last message (which is the output) of the protocol Π is yes when the adversary is Λ_0 is at most $\frac{1}{2} + \frac{1}{N}$ or the probability that the last message of the protocol Π is yes when the adversary is Λ_N is at least $\frac{1}{2}$. Theorem 5.1 follows straightforwardly in either case.

Finishing the proof of Theorem 1.1. Theorem 1.1 immediately follows from Lemma 4.1 and Theorem 5.1.

References

[ABJ+22] Miklós Ajtai, Vladimir Braverman, T. S. Jayram, Sandeep Silwal, Alec Sun, David P. Woodruff, and Samson Zhou. The white-box adversarial data stream model. In *International Conference on Management of Data (PODS)*, pages 15–27, 2022. 2, 3, 4

[AD99] David Aldous and Persi Diakonis. Longest increasing subsequences: From patience sorting to the baik–deift–johansson theorem. *Bull. Amer. Math. Soc. (N. S.)*, 36:413–432, 1999. 1, 3

[ANSS22] Alexandr Andoni, Negev Shekel Nosatzki, Sandip Sinha, and Clifford Stein. Estimating the longest increasing subsequence in nearly optimal time. In *63rd IEEE Annual Symposium on Foundations of Computer Science, FOCS*, pages 708–719, 2022. 3

[BBD+02] Brian Babcock, Shivnath Babu, Mayur Datar, Rajeev Motwani, and Jennifer Widom. Models and issues in data stream systems. In *Proceedings of the Twenty-first ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 1–16. ACM, 2002. 1

[BEJW+22] Omri Ben-Eliezer, Rajesh Jayaram, David P. Woodruff, , and Eylon Yogev. A framework for adversarially robust streaming algorithms. *Journal of the ACM*, 69(2), 2022. 3, 4

[CFH+21] 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 *48th International Colloquium on Automata, Languages, and Programming, ICALP*, volume 198 of *LIPIcs*, pages 54:1–54:20, 2021. 3

[Cha12] Amit Chakrabarti. A note on randomized streaming space bounds for the longest increasing subsequence problem. *Information Processing Letters*, 2012. 4

[EJ15] Funda Ergün and Hossein Jowhari. On the monotonicity of a data stream. *Comb.*, 35(6):641–653, 2015. 1, 4

[FJW24] Ying Feng, Aayush Jain, and David P. Woodruff. Fast white-box adversarial streaming without a random oracle. In *Forty-first International Conference on Machine Learning, ICML*, 2024. 2

[Fre75] Michael L. Fredman. On computing the length of longest increasing subsequences. *Discrete Math.*, 11(1):29–35, January 1975. 3

[FW23] Ying Feng and David P. Woodruff. Improved algorithms for white-box adversarial streams. In *International Conference on Machine Learning (ICML)*, volume 202, pages 9962–9975, 2023. 2

[GG10] Anna Gál and Parikshit Gopalan. Lower bounds on streaming algorithms for approximating the length of the longest increasing subsequence. *SIAM Journal on Computing*, 2010. 1, 4

[GJ21] Paweł Gawrychowski and Wojciech Janczewski. Fully dynamic approximation of lis in polylogarithmic time. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, page 654–667, 2021. 3

[GJKK07] Parikshit Gopalan, T. S. 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, SODA*, pages 318–327. SIAM, 2007. 1, 2, 3, 4

[Gus97] Dan Gusfield. *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology*. Cambridge University Press, Cambridge, MA, 1997. 1

[KMNS21] Haim Kaplan, Yishay Mansour, Kobbi Nissim, and Uri Stemmer. Separating adaptive streaming from oblivious streaming using the bounded storage model. In *Advances in Cryptology - 41st Annual International Cryptology Conference,*

CRYPTO Proceedings, Part III, volume 12827 of *Lecture Notes in Computer Science*, pages 94–121, 2021. 4

[KS21] Tomasz Kociumaka and Saeed Seddighin. Improved dynamic algorithms for longest increasing subsequence. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, page 640–653, 2021. 3

[LZ21] Xin Li and Yu Zheng. Lower bounds and improved algorithms for asymmetric streaming edit distance and longest common subsequence. In *41st IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS*, volume 213 of *LIPIcs*, pages 27:1–27:23, 2021. 4

[LZ23] Xin Li and Yu Zheng. Streaming and query once space complexity of longest increasing subsequence. In *Computing and Combinatorics: 29th International Conference, COCOON*, page 29–60, 2023. 4

[Mut05] S. Muthukrishnan. *Data Streams: Algorithms and Applications*. Found. Trends Theor. Comput. Sci., Now Publisher, Hanover, MA, 2005. 1

[Pev03] Pavel Pevzner. *Computational Molecular Biology: An Algorithmic Approach*. MIT Press, Cambridge, MA, 2003. 1

[RSSS19] Aviad Rubinstein, Saeed Seddighin, Zhao Song, and Xiaorui Sun. Approximation algorithms for lcs and lis with truly improved running times. In *Symposium on Foundations of Computer Science (FOCS)*, 2019. 3

[SS13] Michael E. 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, SODA*, pages 1698–1709, 2013. 4

[SS17] Michael E. Saks and C. Seshadhri. Estimating the longest increasing sequence in polylogarithmic time. *SIAM J. Comput.*, 46(2):774–823, 2017. 3

[SW07] Xiaoming Sun and David P. Woodruff. The communication and streaming complexity of computing the longest common and increasing subsequences. In *Symposium on Discrete Algorithms (SODA)*, 2007. 1, 3

ECCC ISSN 1433-8092

https://eccc.weizmann.ac.il