The Communication and Streaming Complexity of Computing the Longest Common and Increasing Subsequences

Xiaoming Sun* David P. Woodruff†

Abstract

We consider the communication complexity of finding the *longest increasing subsequence* (LIS) of a string shared between two parties. We prove tight bounds for the space complexity of randomized one-pass streaming algorithms for this problem. Our bounds are parameterized in terms of the LIS of the inputs. This resolves an open question in [19]. We also give the first bounds for approximating the LIS and its length.

Next, we consider the communication complexity of finding the *longest common subsequece* (LCS) of two strings held by different parties, as well as the problem of approximating its length. We improve the existing lower bounds for these problems, even in the most difficult case when both parties have a permutation of N symbols. Our results yield tight space bounds for multipass deterministic streaming algorithms. For randomized mutlipass algorithms, our bounds are tight up to a logarithmic factor.

1 Introduction

Finding the longest increasing subsequence (LIS) and the longest common subsequence (LCS) of integer sequences are well-studied problems in theoretical computer science with applications in bioinformatics [3, 9, 27], clustering [6], physics [8], and string-matching [24]. For details on these applications, see [19] for an overview. The LIS/LCS problems and their many variants have also been extensively studied by the mathematics community [1, 4, 5, 7, 8, 14, 15, 25].

To formally define the problems, consider a sequence $\mathcal{S} = x_1, \ldots, x_n$ of elements of a totally-ordered finite alphabet Σ. A *subsequence* of $\mathcal{S}$ is a sequence $x_{i_1}, \ldots, x_{i_k}$ with $i_1 < i_2 < \cdots < i_k$. We say that the subsequence is *increasing*[1] if $x_{i_1} \leq x_{i_2} \leq \cdots \leq x_{i_k}$. The *longest increasing subsequence* problem LIS is to find an increasing subsequence of maximum length, denoted $LIS(\mathcal{S})$. The problem LIS-length is to output the length of such a subequence. The problem LIS-ϵ-apx for $\epsilon < 1$ is to output an increasing subsequence of size at least $(1-\epsilon)|LIS(\mathcal{S})|$, and the problem LIS-length-ϵ-apx is to output a number x for which $(1-\epsilon)|LIS(\mathcal{S})| \leq x \leq |LIS(\mathcal{S})|$.

If $\mathcal{S}$ and $\mathcal{T}$ are two sequences, the *longest common subsequence* problem LCS is to find a subsequence of both $\mathcal{S}$ and $\mathcal{T}$ of maximum size, denoted $LCS(\mathcal{S}, \mathcal{T})$. LCS-length is the problem of outputting the length of such a subsequence. LCS-k-decision is the decision problem of determining if $LCS(\mathcal{S}, \mathcal{T})$ has length at least k. LCS-ρ-approx for $\rho > 1$ is the promise problem of determining if $|LCS(\mathcal{S}, \mathcal{T})| \geq \rho^2$ or $|LCS(\mathcal{S}, \mathcal{T})| \leq \rho$. Note that this becomes trivial for $\rho^2 > n$.

For a survey of algorithmic results for some of these problems, see [11]. There is an easy dynamic programming algorithm which solves LIS and LCS in $O(n^2)$ time, and an early, very efficient algorithm for LIS due to Fredman [10] running in time $O(n \log n)$.

Despite the apparent efficiency of these algorithms, on large data sets these algorithms may no longer be practical. Allowing the algorithm linear space is infeasible if the input size is a few terabytes. In internet applications, routers with only limited memory need to analyze high-speed data passing through them. In these scenarios, there is not enough space to store the input. These complications have motivated a new paradigm for studying algorithmic efficiency - the *data-stream model* [2, 13, 22]. In this model, algorithms are only given a few passes over the input data, which is arranged in adversarial order, and must use limited space to compute statistics of interest. As noted in [19], the LIS and LCS are fundamentally different from many previous problems studied in the streaming model since they depend on the order in which data arrives.

Liben-Nowell, Vee, and Zhu [19] initiate the study of the efficiency of streaming algorithms for the problems above. Let k be the length of the LIS of a string given in the data-stream model. The authors give a 1-pass $O(k \log |\Sigma|)$-space algorithm for LIS-length,

*Tsinghua University, xiaomings@tsinghua.edu.cn. Supported by the National Natural Science Foundation of China Grant 60603005.

†M.I.T., dpwood@mit.edu. Supported by Andrew C. Yao and by an NDSEG fellowship. Most of the work was done while the author was visiting Tsinghua University.

[1] Technically the sequence is non-decreasing, but we adopt the definitions in [19] for consistency.

and an $\lceil \log(1+1/\epsilon) \rceil$-pass algorithm for LIS using $O(k^{1+\epsilon} \log |\Sigma|)$ space for any $\epsilon > 0$. For $\epsilon = 1$ this gives a 1-pass $O(k^2 \log |\Sigma|)$ space algorithm for LIS. They also prove an $\Omega(k)$ space lower bound for these problems for any algorithm making a constant number of passes. The main question left open here is the quadratic gap for 1-pass algorithms for solving LIS. Note that 1-pass algorithms are particularly important in internet applications since once data has passed it cannot return. The authors also leave open and suggest the study of LIS-ϵ-apx and LIS-length-ϵ-apx.

For LCS, the same authors show that for general alphabets Σ, even for randomized multipass algorithms, even for constant-factor approximations, and even for LCS-length, there is an $\Omega(N)$ lower bound on the space required. Next, the authors consider the possibly more promising case when both strings are permutations of $[N] = \{1, 2, \ldots, N\}$. For LCS-$k$-decision, the authors show an $\Omega(k)$ lower bound for randomized multipass algorithms. We note that for two random permutations of $[N]$, the expected length of the LCS is $\Theta(\sqrt{N})$ [5], and the distribution is tightly concentrated about its expectation. Thus, since the lower bound of [19] is only $\Omega(k)$, it may be the case that for average-case permutations of $[N]$, or at least whenever the LCS has length $O(\sqrt{N})$, there is an $O(\sqrt{N})$-space upper bound.

Our Contributions: In this paper, we answer the questions raised above. For LIS we improve the previous bounds [19] of $\Omega(k)$ and $O(k^2 \log |\Sigma|)$ to a tight $\Theta(k^2 \log |\Sigma|/k)$ for 1-pass randomized streaming algorithms, provided that $|\Sigma| = \Omega(k^{2+\delta})$ for any constant $\delta > 0$. Our techniques also work for other choices of Σ. When $k \leq |\Sigma| = O(k^2)$, our lower bound becomes $\Omega(|\Sigma|)$, and for smaller $|\Sigma|$ becomes $\Theta(|\Sigma|^2 \log k/|\Sigma|)$.

For LIS-length, we show a tight bound of $\Theta(k \log |\Sigma|/k)$ for deterministic constant-pass algorithms and 1-pass randomized algorithms when $|\Sigma| = \Omega(k)$. For constant-pass randomized algorithms, we show a lower bound of $\Omega(k)$. We note that the bounds of [19] for LIS-length only hold assuming that $|\Sigma| = \Omega(k^2)$, whereas all of our bounds hold for any $|\Sigma| = \Omega(k)$. Finally, when $|\Sigma| = O(k)$, we show tight bounds of $\Theta(|\Sigma| \log k/|\Sigma|)$ for LIS-length.

We also initiate the study of LIS-length-ϵ-apx and LIS-ϵ-apx. If k denotes the output of LIS-length, in the two-party setting we show tight bounds of $\Theta((1/\epsilon) \log(\epsilon |\Sigma|))$ and $\Theta((k/\epsilon) \log(\epsilon |\Sigma|))$ for multi-round deterministic and 1-round randomized protocols.

Next, for LCS-k-decision, and thus for LCS-length and LCS, even when both strings are permutations of $[N]$, we show that for every $3 \leq k \leq N/2$, the randomized multi-round communication complexity of this problem is $\Omega(N)$. This bounds the space for constant pass randomized streaming algorithms, improving the previous bound of $\Omega(k)$ in [19]. Our restriction on k is somewhat necessary: for $k = 2$, the problem reduces to a randomized equality test, which has low communication complexity [18]. Also, for large k, we give a 1-pass randomized upper bound of $O((N-k) \log N)$, which is useful for very similar strings. We note that for deterministic streaming algorithms we can improve our lower bound to $\Omega(N \log N)$.

