On Distance to Monotonicity and Longest Increasing Subsequence of a Data Stream

Funda Ergun* Hossein Jowhari†

Abstract

In this paper we consider problems related to the sortedness of a data stream. First we investigate the problem of estimating the distance to monotonicity; given a sequence of length n, we give a deterministic $(2+\epsilon)$-approximation algorithm for estimating its distance to monotonicity in space $O(\frac{1}{\epsilon^2}\log^2(\epsilon n))$. This improves over the randomized $(4+\epsilon)$-approximation algorithm of [3]. We then consider the problem of approximating the length of the longest increasing subsequence of an input stream of length n. We use techniques from multi-party communication complexity combined with a fooling set approach to prove that any $O(1)$-pass deterministic streaming algorithm that approximates the length of the longest increasing subsequence within $1+\epsilon$ requires $\Omega(\sqrt{n})$ space. This proves the conjecture in [3] and matches the current upper bound.

1 Introduction

Streaming algorithms are those that are allowed to have a small number (preferably one) of passes over the input data and use sublinear space (and small per-item running time). These restrictions often arise in real-world applications that deal with massive data sets and thus streaming algorithms have attracted considerable attention in the past few years [8]. The main theoretical challenges in this area are (i) reducing the space usage as much as possible and (ii) establishing tight bounds for the space complexity. Establishing such upper and lower bounds poses unique challenges for which novel techniques are required.

Among the set of problems that have been studied in this area, estimating the sortedness of an array has captured special attention [1, 6, 3, 9]. There are several measures for quantifying the sortedness of an array. Given a sequence σ of length n, the number of inversions in σ, denoted by $K(\sigma)$, is a measure for sortedness of σ which is defined as the number of pairs of items in σ which violate the natural ordering property. Another measure for sortedness is the length of the longest increasing subsequence in σ, denoted by $LIS(\sigma)$. One final measure is distance to monotonicity, denoted by $ED(\sigma)$, defined as the minimum number of single item deletions needed to reach a sorted sequence σ'. (It can be easily verified that in fact $ED(\sigma) = n - LIS(\sigma)$.) Both exact and approximation algorithms have been studied for these measures in the conventional and streaming models. Our focus in this paper is on approximating both the *distance to monotonicity* and the length of the *longest increasing subsequence.*

Previous Results. The interest in longest increasing subsequence and distance to monotonicity in the streaming model is relatively recent; we start with a brief list of known upper bounds. The first work on measuring the sortedness of a sequence was on the number of inverted pairs; Ajtai *et al.* in [1] showed that given an input sequence which is a permutation of $\{1...n\}$, $K(\sigma)$ can be approximated within a $1+\epsilon$ factor by a deterministic algorithm that uses only $O(\frac{1}{\epsilon}\log\log n)$ space. In a subsequent work, Gupta *et al.* [4] gave a $O(\frac{1}{\epsilon^3}\log n^3)$ space randomized streaming algorithm for general sequences. In [6], Liben-Nowell *et al.* considered the LIS problem and gave an exact deterministic algorithm for computing $LIS(\sigma)$ in space $O(LIS(\sigma))$. More recently Gopalan et al. [3] gave an $O(\sqrt{\frac{n}{\epsilon}})$ space deterministic algorithm for approximating both $LIS(\sigma)$ and $ED(\sigma)$ within a $1+\epsilon$ factor. Furthermore they described a $4+\epsilon$ factor randomized approximation algorithm for $ED(\sigma)$ that uses $O(\frac{1}{\epsilon^2}\log^2 n)$ space.

In terms of lower bounds, it is known that computing $K(\sigma), LIS(\sigma)$ and $ED(\sigma)$ exactly requires linear space even if σ is known to be a permutation [1, 3, 9]. For approximating LIS (within a constant factor) it has been conjectured by Gopalan, Jayram, Krauthgamer and Kumar [3] that any deterministic algorithm requires $\Omega(\sqrt{n})$ space. In a work independent of this paper, Gal and Gopalan proved this conjecture by showing a lower bound of $\Omega(\sqrt{n/\epsilon})$ which exactly matches the current upper bound for all approximation factors [2]. Finally Woodruff *et al.*[9] have shown some lower bounds for the problem of finding the longest

*School of Computing Science, Simon Fraser University, BC, Canada. funda@cs.sfu.ca.

†School of Computing Science, Simon Fraser University, BC, Canada. hjowhari@cs.sfu.ca.

increasing subsequence itself.