For LCS-ρ-approx we show that for deterministic constant-pass streaming algorithms, even when both strings are permutations of $[N]$, the space is $\Omega(N)$. This improves the $\Omega(N/\rho^2)$ bound of [19], although ours only holds for deterministic algorithms. We also give an $O(\log N + \frac{N}{\rho} \log \frac{N}{\rho})$ 1-pass randomized algorithm.

Related Work: Independently of our work, and to appear in the same conference, Gopalan *et al* [12] obtained the same lower bound for the randomized multi-round complexity of LIS-length (our Section 4.3) when the alphabet size $|\Sigma| = \Omega(k)$, thus also improving a bound of [19], which held only for $|\Sigma| = \Omega(k^2)$. Moreover, they have the same two-party upper bound for computing LIS-length-ϵ-apx (our Section 9.1), but were cleverly able to extend it to a deterministic 1-pass streaming algorithm with $O(\sqrt{N/\epsilon} \log |\Sigma|)$ space. They also prove a lower bound of $\Omega(\sqrt{N})$ for a natural class of algorithms for this problem. The best unconditional lower bound still appears to be our $\Omega(\frac{1}{\epsilon} \log(\epsilon |\Sigma|))$.

Although similar, the focus of their paper is different from ours. We look at the complexity of outputting the LIS (Section 3), rather than just its length, as well as improving the bounds in [19] for many problems related to LCS (Section 5). Their focus is on approximating the *distance to monotonicity* (our $n - \mathsf{LIS}(a, b)$). Also, we want instance-dependent bounds, e.g., bounds that depend on the size of the LIS, or the similarity of strings for the LCS (e.g., Section 5.2).

Techniques: We consider the following two-party protocols. For LIS and LIS-length, Alice is given $a \in \Sigma^n$, Bob is given $b \in \Sigma^n$, and the goal is to output $\mathsf{LIS}(a \circ b)$ or $\mathsf{LIS\text{-}length}(a \circ b)$. For LCS and related problems, our reductions work by giving Alice $a \in \Sigma^n$, Bob $b \in \Sigma^n$, and the goal is to output $\mathsf{LCS}(a, b)$. As in [19], we especially focus on proving lower bounds when a, b are permutations of $[N]$. We devise various novel reductions from known problems of high communication complexity, and use tools such as the *round-elimination lemma* [21], and the *rank method* [18] (see Section 2).

Roadmap: In Section 2, we give background. In Section 3, we lower bound the space of 1-pass streaming algorithms computing LIS. Our improved upper bound for LIS is in Appendix 6. In Section 4, we lower bound the two-party communication of LIS-length assuming

$|\Sigma| = \Omega(k)$, as well as give a better streaming algorithm for it. The case when $|\Sigma| = O(k)$ is handled in Appendix 7, a randomized multi-round upper bound in the two-party setting for LIS-length is in Appendix 8, and the extension of our results to the approximation versions of LIS and LIS-length are in Appendix 9. In Section 5 we give bounds for LCS, LCS-length, LCS-k-decision, and LCS-ρ-approx.

2 Preliminaries

We review some definitions and facts from communication complexity. Due to space constraints, we refer the reader to [18] for more details. Consider a function $f : \mathcal{X} \times \mathcal{Y} \to \{0,1\}$ whose first input is held by a party Alice, and second input by a party Bob. We use $D(f)$ to denote the deterministic multi-round communication complexity of f, that is, the maximum number of bits transmitted between Alice and Bob, over all pairs of possible inputs, in the best deterministic protocol for computing f. Similarly, we define $R_\delta(f)$ to be the randomized multi-round comunication complexity with 2-sided error at most δ, and $R_\delta^{1-way}(f)$ to be the randomized communication complexity with 2-sided error at most δ in which only a single message is sent from Alice to Bob. So, $R_\delta(f) \le D(f)$ and $R_\delta(f) \le R_\delta^{1-way}(f)$.

The *communication matrix* M_f associated with f is the $|\mathcal{X}| \times |\mathcal{Y}|$ matrix with (x, y)th entry equal to $f(x, y)$. For our deterministic multi-round lower bounds, we use the *rank method* which states that $D(f) \ge \log(rank(M_f))$. For bounding $R_\delta^{1-way}(f)$, we need the following definition.

Definition 2.1. *([21]) Let $f : \mathcal{X} \times \mathcal{Y} \to \{0,1\}$ be a function, and let $f^{(k)}$ be the following variation of f: Alice has k inputs $x_1, \ldots, x_k$ and Bob has inputs y, i, and $x_1, \ldots, x_{i-1}$ (note the overlap of Alice and Bob inputs). On these inputs $f^{(k)}$ evaluates to $f(x_i, y)$.*

To bound $R_\delta^{1-way}(f)$, one idea is to express f in the form $g^{(k)}$ for some g and some k. If Alice does not send many bits in the first round, her message is likely not to reveal anything useful about x_i for the $i \in [k]$ given to Bob. Thus, we can eliminate the first round of the protocol. We are then left with no rounds, and are likely to have a contradiction since non-trivial functions require interaction. The following is the *round-elimination lemma* which formalizes this.

Theorem 2.1. *([21]) If there is a one-round protocol with error probability δ for $f^{(k)}$ in which Alice sends a message of $\le a$ bits to Bob, then there is a zero-round protocol for f with error probability $\delta + O(\sqrt{a/k})$.*

Sometimes we will apply this theorem directly, and other times we will reduce from the indexing function $IND : \{0,1\}^n \times [n] \to \{0,1\}$, where $IND(x, i) = x_i$. The following is well-known (see [17]).

Lemma 2.1. $R_{1/3}^{1-way}(IND) = \Omega(n)$.

To bound $R_\delta(f)$, we use the disjointness function $DIS : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$, where $DIS(x, y) = 0$ if and only if there is some $i \in [n]$ for which $x_i = y_i = 1$. The following is a celebrated theorem of Kalyanasundaram and Schnitger, [16] as well as of Razborov [23]:

Theorem 2.2. $R_{1/3}(DIS) = \Omega(n)$.

Suppose a is Alice's input and b is Bob's input, and we have a function $f(a, b)$. We can derive lower bounds for computing $f(a, b)$ in the streaming model by considering the stream $a \circ b$. Any r-pass streaming algorithm A yields a $(2r-1)$-round communication protocol for f in the following way. Alice computes $A(a)$ and sends the state of the algorithm to Bob, who compute $A(a \circ b)$. This corresponds to the first pass of A on the stream $a \circ b$. Bob sends the state of A back to Alice, who continues the execution of A (the second pass) on the stream $a \circ b$. In the last pass Bob outputs the answer. The communication is $2r-1$ times the space of the streaming algorithm. Thus, for constant r, the space complexity must be at least the communication complexity of the function f, up to a constant factor.

3 LIS

We consider lower bounds for LIS. Alice is given $a \in \Sigma^n$, Bob $b \in \Sigma^n$, and their goal is to compute $\mathsf{LIS}(a \circ b)$. Assume $|\Sigma| = \Omega(k^{2+\delta})$. Identify Σ with the first $|\Sigma|$ non-negative integers. Put $M = \Theta(k^{1+\delta})$, and assume M/k is a power of 2. We construct a as follows:

1. For $0 < i < k$, choose a sequence $a_i = (a_{i,1}, \ldots, a_{i,i})$ of i integers as follows: for $1 \le j \le i$, $a_{i,j}$ is an integer subject to
$$a_{i,j} \quad > \quad 2M(i-1) + \frac{(j-1)M}{k}, \text{ and}$$
$$a_{i,j} \quad \le \quad 2M(i-1) + \frac{(j-1)M}{k} + \frac{M}{k}.$$
2. Set $a = a_{k-1} \circ a_{k-2} \circ a_{k-3} \circ \cdots \circ a_1$.

We construct b as follows:

1. Choose an i, $0 < i < k$.
2. b is the concatenation of 0^{i-1} with $2Mi - k + i + 1,\ 2Mi - k + i + 2,\ \ldots,\ 2Mi$.

Lemma 3.1. *$\mathsf{LIS}(a \circ b)$ is a_i concatenated with $2Mi - k + i + 1,\ 2Mi - k + i + 2, \ldots, 2Mi$.*

Proof. Let $s = a_i \circ 2Mi - k + i + 1,\ 2Mi - k + i + 2,\ \ldots,\ 2Mi$. We show s is increasing and of length k. For this, it suffices to show that the last element of a_i is at most $2Mi - k + i + 1$. But the last element of a_i is at most $2M(i-1) + M$. Moreover, $2Mi - k + i + 1 \geq 2Mi - k \geq 2M(i-1) + M + k^{1+\delta} - k \geq 2M(i-1) + M$, as needed (we have used that $M \geq k^{1+\delta}$).

We show that any other sequence t of $a \circ b$ of length at least k is not increasing. Suppose t intersects a_j for some j. Let $x \in a_j \cap t$. Then t cannot intersect $a_{j'}$ for any other j', since if $j' > j$, any element y in $a_{j'}$ satisfies $y > x$ and also y comes before x in $a \circ b$. Moreover, if $j' < j$, any element y in $a_{j'}$ satisfies $y < x$ and also y comes after x in $a \circ b$.

Thus, t can contain only elements of $a_j \circ b$. Now if $j > i$, then every element of a_j is greater than every element of b, and thus t can only contain elements in a_j or b. But a_j and b have size less than k, contradicting that t has size at least k. Suppose then that $j < i$. Then $a_j \circ b$ contains at most $j + (k - i) < k$ elements, again contradicting the size of t. Thus, $j = i$, and then $a_j \circ b$ is the only subsequence of $a_j \circ b$ with size at least k, so $t = s$. This completes the proof.

THEOREM 3.1. *Assume* $|\Sigma| = \Omega(k^{2+\delta})$ *for a constant* $\delta > 0$. *Then* $R_{1/3}^{1-way}(\mathsf{LIS}) = \Omega(k^2 \log \frac{|\Sigma|}{k})$.

Proof. We reduce from the indexing function IND. Since $\frac{M}{k}$ is a power of 2, a may be viewed as a bit string of length $t = \sum_{i=1}^{k-1} i \log \frac{M}{k} = \Theta(k^2 \log \frac{|\Sigma|}{k})$, and any such string gives rise to a sequence a. Suppose Alice and Bob are given an instance of $IND : \{0,1\}^t \times [t] \rightarrow \{0,1\}$. Alice can interpret her given string a as a sequence $a = a_{k-1} \circ a_{k-2} \circ a_{k-3} \circ \cdots \circ a_1$ as in the above construction of a. Bob can then interpret his index as uniquely indexing the ℓth bit of some integer $a_{i,j}$ in some sequence a_i in a. Bob constructs b by choosing this value of i in step 1 of the above procedure for generating b. By Lemma 3.1, $\mathsf{LIS}(a, b)$ contains a_i, and thus $a_{i,j}$, and thus the ℓth bit of $a_{i,j}$. Thus $R_{1/3}^{1-way}(\mathsf{LIS}) \geq R_{1/3}^{1-way}(IND) = \Omega(t) = \Omega\left(k^2 \log \frac{|\Sigma|}{k}\right)$.

THEOREM 3.2. *Assume* $|\Sigma| = \Omega(k^{2+\delta})$ *for some constant* $\delta > 0$. *Any (possibly randomized)* 1*-pass streaming algorithm for* LIS *has space complexity* $\Omega(k^2 \log \frac{|\Sigma|}{k})$.

One can get analogous lower bounds for smaller Σ. For instance, if $|\Sigma| = \Omega(k)$, one can partition $[|\Sigma|]$ into $|\Sigma|/(2k)$ intervals each of size $2k$. Further partition each interval into pairs of integers. There are 2^i different choices for a_i (two choices for each pair), and $R_{1/3}^{1-way}(\mathsf{LIS}) = \Omega(|\Sigma|)$ follows along similar lines. We omit the details.

4 LIS-length

We first give bounds for LIS-length in the two-party communication model. We assume $|\Sigma| = \Omega(k)$, and handle the case $k = \Omega(|\Sigma|)$ in Appendix 7. Alice is given $a \in \Sigma^n$, Bob is given $b \in \Sigma^n$, and their goal is to compute $\mathsf{LIS\text{-}length}(a \circ b)$. Later we consider these problems in the data-stream model.

4.1 Deterministic multi-round lower bounds

We reduce from the following function.

DEFINITION 4.1. $ORD_k(x_1, \ldots, x_k, y_1, \ldots, y_k) =$
1, *if* $x_j \geq y_j, j = 1, \ldots, k,$
0, *otherwise.*
where $x_i, y_i \in \{1, \ldots, m\}$.

Consider the related $m^k \times m^k$ communication matrix M whose $(x_1, \ldots, x_k,\ y_1, \ldots, y_k)$th entry is 1 iff $ORD_k(x_1, \ldots, x_k, y_1, \ldots, y_k) = 1$. By appropriately ordering the tuples $(x_1, \ldots, x_k)$ and $(y_1, \ldots, y_k)$, M can be written as a lower triangular matrix with diagonal 1. Thus, M has full rank. Therefore, $D(ORD_k) = \Omega(\log rank(ORD_k)) = \Omega(\log m^k) = \Omega(k \log m)$.

Assume we have an ORD_k instance, and consider the following LIS-length instance: the input of Alice is $a = a_1, \ldots, a_k$, with $a_j = 2m(j-1) + 2x_j$, where $j = 1, \ldots, k$, and the input of Bob is $b = b_1, \ldots, b_k$, with $b_j = 2m(j-1) + 2y_j - 1$, where $j = 1, \ldots, k$. Here $\Sigma = \{1, \ldots, 2mk\}$.

We claim that $\mathsf{LIS\text{-}length}(a \circ b) = k + 1 - ORD_k(x, y)$. From the construction of a and b we know that $a_1 < a_2 < \cdots < a_k$, $b_1 < b_2 < \cdots < b_k$, and $a_{j+1} > b_j$ for $j = 1, \ldots, k-1$. So $\mathsf{LIS\text{-}length}(a \circ b) = k$ or $k+1$. $\mathsf{LIS\text{-}length}(a \circ b) = k + 1$ if and only if there exists a j such that $a_j \leq b_j$, i.e. $2x_j \leq 2y_j - 1$, or $x_j < y_j$. This is equivalent to $ORD_k(x, y) = 0$. Thus $\mathsf{LIS\text{-}length}(a \circ b) = k + 1 - ORD_k(x, y)$. Therefore, $D(\mathsf{LIS\text{-}length}(a \circ b)) = D(ORD_k(x, y)) = \Omega(k \log m) = \Omega(k \log |\Sigma|/k)$.

4.2 Randomized one-round lower bounds

Our lower bounds for $R_{1/3}^{1-way}(\mathsf{LIS\text{-}length})$ are derived from $R_{1/3}^{1-way}(ORD_k)$.

LEMMA 4.1. $R_{1/3}^{1-way}(ORD_k) = \Omega(k \log m)$.

Proof. Let $f : \{0,1\} \times \emptyset \rightarrow \{0,1\}$ be defined by $f(z) = z$. Let $t = k \log m$, and consider a variation of f, $f^{(t)}$: Alice has t inputs $z_1, \ldots, z_t$, and Bob has inputs i and $z_1, \ldots, z_{i-1}$, and the goal is to output z_i. Suppose in a 1-round protocol Alice sends $O(t)$ bits to Bob, and Bob outputs z_i with probability at least $2/3$. Then, for an appropriate choice of constant in the big-Oh, Theorem 2.1 gives a zero-round protocol in which Bob outputs z_i

with probability at least 3/5. This is impossible since $z_i \in \{0,1\}$ is arbitrary, and so Bob's probability is at most 1/2. Thus $R^{1-way}_{1/3}(f^{(t)}) = \Omega(t)$.

Now suppose Alice and Bob are given an instance of $f^{(t)}$. Assuming m is a power of 2 (the general proof works by replacing m with an integer in $[m/2, m]$), Alice can group the $z_1, \ldots, z_t$ into k integers $x_1, \ldots, x_k \in [m]$. This works by setting z_1 to be the most significant bit of x_1, $z_{\log m}$ the least significant bit of x_1, etc. Then Bob can also group $z_1, \ldots, z_{i-1}$ into integers, obtaining some number j of integers $x_1, \ldots, x_j$, together with the leading bits of x_{j+1}. Bob then creates k integers y_ℓ as follows. If $\ell \neq j+1$, Bob sets $y_\ell = 1$. Bob creates y_{j+1} by concatenating the leading bits of x_{j+1} with the bit sequence $100\ldots0$. It is easy to see that $x_\ell \geq y_\ell$ for all ℓ iff $x_{j+1} \geq y_{j+1}$, which holds if and only if $z_i = 1$, and thus $R^{1-way}_{1/3}(ORD_k) \geq R^{1-way}_{1/3}(f^{(t)}) = \Omega(t)$.