Our contributions. In this paper, we first consider the problem of approximating $ED(\sigma)$. We give a deterministic streaming algorithm which approximates $ED(\sigma)$ within a factor of $2+\epsilon$ in $O(\frac{1}{\epsilon^2}\log^2 \epsilon n)$ space; this improves the previous $4+\epsilon$ factor randomized approximation algorithm of [3]. Our algorithm uses an improved version of the estimator used in [3] and is based on the characterization of distance-to-monotonicity by inversions. In contrast to [3] which uses sampling, we use a specific data structure for quantile estimation from [7] that enables us to estimate the median in any window over the stream without using randomization.

The second main result of this paper is a proof for the conjecture of Gopalan, Jayram, Krauthgamer and Kumar [3]. More specifically we show that any $O(1)$-pass deterministic algorithm that approximates $LIS(\sigma)$ within a factor of $1+\epsilon$ for small ϵ requires $\Omega(\sqrt{n})$ (bits of) space. Our proof is based on multi-player communication complexity combined with a direct-sum approach that is suggested in [3]. Although our proof uses a similar framework to one used in [2, 3], our method towards deriving a bit-complexity lower bound is based on a novel primitive function. Our primitive function enables us to get a lower bound which holds even in the blackboard model and also directly implies a lower bound for $O(1)$-pass algorithms. We use probabilistic method to show the existence of a large fooling set for the primitive function which can be easily generalized to find a large fooling set for the gap version of LIS.

2 Preliminaries

Let $\sigma = \sigma(1), ..., \sigma(n)$ be a sequence of n elements from the set $\{1, ..., m\}$. Let $LIS(\sigma)$ be the length of a longest increasing subsequence in σ (sometimes we use LIS alone to refer to the longest increasing subsequence itself). Let $ED(\sigma)$ denote the distance of σ to the closest monotone sequence, i.e., $ED(\sigma)$ is the minimum number of delete operations needed to reach a monotone sequence from σ. Note that this is the same as $n - LIS(\sigma)$.

We use $[m]$ to denote the set $\{1, \ldots, m\}$. We sometimes abuse notation and apply set operations to lists and intervals of discrete numbers. These are to be understood as being applied to the set of elements comprising these lists and intervals.

3 Approximating the distance to monotonicity

In this section we present an algorithm for approximating the distance to monotonicity. Recently Gopalan et al [3] have presented two streaming algorithms for approximating this quantity: a deterministic $(1+\epsilon)$-approximation algorithm that uses $O(\sqrt{n/\epsilon})$ space and a randomized $4+\epsilon$-approximation algorithm that uses $O(\frac{1}{\epsilon^2}\log^2 n)$ space. In this paper, we improve these results by presenting a deterministic $(2+\epsilon)$-approximation algorithm that uses $O(\frac{1}{\epsilon^2}\log^2 n)$ space.

3.1 Our general approach We first define an estimator which approximates the size of the LIS, then design an algorithm which approximates the value of the estimator itself. On a high level, our estimator, which is based on a modification to an estimator in [3], identifies a set of disjoint non-increasing subsequences in σ and uses the sum of the lengths of these sequences as a lower bound for $ED(\sigma)$. The following lemma shows the relationship between these non-increasing subsequences and $ED(\sigma)$.

LEMMA 3.1. *Let $P = \{\sigma_1, ..., \sigma_t\}$ be a set of disjoint and non-increasing subsequences in σ. We have $ED(\sigma) \geq (\sum_{i=1}^{t} |\sigma_i|) - t$.*

Proof. Let π be a longest increasing subsequence in σ. It is easy to see that in any non-increasing subsequence of length k in σ, at least $k-1$ of the items do not belong to π.

Later, we show how to estimate the sum of lengths of these subsequences and how to use this to bound $ED(\sigma)$.

3.2 An improved estimator We now design an estimator which gives a 2-approximation to $LIS(\sigma)$. Let σ be a sequence of length n. Let $inv(i)$ be the set of indices $j < i$ such that $\sigma(j) > \sigma(i)$. We say $R \subset [n]$ is a *red set* for σ if $\forall i \in R$ at least one of the following is true:

(i) $i-1 \in inv(i)$, or,

(ii) there is an interval $I = [j, i-1]$ such that $|inv(i) \cap I| > |R \cap I|$.

When $i \in R$ we say i is a red index; every red index has a witness in the form of another index (if (i) is satisfied) or an interval (if (ii) is satisfied). We call the red set R *total* if $\forall i \notin R$, i does not have a witness. The above definition is similar to the definition used in [3] except that the membership of an index in R depends not only on inversions but also on the number of red indices to its left in σ. Our main observation is the following lemma, which links the size of the red set to the distance to monotonicity.