By the reduction in Section 4.1, it follows that $R^{1-way}_{1/3}(\textsf{LIS-length}) = \Omega(k \log |\Sigma|/k)$.

4.3 Randomized multi-round lower bounds This time we reduce from disjointness DIS on inputs of size k. From Theorem 2.2, $R_{1/3}(DIS) = \Omega(k)$. Assume we have an instance of DIS. The inputs are two k bit strings x, y. Now consider the following LIS-length problem: the input of Alice is $a = a_1 \ldots a_k$, with $a_j = 4j - 2x_j$, where $j = 1, \ldots, k$, and the input of Bob is $b = b_1 \ldots b_k$, with $b_j = 4(j-1) + 1 + 2y_j$, where $j = 1, \ldots, k$. Here $\Sigma = \{1, \ldots, 4k\}$. We claim that $\textsf{LIS-length}(a \circ b) = k + 1 - DIS_k(x, y)$.

One can see that $\textsf{LIS-length}(a \circ b) = k$ or $k+1$. Moreover, $\textsf{LIS-length}(a \circ b) = k+1$ if and only if there exists a $j \in \{1, \ldots, k\}$ such that $a_j \leq b_j$, i.e., $4j - 2x_j \leq 4(j-1) + 1 + 2y_j$, or $2x_j + 2y_j \geq 3$. The only time this can happen is if $x_j = y_j = 1$. This is equivalent to $DIS(x, y) = 0$. Thus $\textsf{LIS-length}(a \circ b) = k + 1 - DIS(x, y)$. Therefore, $R(\textsf{LIS-length}(a \circ b)) = R(DIS(x, y)) = \Omega(k)$. The proof uses $|\Sigma| = 4k$, but it works just as well when $|\Sigma| > 4k$.

4.4 LIS-length in the streaming model The lower bounds in the previous three subseections, as well as those in Appendix 7 apply to streaming algorithms via the reduction in Section 2. Therefore,

THEOREM 4.1. *Any $O(1)$-pass deterministic streaming algorithm for* LIS-length *and any 1-pass randomized streaming algorithm for* LIS-length *requires space $\Omega(\log \binom{|\Sigma|+k-1}{k-1})$. Any $O(1)$-pass randomized streaming algorithm for* LIS-length *requires space $\Omega(\min(k, |\Sigma|))$.*

In [19] the authors achieve $O(k \log |\Sigma|)$ space in the streaming model. We improve their data-stream algorithm to match the lower bound of Theorem 4.1. Recall their algorithm:

There is an array A, which always contains at most $k+1$ entries. The invariant maintained is that at any point in time processing the input stream, $A[i]$ contains the final entry of an LIS of length i whose last element is smallest possible over all such subsequences. The exception is that the last entry of A contains ∞. This implies $A[1] \leq A[2] \leq A[3] \cdots \leq A[r] < A[r+1]$, where $r + 1 \leq k + 1$ is the last entry of A. To maintain the invariant, when processing stream element x, one finds i for which $A[i] \leq x < A[i+1]$, replaces $A[i+1]$ with x, and increases r if necessary. This works because x extends the sequence of length i to a sequence of length $i+1$ with a smaller final element.

We improve the space complexity as follows. Since $A[1] \leq A[2] \leq \cdots A[r+1]$, we can encode the entries of A by $A[1], A[2] - A[1], A[3] - A[2], \ldots, A[r] - A[r-1]$. To process item x, sum the differences from left to right until the value x is exceeded. Suppose we find that $A[i] \leq x < A[i+1]$. In our list replace $A[i+1] - A[i]$ with $x - A[i]$ and replace $A[i+2] - A[i+1]$ with $A[i+2] - x$.

THEOREM 4.2. *There exists a 1-pass algorithm for* LIS-length *using space $O(k \log |\Sigma|/k) = O(\log \binom{|\Sigma|+k-1}{k-1})$.*

We note that though this is optimal in space, we have increased the time complexity.

5 Longest Common Subsequence

In [19], the authors show that when Σ is unrestricted, $R_{1/3}(\textsf{LCS-length}) = \Omega(N)$ (which thus holds for LCS). They also mention that there is a trivial $O(N \log |\Sigma|)$ 1-pass deterministic upper bound, which simply stores the entire stream. We note that by choosing a pairwise-independent hash function, this can be reduced to $O(N \log N)$ for 1-pass randomized algorithms.

Moreover, if one insists on deterministic constant-pass algorithms, there is a simple $\Omega(N \log |\Sigma|)$ lower bound. Consider the two party-setting in which Alice is given a and Bob is given b. Then $\textsf{LCS-length}(a, b) = N$ iff $a = b$, and thus there is a reduction from $EQ(a, b)$, which satisfies $D(EQ) = \Omega(|a|) = \Omega(N \log |\Sigma|)$.

As these bounds are trivial, the authors consider the case when both inputs a, b are permutations of $[N]$. We now focus on this case. Recall the LCS-k-decision problem: Alice and Bob are given permutations a, b of $[N]$ and the goal is to output 1 iff $|LCS(a, b)| \geq k$.

5.1 Lower bounds for LCS-k-decision The following padding lemma helps us focus on constant k.

LEMMA 5.1. *If for some constant k_0,* LCS-k_0-decision *on inputs with size m has $D(\cdot)$ or $R_{1/3}(\cdot)$ at least*

$\Omega(f(m))$ *for a non-decreasing function* $f(*)$*, then for any* $k_0 \le k \le n/2$, LCS-k-decision *on inputs with size* n *has* $D(\cdot)$ *or* $R_{1/3}(\cdot)$ *at least* $\Omega(f(n/2))$.

Proof. Consider an instance of LCS-k_0-decision problem on inputs of size $n/2$. Suppose the inputs of Alice and Bob are a and b, respectively. Pad a and b by a string $s = n/2+1, n/2+2, \cdots, n/2+k-k_0$. Write $a' = a \circ s$, $b' = b \circ s$. The size of a', b' is $|a'| = |b'| = n/2 + k - k_0 < n$. It is easy to show that $LCS(a', b') = LCS(a, b) + |s| = LCS(a, b) + k - k_0$. Therefore, $LCS(a', b') \ge k \Leftrightarrow LCS(a, b) \ge k_0$. Since the LCS-$k_0$-decision problem has $D(\cdot)$ or $R_{1/3}(\cdot)$ at least $\Omega(f(n/2))$, so the LCS-k-decision problem has $D(\cdot)$ or $R_{1/3}(\cdot)$ at least $\Omega(f(n/2))$.

Lemma 5.2. D(LCS-2-decision) $= \Omega(N \log N)$.

Proof. Since a and b are permutations of $[N]$, it is easy to show that $LCS(a, b) = 1$ iff a is the reverse of b. Therefore, the LCS-2-decision problem is equivalent to the decision problem $EQ(a, r(b))$ (here we use $r(b)$ to represent the reserve permutation of b). If we sort the inputs a and b in an appropriate order, the communication matrix M of the $EQ(a, r(b))$ function is the identity matrix. By the rank lower bound, $D(EQ(a, r(b))) \ge \log(rank(M)) = \log N! = \Omega(N \log N)$.

Combining Lemma 5.1 and Lemma 5.2,

Theorem 5.1. *Assume* $2 \le k \le N/2$*. Then* D(LCS-k-decision) $= \Omega(N \log N)$.

For the $N/2 < k \le N$ case, we give a different proof,

Theorem 5.2. *Assume* $N/2 < k \le N$*. Then* D(LCS-k-decision) $= \Omega(N \log N)$.

Proof. Consider the following inputs (a, b): $a = a' \circ k, k+1, \ldots, N$, $b = b' \circ N, N-1, \ldots, k$, where a', b' are permutations of $[k-1]$. It is clear $LCS(a, b) = LCS(a', b') + 1$. Thus $LCS(a, b) \ge k$ iff $LCS(a', b') \ge k-1$. But a', b' are both permutations of $[k-1]$, so $LCS(a', b') \ge k-1 \Leftrightarrow a' = b'$. Therefore, $LCS(a, b) \ge k$ iff $a' = b'$, so the communication matrix is the identity matrix. From the rank lower bound, D(LCS-k-decision) $= \Omega(\log(k-1)!) = \Omega(k \log k) = \Omega(N \log N)$.

In [19] an $\Omega(N/\rho^2)$ bound is given for LCS-ρ-approx, the promise problem of determining whether $|LCS(a, b)| \ge \rho^2$ or $|LCS(a, b)| \le \rho$, for any $\rho \ge 2$. Thus, $R_{1/3}$(LCS-4-decision) $= \Omega(N)$. Lemma 5.1 allows us to extend this to larger k, greatly improving the $\Omega(k)$ bound of [19]:

Theorem 5.3. *Assume* $4 \le k \le N/2$*. Then* $R_{1/3}$(LCS-k-decision) $= \Omega(N)$.

Here we show that LCS-3-decision also has high randomized complexity.

Lemma 5.3. $R_{1/3}$(LCS-3-decision) $= \Omega(N)$.

Proof. We reduce from DIS. W.l.o.g., we assume N is a multiple of 3, and divide the integers $\{1, ..., N\}$ into groups of size 3: $G_1 = \{1, 2, 3\}$, $G_2 = \{4, 5, 6\}, \ldots, G_{N/3} = \{N-2, N-1, N\}$.

Suppose we have an instance of DIS. Alice is given an $N/3$ bit string x, and Bob an $N/3$ bit string y, and they want $DIS(x, y)$. By Theorem 2.2 this has randomized complexity $\Omega(N)$.

Now Alice creates input $a = p_1(G_1) \circ p_2(G_2) \circ \cdots \circ p_{N/3}(G_{N/3})$, where $p_i(m+1, m+2, m+3)$ outputs a permutation of $\{m+1, m+2, m+3\}$ defined as follows:

$$p_i(x+1, x+2, x+3) = \begin{matrix} m+1, m+2, m+3 & \text{if} & x_i = 0 \\ m+1, m+3, m+2 & \text{if} & x_i = 1 \end{matrix}$$

Bob creates $b = q_{N/3}(G_{N/3}) \circ q_{N/3-1}(G_{N/3-1}) \circ \cdots q_1(G_1)$, here

$$q_i(m+1, m+2, m+3) = \begin{matrix} m+3, m+2, m+1 & \text{if} & y_i = 0 \\ m+1, m+3, m+2 & \text{if} & y_i = 1 \end{matrix}$$

Note that a, b are both permutations of $[N]$. We claim $LCS(a, b) \le 2$ iff x and y are disjoint. Any common subsequence of a and b can contain integers from at most one G_i. Indeed, if j and l were from different G_i, j would appear before l in one sequence, but after l in the other.

Consider any $G_i = \{m+1, m+2, m+3\}$. Then $LCS(p_i(G_i), q_i(G_i)) = 3$ iff $p_i(G_i) = q_i(G_i)$ which occurs iff $x_i = y_i = 1$. Otherwise, $LCS(p_i(G_i), q_i(G_i)) \le 2$. So $LCS(a, b) \le 2$ if $DIS(x, y) = 1$, and $LCS(a, b) = 3$ otherwise. We conclude $R_{1/3}$(LCS-3-decision) $\ge R_{1/3}(DIS) = \Omega(N)$.

5.2 Upper bounds for LCS-k-decision We give an $O(r \log N)$ protocol for LCS-k-decision when $k = N - r$ for small r, which is useful for similar strings. We will use the 1-round $O(\log n)$ protocol for $EQ(x, y)$ function as a subroutine [18]. We use the property that this protocol has a small one-sided error: when $x = y$, the error is 0 and when $x \ne y$, the error is less than $1/n$. Now we give the protocol for LCS-k-decision:

Alice and Bob represent their permutations as $O(N \log N)$ bit strings. Alice runs the EQ protocol on her string $3r$ times independently, and sends all the results to Bob (this will need $O(3r \log(N \log N)) = O(r \log N)$ bits). Bob does the following check: for every permutation a' such that $LCS(a', b) \ge N - r$, use the EQ protocol to check whether $a = a'$. Since Alice sends $3r$ copies, for each a' Bob needs to check $3r$ times.

Bob outputs 1 if there exists an a' such that a' passes all the $3r$ EQ tests, and outputs 0 otherwise.

Now we prove the protocol is correct: if $LCS(a,b) \geq N-r$, then Bob always outputs 1. Indeed, the EQ protocol does not have error when the inputs are equal. Now if $LCS(a,b) < N-r$, Bob outputs 1 if and only if there exists some a' such that $LCS(a',b) \geq N-r$, and a' passes all the $3r$ times EQ tests. For a string $a' \neq a$, the probability that a' passes all the $3r$ times tests is less than N^{-3r}. So by the union bound, $\Pr[\text{Bob outputs 1}] \leq \sum_{a':LCS(a',b)\geq N-r} \frac{1}{N^{3r}}$.

We claim that there are at most N^{2r} sequences with $LCS(a',b) \geq N-r$. Suppose π is the permutation such that $\pi(b_i) = i$ for $i = 1,\ldots,N$. Then $LCS(a',b) = LIS(\pi(a'_1)\pi(a'_2)\ldots\pi(a'_n))$. So we only need to bound the number of permutations with LIS at least $N-r$. One can choose an increasing subsequence of size $N-r$ in $\binom{N}{N-r} < N^r$ ways, then insert the remaining r elements in $(N-r+1)(N-r+2)\cdots N < N^r$ ways. Thus there are at most N^{2r} such sequences. Therefore, $\Pr[\text{Bob outputs 1}] \leq N^{2r}\frac{1}{N^{3r}} < \frac{1}{3}$, so $R_{1/3}(\mathsf{LCS}\text{-}k\text{-decision}) = O(r\log N)$.

5.3 Bounds for LCS-ρ-approx

Fact 5.1. *(see, e.g., [20]) There exist $2^{\Omega(n)}$ elements in $\{0,1\}^n$ such that for any two elements $x \neq y$, their hamming distance $\Delta(x,y)$ satisfies $\Delta(x,y) \geq (\frac{1}{2}-\delta)n$. Here $\delta > 0$ can be any constant.*

Theorem 5.4. *Assume $\rho^2 < (\frac{1}{2}-\delta)N$. Then $D(\mathsf{LCS}\text{-}\rho\text{-approx}) = \Omega(N)$.*

Proof. Let $n = N/2$ and S be the set of Fact 5.1. Then $|S| = 2^{\Omega(N)}$. Consider the $EQ(x,y)$ problem on input $S \times S$. Then $D(EQ) \geq \log|S| = \Omega(N)$. Now we create an LCS problem. For $(x,y) \in S \times S$, Alice creates $a = p_1p_2\ldots p_{N/2}(N+1-p_{N/2})(N+1-p_{N/2-1})\ldots(N+1-p_1)$, where

$$p_i = \begin{cases} i & \text{if} \quad x_i = 0, \\ N+1-i & \text{if} \quad x_i = 1. \end{cases} \quad (i = 1,\ldots,N/2).$$

Bob creates $b = q_1q_2\ldots q_{N/2}(N+1-q_{N/2})(N+1-q_{N/2-1})\ldots(N+1-q_1)$, where

$$q_i = \begin{cases} N+1-i & \text{if} \quad y_i = 0, \\ i & \text{if} \quad y_i = 1. \end{cases} \quad (i = 1,\ldots,N/2).$$

Note that a,b are both permutations of $[N]$. From the construction it is easy to show that if $x = y$, then $a = r(b)$, thus $LCS(a,b) = 1$. Moreover, $LCS(a,b) \geq 2\Delta(x,y)$. Indeed, consider the positions i such that $x_i \neq y_i$, and also positions $N+1-i$. They form a $2\Delta(x,y)$-length common subsequence of a and b. Since in set S, if $x \neq y$, $\Delta(x,y) \geq (\frac{1}{2}-\delta)n = (\frac{1}{4}-\frac{\delta}{2})N$. Thus if $x \neq y$, $LCS(a,b) \geq (\frac{1}{2}-\delta)N$. Therefore, if we can separate the cases $LCS(a,b) = 1$ or $LCS(a,b) \geq (\frac{1}{2}-\delta)N$, then we can decide whether $x = y$. Since $D(EQ) = \Omega(N)$, we have $D(\mathsf{LCS}\text{-}\rho\text{-approx}) = \Omega(N)$ for any $\rho^2 < (\frac{1}{2}-\delta)N$.