LEMMA 3.2. *Let R be a red set for the sequence σ. We have $|R| \leq ED(\sigma)$. Moreover if R is total then $|R| \geq \frac{1}{2}ED(\sigma)$.*

Proof. For the first part, suppose the set $R = \{j_1, ..., j_t\}$ is a red set for σ. Let $G = (V, E)$ be a graph where $V = \{\sigma(1), ..., \sigma(n)\}$. We now introduce an inductive procedure that defines the edge set E. Initially $E = \emptyset$. We scan σ from left to right, and for every index i which is in R, find some $k < i$ such that $k \in inv(i)$ and the indegree of $\sigma(k)$ is zero in G. We then add a directed edge $(\sigma(i), \sigma(k))$ to G. By induction over the indices in R, we prove that this procedure is possible at every step. The base case is trivial and we can add an arbitrary edge $(\sigma(j_1), \sigma(j'_1))$ to E when $j'_1 \in inv(j_1)$. Suppose the claim is true for up to j_{r-1}. Consider the index j_r. Since $j_r \in R$, by definition, we are in one of two cases. The first case is when $j_r - 1 \in inv(j_r)$, in which case we can add the edge $(\sigma(j_r), \sigma(j_r - 1))$ to E. The second is when there exists an interval $I = [l, j_r - 1]$ such that $|inv(j_r) \cap I| > |R \cap I|$. Suppose for $\forall x \in I \cap inv(j_r)$, indegree$(x)$ is nonzero. This is not possible since the edges only originate from the vertices that belong to R and this implies that $|R \cap I| \geq |I \cap inv(j_r)|$ which contradicts our assumption. Therefore we will be able to add an edge starting from j_r as well.

Now consider the graph $G = (V, E)$ at the end of the above process and make the edges undirected (with a little notational abuse, we call the new graph G as well). It is easy to observe that G is composed of a set of disjoint paths. Consider any maximal path $p = (\sigma(j_i), ..., \sigma(j_{i+k}))$; p represents a decreasing subsequence of length k. Additionally, we have $\{j_{i+1}, ..., j_{i+k}\} \in R$. By using Lemma 3.1, we can conclude that $|R|$ is a lower bound for $ED(\sigma)$.

Now we prove the second part of the lemma. Let R be a total red set for σ. We define an iterative pruning procedure that deletes at most $2|R|$ elements from σ and leaves a sorted sequence, similar to that used in [3] for showing the lower bound. First let $i = n + 1$ and $\sigma(n + 1) = m$ where m is larger than all of the elements in the sequence. Then iteratively do the following until σ is exhausted: If $i - 1 \notin R$ then proceed to $i - 1$ and repeat. If $i - 1 \in R$; let j be the largest index such that $j < i$ and $j \notin R \cup inv(i)$. Prune the interval $[j + 1, i - 1]$, proceed to j, and repeat.

It is easy to see that at the end of this procedure the resulting sequence is sorted. To bound the number of elements pruned, observe that when we delete an interval, at least half of the elements in the interval belong to R. Thus in total we delete at most $2|R|$ elements. The proof follows.

3.3 Approximating the estimator We now show a deterministic algorithm for approximating our estimator. Even though the estimator has similarities to that in [3], due to its definition involving the comparison of $|inv(i) \cap I|$ with $|R \cap I|$, we must design a novel algorithm to compute it. Notice that the exact computation of this quantity can be quite costly if the two quantities are close to each other, or are very small. We show below that an efficient deterministic algorithm which gives an inexact estimate of the quantities suffices to construct a good approximation algorithm. To be precise, we use an algorithm, which, instead of comparing $|inv(i) \cap I|$ and $|R \cap I|$, checks for two conditions: whether the number of inversions is in the majority and the red indices are far from being majority in an interval. If the test for i passes for any one interval then we make the index i a red index. The detected red set might not be total, however we show that it is large enough to give us a $2 + O(\epsilon)$ approximation.

The majority test. Given some x and an interval I, we want to check whether the number of elements in I that are larger than x is more than $|I|/2$ or not. One can perform this test by comparing x with the median of the elements in I. Since we only require a relaxed version of the majority test, we can use an approximate median for this purpose. The problem of finding the approximate median (and other quantiles) deterministically in a stream is well studied in the literature. Since we need to obtain the approximate median for all widow sizes over the stream, we use a special algorithm from Lu et al [7], whose properties we describe below.