5.4 Streaming algorithm We give an $O(\frac{N}{\rho}\log\frac{N}{\rho} + \log N)$-space randomized 1-pass streaming algorithm for this problem. The streaming algorithm $A(a \circ b)$ first chooses a pairwise-independent hash function $h : [N] \to [N]$. Let a', b' be the subsequences of a and b, respectively, of elements $i \in [N]$ for which $h(i) \leq 4N/\rho$. A stores a' and b' in their entirety, provided that $|a'| + |b'| = 2|a'| \leq 48N/\rho$. If this condition is not met, A reports that $|LCS(a,b)| \leq \rho$ (alternatively, A could output fail). The expected length of a' is $4N/\rho$, so the probability of this event is at most 1/6. A computes $X = |LCS(a',b')|$. If $X > \rho$, A reports $|LCS(a,b)| \geq \rho^2$, and otherwise reports $|LCS(a,b)| \leq \rho$.

Since h is pairwise-independent, Chebyshev's inequality implies that A outputs the correct answer with constant probability - we omit the details due to space constraints. A can compute h in $O(\log N)$ space. Then, in $O(\log\binom{N+\Theta(N/\rho)}{\Theta(N/\rho)})$ space A can create a list $\delta_1,\delta_2,\ldots,\delta_{\Theta(N/\rho)}$ such that $\delta_i > 0$ for all i, and $h(x) \leq 4N/\rho$ for $x \in [N]$ iff $x = \sum_{i=1}^{j}\delta_i$ for some j. Next, when processing element x of a, A can sum the δ_i from left to right. If it finds $x = \sum_{i=1}^{j}\delta_i$ for some j, A maps $x \in a$ to the integer j, and appends j to a'. A similarly constructs b'. a' and b' then only consume $O(N/\rho\log N/\rho)$ space. A can then use the linear-space algorithm of Fredman [10] to compute $LCS(a',b')$ (since a' and b' are permutations on the same $|a'|$ symbols, this can be transformed into an instance of LIS). Thus, A's total space is $O(\log N + N/\rho\log N/\rho)$.

Open Questions: One key open problem is the complexity of multi-pass LIS. We have shown a tight $\Theta(k^2\log|\Sigma|/k)$ bound for 1-pass randomized algorithms, and an essentially tight $\Omega(k)$ bound for a large number of passes, but we don't have results for an intermediate number of passes. Resolving the gaps for the other problems is also a challenging open problem.

References

[1] D. Aldous and P. Diaconis. *Longest increasing subsequences: from patience sorting to the Baik-Deift-Johansson theorem*, Bull. Amer. Math. Soc., **36**, 1999, pp. 413-432.

[2] N. Alon, Y. Matias, and M. Szegedy. *The space complexity of approximating the frequency moments.* In STOC, 1996.

[3] S.F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman. *Basic local alignment search tool.* Journal of Molecular Biology, 215:403-410, 1990.

[4] A. Apostolico and C. Guerra. *The longest common subsequence problem revisited. Algorithmica*, 2:315-336, 1987.

[5] J. Baik, P. Deift, and K. Johansson. *On the distribution of the length of the longest increasing subsequence of random permutations,* J. Amer. Math. Soc. 12 (1999), no. 4, 1119-1178.

[6] A. Banerjee and J. Ghosh. *Clickstream clustering using weighted longest common subsequence.* In: ICM Workshop on Web Mining, 2001.

[7] S. Bespamyatnuikh and M. Segal. *Enumerating longest increasing subsequences and patience sorting.* Information Processing Letters, 76(1-2):7-11, 2000.

[8] P. Deift. *Integrable systems and combinatorial theory.* Notices Amer. Math. Soc., **47**, 2000, pp. 631-640.

[9] A. L. Delcher, S. Kasif, R. D. Fleischmann, J. Peterson, O. White, and S. L. Salzberg. *Alignment of whole genomes.* Nucleic Acides Research, 27(11):2369-2376, 1999.

[10] M.L. Fredman. *On computing the length of the longest increasing subsequences.*. In Discrete Mathematics **11**, 1975, pp. 29-35.

[11] G. Gonnet and R. Baeza-Yates. *Handbook of Algorithms and Data Structures.* Addison-Wesley, Wokingham, England, second edition, 1991.

[12] P. Gopalan, T.S. Jayram, R. Krauthgamer, and R. Kumar. *Estimating the Sortedness of a Data Stream.* In SODA 2007, to appear.

[13] M.R. Henzinger, P. Raghavan, and S. Rajagopalon. *Computing on data streams.* Technical Report 1998-011, Digital Equipment Corp., Systems Res. Center, 1998.

[14] D. Hirschberg. *Algorithms for the longest common subsequence problem.* Journal of the ACM, 24:644-675, 1977.

[15] J. Hunt and T. Szymanski. *A fast algorithm for computing longest common subsequences.* Communications of the ACM, 20:350-353, 1977.

[16] B. Kalyanasundaram and G. Schnitger. *The probabilistic communication complexity of set intersection.* SIAM J. Disc. Math **5** (1992), 545-557.

[17] I. Kremer, N. Nisan, and D. Ron. *On randomized one-round communication complexity.* In CCC, 1999.

[18] E. Kushilevitz and N. Nisan. *Communication Complexity.* Cambridge University Press, 1997.

[19] D. Liben-Nowell, E. Vee, and A. Zhu. *Finding Longest Increasing and Common Subsequences in Streaming Data.* In COCOON, 2005.

[20] F. J. MacWilliams and N. J. A. Sloane. *The Theory of Error-Correcting Codes*, North-Holland, Amsterdam, 1977.

[21] P. Bro Miltersen, Noam Nisan, S. Safra, and A. Wigderson. *On data structures and asymmetric communication complexity. Journal of Computer and System Sciences*, 57(1):37-49, 1998.

[22] M. Muthukrishnan. *Data Streams: Algorithms and Applications.* In SODA, 2003.

[23] A. Razborov. *On the distributional complexity of disjointness.* JCSS **28**, 1984.

[24] D. Sankoff and J. Kruskal. *Time warps, string edits, and macromolecules: The theory and practice of sequence comparison*, Addison-Wesley, 1983.

[25] C. Schensted. *Longest increasing and decreasing subsequences.* Canadian Journal of Mathematics, 13:179-191, 1961.

[26] A. C.-C. Yao. *Lower bounds by probabilistic arguments.* In FOCS, 1983.

[27] H. Zhang. *Alignment of BLAST high-scoring segment pairs based on the longest increasing subsequence algorithm.* Bioinformatics, **19**(11):1391-1396, 2003.

6 Appendix: Upper Bound for LIS

We first recall the streaming algorithm of [19] for computing the LIS of a data stream in $\lceil \log(1+1/\epsilon) \rceil$ passes and using $O(k^{1+\epsilon} \log |\Sigma|)$ space for any $\epsilon > 0$.

The algorithm for 1-pass is similar to that in Section 4.4. We have an array A containing at most $k+1$ entries. The difference is that the ith entry of A also contains an LIS (not just the final entry) of length i whose last element is smallest possible. One maintains the invariant on $A[i]$ as before, except instead of replacing $A[i+1]$ with x when $A[i] \le x < A[i+1]$, one replaces $A[i+1]$ with the LIS in $A[i]$, prepended to x.

For multiple passes, the idea in [19] is in the first pass to only store every qth element of each LIS appearing in an entry of A. Then after the first pass the state is $b_1, b_{q+1}, b_{2q+1}, \ldots$, where $b_1 \le b_2 \le b_3 \le \cdots \le b_k$ is an LIS of the input stream. In subsequent passes the idea is to "fill in the blanks". Note that the LIS between b_1 and b_{q+1} occurs before the LIS between b_{q+1} and b_{2q+1}, so one can first compute the LIS between b_1 and b_{q+1}, then between b_{q+1} and b_{2q+1}, etc. The point is that the LIS between b_1 and b_{q+1} can be computed, after which we only retain the answer $b_1, b_2, b_3, \ldots, b_{q+1}$. We then free up the rest of the space and compute the LIS between b_{q+1} and b_{2q+1}, etc.

Our encoding trick is to represent each LIS in each entry $A[i]$ as a sequence of differences. Instead of storing $b_1, \ldots, b_i$, we store $b_1, b_2 - b_1, b_3 - b_2$, etc. Thus each LIS takes $\Theta\left(\log \binom{i+|\Sigma|-1}{i-1}\right)$ space. For the 1-pass algorithm the total space is $\Theta\left(\sum_{i=1}^{k} i \log \frac{|\Sigma|}{i}\right) \le \Theta\left(k \sum_{i=1}^{k} \log \frac{|\Sigma|}{i}\right) = \Theta\left(k \log \frac{|\Sigma|^k}{k!}\right) = \Theta\left(k^2 \log \frac{|\Sigma|}{k}\right)$.

The same trick can be applied to the multiple-pass algorithm, since in each pass the array A contains an increasing subsequence. The update rule is similar to that of Section 4.4.

THEOREM 6.1. *There is a deterministic* $\lceil \log(1 + \frac{1}{\epsilon}) \rceil$-

pass streaming algorithm for LIS *using space* $O(k^{1+\epsilon} \log \frac{|\Sigma|}{k})$ *for any* $\epsilon > 0$.

7 Appendix: Small Alphabets

In this section we assume $k = \Omega(|\Sigma|)$, and we prove lower bounds for LIS-length.

7.1 Deterministic multi-round We use the following direct-sum function:

DEFINITION 7.1. $\overline{ORD}_{|\Sigma|}(x_1, \ldots, x_{|\Sigma|}, y_1, \ldots, y_{|\Sigma|}) =$
1, *if* $x_j \le y_j, j = 1, \ldots, |\Sigma|$,
0, *otherwise*.
where $x_1 \le x_2 \cdots \le x_{|\Sigma|}, y_1 \le y_2 \cdots \le y_{|\Sigma|}$, *and* $x_i, y_i \in \{1, \ldots, m\}$.

The value of m used here is different from that of Definition 4.1. The related communication matrix of the $\overline{ORD}_{|\Sigma|}$ function is upper triangular with diagonal 1. Thus, the matrix has full rank. Thus, $D(\overline{ORD}_{|\Sigma|}) = \Omega(\log rank(\overline{ORD}_{|\Sigma|})) = \Omega(\log \binom{m+|\Sigma|-1}{m}) = \Omega(|\Sigma| \log m/|\Sigma|)$.

Assume we have a $\overline{ORD}_{|\Sigma|}$ problem, the inputs are x and y. Now consider the following LIS-length problem: The input of Alice is $a = |\Sigma|^{x_{|\Sigma|}} \cdots 2^{x_2} 1^{x_1}$, where j^{x_j} means $\overbrace{j \cdots j}^{x_j}$. The input of Bob is $b = |\Sigma|^{(2m-y_{|\Sigma|})} \cdots 2^{2m-y_2} 1^{2m-y_1}$. Since $x_1 \le \cdots \le x_{|\Sigma|}$, $y_1 \le \cdots \le y_{|\Sigma|}$, it is not hard to see that LIS-length$(a \circ b) = \max_{1 \le j \le |\Sigma|}(x_j + 2m - y_j)$. Since $x_i, y_i \in \{1, \ldots, m\}$, $k =$LIS-length$(a \circ b) \in [m, 3m]$.

Looking at the definition of $\overline{ORD}_{|\Sigma|}$, we have LIS-length$(a \circ b) \le 2m$ iff $ORD_{|\Sigma|}(x, y) = 1$. Therefore, $D($LIS-length$(a \circ b)) = D(ORD_{|\Sigma|}(x, y)) = \Omega(|\Sigma| \log m/|\Sigma|) = \Omega(|\Sigma| \log k/|\Sigma|)$.

7.2 Randomized one-round We now give the lower bound for $R^{1-way}_{1/3}($LIS-length$)$ when $|\Sigma| = O(k)$.

LEMMA 7.1. $R^{1-way}_{1/3}(\overline{ORD}_{|\Sigma|}) = \Omega(|\Sigma| \log m/|\Sigma|)$.

Proof. We consider the same function $f^{(t)}$ as in the case $|\Sigma| = \Omega(k)$, but with the value $t = |\Sigma| \log m/|\Sigma|$. Suppose Alice is given $z_1, \ldots, z_t$. She creates $|\Sigma|$ integers $x_1 \le x_2 \le \cdots \le x_{|\Sigma|}$ as follows. She partitions $[m]$ into $|\Sigma|$ contiguous blocks of size $m/|\Sigma|$. Then she uses the first $\log m/|\Sigma|$ bits of $z_1, \ldots, z_t$ to determine x_1 in the first block, the next $\log m/|\Sigma|$ bits to determine x_2, etc. Note that $x_1 < x_2 < \cdots < x_{|\Sigma|}$. Bob uses the $z_1, \ldots, z_{i-1}$ to determine $x_1, \ldots, x_j$ for some j, together with the leading bits of x_{j+1}. Bob then creates the $y_1, \ldots, y_{|\Sigma|}$ as follows. Bob sets $y_\ell = m$ for $\ell \le j$. Bob sets $y_\ell = m$ for $\ell > j+1$. Bob sets y_{j+1} to be the element which agrees with the leading bits of x_{j+1}, is 0 in the next bit, and is one in the remaining bits. It follows immediately that $R^{1-way}_{1/3}(\overline{ORD}_{|\Sigma|}) \ge R^{1-way}_{1/3}(f^{(t)}) = \Omega(t)$.

By the reduction in Section 4.1, when $|\Sigma| = O(k)$ we have $R^{1-way}_{1/3}($LIS-length$) = \Omega(|\Sigma| \log k/|\Sigma|)$.

7.3 Randomized multi-round Assume we have a DIS problem on inputs of size $|\Sigma|$, the inputs are x and y. Now consider the following LIS-length problem: The input of Alice is $a = |\Sigma|^{a_{|\Sigma|}} \cdots 2^{a_2} 1^{a_1}$, where $a_j = 2j + x_j$ $(j = 1, \ldots, |\Sigma|)$. The input of Bob is $b = |\Sigma|^{b_{|\Sigma|}} \cdots 2^{b_2} 1^{b_1}$, where $b_j = 2|\Sigma| - (2j - y_j)$. Then $a_1 \le \cdots \le a_{|\Sigma|}$, $b_1 \ge \cdots \ge b_{|\Sigma|}$, and it follows

$$\text{LIS} - \text{length}(a \circ b) = \max_{1 \le j \le |\Sigma|}(a_j + b_j) = 2|\Sigma| + \max_{1 \le j \le |\Sigma|}(x_j + y_j).$$

Thus, $k =$LIS-length$(a \circ b) \in \{2|\Sigma|, 2|\Sigma| + 1, 2|\Sigma| + 2\}$.

Looking at the definition of $DIS(x, y)$, we have LIS-length$(a \circ b) \ge 2|\Sigma| + 2$ iff $DIS(x, y) = 0$. Therefore, $R_{1/3}($LIS-length$(a \circ b)) \ge R_{1/3}(DIS(x, y)) = \Omega(|\Sigma|)$. The proof uses $k \sim 2|\Sigma|$, but it also works for large k with a minor modification.

7.4 1-pass deterministic streaming algorithm We have the array A in Section 4.4, but we save bits in a different way. Since $A[1] \le A[2] \le \ldots \le A[k+1]$, we can, for each $i \in [|\Sigma|]$, keep track of the smallest j and largest j' for which $A[j]$ and $A[j']$ contain i. By recording differences in this list, one can show the space is $O(|\Sigma| \log(k/|\Sigma|) + \log k)$. Details omitted.

8 Appendix: Randomized Multi-round Upper Bound for LIS-length

We show how to improve the $\log \binom{k+|\Sigma|-1}{k-1}$ space in the upper bound for LIS-length in Section 4 (which was stated in the streaming model, but is also a 1-round deterministic protocol) by allowing randomization and multiple rounds. This shows randomization plus multiple rounds is strictly more powerful than either 1-round or deterministic protocols for this problem.

Consider the case $|\Sigma| \ge k$. In this case Alice has $s_1 \le s_2 \le \cdots \le s_k$, Bob has $t_1 \ge t_2 \ge \cdots \ge t_k$, and they must compute LIS-length$(a \circ b) = \max_{\substack{1 \le i, j \le k \\ s_i \le t_j}} (i + j)$. We use the well-known fact [18] that the function $LT : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ defined by $LT(x, y) = 1$ iff $x < y$, satisfies $R_\delta(LT) = O(\log n + \log \frac{1}{\delta})$. The idea is for Alice to merge her s_i with the t_j, so that Bob obtains a sorted list containing both the s_i and the t_j. Then LIS-length$(a \circ b)$ can be determined locally by Bob.

In our application $n = \log |\Sigma|$, and we choose to set $\delta = O(1/k)$. Alice first compares s_1 with t_k. If

$s_1 > t_k$, she then compares s_1 with t_{k-1}, and continues until she finds a j for which $s_1 \le t_j$. Then she compares s_2 with t_j, and continues until she find a j' for which $s_2 \le t_{j'}$. This process repeats until Bob learns the entire sorted list. Clearly the number of invocations of LT is $O(k)$, and so by a union bound, Bob obtains the correct sorted list with probability at least 2/3. Thus, $R_{1/3}(\mathsf{LIS\text{-}length}) = O(k(\log k + \log\log|\Sigma|))$.

Now suppose $|\Sigma| < k$. Then Alice has $p_1 \le p_2 \le \cdots \le p_{|\Sigma|}$ and Bob has $q_1 \ge q_2 \ge \cdots \ge q_{|\Sigma|}$, and they want to compute $\mathsf{LIS\text{-}length}(a \circ b) = \max_{1 \le i \le |\Sigma|}(p_i + q_i)$. The idea again is to use an efficient LT sub-protocol. Alice sends p_1 to Bob, who computes $p_1 + q_1$. Then Alice computes $x_2 = p_2 - p_1 \ge 0$ and Bob computes $y_2 = q_1 - q_2 \ge 0$. Clearly, $p_2 + q_2 > p_1 + q_1$ iff $x_2 > y_2$. Alice and Bob run $LT(y_2, x_2)$ to do the comparison. If $p_2 + q_2 \le p_1 + q_1$, they then compute $x_3 = p_3 - p_1 \ge 0$ and $y_3 = q_1 - q_3 \ge 0$, and repeat. Otherwise, they record that $p_2 + q_2 > p_1 + q_1$ (without actually computing this sum), and then compute $x_3 = p_3 - p_2 \ge 0$ and $y_3 = q_2 - q_3 \ge 0$, and repeat. At the end, assuming all of the comparisons were correct, they determine the index i for which $p_i + q_i$ is maximum. Then Alice sends p_i to Bob who announces the sum. There are $O(|\Sigma|)$ comparisons of $\log k$ bit numbers, so setting $\delta = O(1/|\Sigma|)$ gives $R_{1/3}(\mathsf{LIS\text{-}length}) = O(|\Sigma|(\log|\Sigma| + \log\log k) + \log k)$.

9 Appendix: LIS-length-ϵ-apx and LIS-ϵ-apx

We assume that $\frac{1}{\epsilon},\ k = O(|\Sigma|)$. The section can easily be modified if either of these conditions does not hold.

9.1 Upper Bounds We start with an upper bound for LIS-length-ϵ-apx. Recall that Alice is given $a \in \Sigma^n$ and Bob $b \in \Sigma^n$. Let $k = \mathsf{LIS\text{-}length}(a \circ b)$. The goal is to output a number x with $(1-\epsilon)k \le x \le k$.

We start with the following 1-round deterministic protocol. Alice computes $k' = \mathsf{LIS\text{-}length}(a)$. She partitions $[k']$ into intervals

$$(0, \epsilon k'], (\epsilon k', 2\epsilon k'], (2\epsilon k', 3\epsilon k'], \ldots, ((k'-1)\epsilon k', k'].$$

For each interval I, she finds the smallest integer $x(I)$ for which there exists an LIS in a of length in I which ends in $x(I)$. If no such LIS exists, this interval is ignored. For all I she transmits $x(I)$ to Bob. She also transmits k'.

Bob uses b to extend each of the $x(I)$ into an increasing subsequence of $(a \circ b)$. Bob also computes the LIS of b. Suppose he finds that extending $x(I)$, for $I = (i\epsilon k', (i+1)\epsilon k']$, gives a longest such subsequence. Then Bob will output $i\epsilon k'$ plus the length of his extension in b. If the LIS of b is longest he will just output this length.

Fix an LIS s. If s occurs entirely in b, Bob outputs k. Otherwise split s into $s_a \circ s_b$, where s_a is the part of s in a. Suppose $|s_a| \in I$. Then Alice will send some $x(I)$ to Bob. It follows that $x(I)$ is less than the last element of s_a, and that Bob can extend the increasing subsequence ending in $x(I)$ with s_b. Thus his output is at least $|s_a| - \epsilon k' + |s_b| \ge (1-\epsilon)k$.

For the communication, there are at most $\lceil 1/\epsilon \rceil$ intervals. The $x(I)$ are non-decreasing with increasing I. Our usual encoding trick yields communication

$$O\left(\log\binom{\lceil 1/\epsilon\rceil + |\Sigma| - 1}{\lceil \frac{1}{\epsilon} \rceil}\right) = O\left(\frac{1}{\epsilon}\log(\epsilon|\Sigma|)\right).$$

Now, observe that instead of Alice transmitting $x(I)$ to Bob for each I, Alice can transmit an LIS in a of length in I which ends in $x(I)$ to Bob. This gives an upper bound for LIS-ϵ-apx of $O\left(\frac{k}{\epsilon}\log(|\Sigma|/k)\right)$.

9.2 Lower Bounds Let $t = \Theta(1/\epsilon)$. For LIS-length-ϵ-apx, the idea is to reduce from ORD_t. Suppose $x_1, \ldots, x_t$ is Alice's input and $y_1, \ldots, y_t$ is Bob's input. Alice first creates a stream a of length t in the same way as before, that is, she sets $a_j = 2m(j-1) + 2x_j$ for $j = 1, \ldots, t$. Next, she creates a stream a' of length k by replacing each element a_j in a with a block of k/t copies of a_j. Bob first creates a stream b of length t by setting $b_j = 2m(j-1) + 2y_j - 1$ for $j = 1, \ldots, t$. Then he creates a stream b' of length k by replacing each element b_j in b with a block of k/t copies of b_j. It is easy to see $\mathsf{LIS\text{-}length}(a' \circ b') = k + \Theta(\epsilon)k\,ORD_t(x, y)$.

As any streaming algorithm cfor LIS-length-ϵ-apx can distinguish between a stream of length $k + \Theta(\epsilon)k$ and a stream of length k (for an appropriate constant in the $\Theta(\cdot)$), it follows that its space complexity (for $O(1)$-pass algorithms) must be at least the communication complexity of ORD_t. Thus by the previous sections (note that we use $m = \Theta(\epsilon|\Sigma|)$ in the ORD_t definition): $D(\text{LIS-length-}\epsilon\text{-apx}) = \Omega\left(\frac{1}{\epsilon}\log(\epsilon|\Sigma|)\right)$ and $R_{1/3}^{1-way}(\text{LIS-length-}\epsilon\text{-apx}) = \Omega\left(\frac{1}{\epsilon}\log(\epsilon|\Sigma|)\right)$.

To get the lower bound for LIS-ϵ-apx, we can adapt the lower bound technique in Section 3. The rough idea is to create $t = \Theta(1/\epsilon)$ different a_i, and then set the stream $a = a_t \circ a_{t-1} \circ \cdots \circ a_1$, where every element in a_i is greater than any element in a_j for $j < i$. The difference now is that a_t has length $k-1$, a_{t-1} length $k - 1 - \frac{k-2}{t}$, a_{t-2} length $k - 1 - \frac{2(k-2)}{t}$, etc. The adversary then chooses b to extend exactly one of these sequences. We omit the details, but simply state that for large enough $|\Sigma|$, this gives the bound $R_{1/3}^{1-way}(\mathsf{LIS} - \epsilon - apx) = \Omega\left(\frac{k}{\epsilon}\log\frac{|\Sigma|}{k}\right)$.

For $R_{1/3}(\text{LIS-length-}\epsilon\text{-apx})$ (and thus $R_{1/3}(\text{LIS-}\epsilon\text{-apx})$), we reduce from DIS on inputs of size $\Theta(\frac{1}{\epsilon})$. We create the corresponding streams as in the reduction of Section 4, and we duplicate each element $\Theta(\epsilon k)$ times.