Let S be a set with N elements. A ϕ-quantile ($\phi \in (0, 1]$) of S is the element of rank $\lceil \phi N \rceil$. An element is said to be ϵ-approximate ϕ-quantile if its rank is in $[\lceil (1 - \epsilon)\phi N \rceil, \lceil (1 + \epsilon)\phi N \rceil]$. The below theorem is a modified version of the theorem of [7]

THEOREM 3.1. *There is a deterministic streaming algorithm which, given an input stream of length N, using $O(\frac{1}{\epsilon^2} \log^2(\epsilon N))$ space forms a sketch of the stream and can output on demand, using this sketch, an ϵ-approximate quantile of the n most recent elements of the given stream in for any n.*

Let A be the algorithm described in the above theorem. Our algorithm will be making queries to A as follows. Let the output of $A(S, k, \phi)$ be an ϵ-approximation for the ϕ-quantile of the k most recent elements in stream S. While going over the sequence, we assume that we generate a binary sequence that represents the red elements detected so far. let R' be the sequence of these elements, i.e. $R'(i) = 1$ if and only if the algorithm has identified i^{th} element as a red element. We apply the subroutine A to both the input stream (σ) and the sequence R'. We present the algorithm below for a particular interval size $i-j$.

```
Procedure $RedTest(j,i)$
    1. Let $a = A(\sigma, i-j, \frac{1}{2}-\epsilon)$.
    2. If $a \leq \sigma(i)$ return FALSE.
    3. Let $a' = A(R', i-j, \frac{1}{2}+\epsilon)$.
    4. If $a' = 0$ then return TRUE otherwise return FALSE.
```

In the following lemma, we analyze the procedure RedTest.

Lemma 3.3. *Let $I = [j, i-1]$. If the majority of elements in I are not in $inv(i)$ then $RedTest(j,i)$ returns FALSE. If more than $(\frac{1}{2}+2\epsilon)|I|$ of the elements in I are in $inv(i)$ and the number of (detected) red elements in I is less than $(\frac{1}{2}-2\epsilon)|I|$ then $RedTest(j,i)$ returns $TRUE$.*

Proof. First part: if the majority of the elements in I are not in $inv(i)$ then the median of I is at most $\sigma(i)$ and since the rank of a is in the range $(\frac{1}{2}-2\epsilon, \frac{1}{2})|I|$ then we should have $a \leq \sigma(i)$; the test returns FALSE. Second part: since more than $(\frac{1}{2}+2\epsilon)$ fraction of I are in $inv(i)$ and the rank of a is in the range $(\frac{1}{2}-2\epsilon, \frac{1}{2})|I|$, it follows that $a > \sigma(i)$. Also since the rank of a' is in the range $(\frac{1}{2}, \frac{1}{2}+2\epsilon)|I|$, we should have $a' = 0$ and the test returns TRUE.

We now give the main algorithm.

```
Main Algorithm. Upon arrival of element $\sigma(i)$ do the following.
1. For each $j \in [1, i-1]$, do $RedTest(j,i)$. If there exists $j$ such that $RedTest(j,i) = TRUE$ then let $d = d+1$ and $R'(i) = 1$; otherwise $R'(i) = 0$
2. Proceed to $i+1^{st}$ element.
At the end, output $d$.
```

Lemma 3.4. *At the end of Main Algorithm we have $(\frac{1}{2}-O(\epsilon))ED(\sigma) \leq d \leq ED(\sigma)$.*

Proof. Let $i \in R'$. By Lemma 3.3 there exists an interval $I = [j, i-1]$ such that $|inv(i) \cap I| > |R' \cap I|$. It follows that this interval is a witness for i and hence R' is a red set for σ. By Lemma 3.2 $d = |R'| \leq ED(\sigma)$.

Now we show the lower bound. The set R' is not necessarily total. However we show that it is big enough to be bigger than $(\frac{1}{2}-2\epsilon)ED(\sigma)$. We use the same pruning procedure that we used in the proof of Lemma 3.2. Consider the point where $i \notin R'$ and we eliminate the interval $I = [j, i-1]$. By definition of pruning procedure, the elements in I are either in $inv(i)$ or in R'. Suppose $|I \cap R'| < (\frac{1}{2}-2\epsilon)$. Then we should have $|inv(i) \cap I| > (\frac{1}{2}+2\epsilon)$ and hence by Lemma 3.3 $RedTest(j,i)$ should output TRUE. This contradicts $i \notin R'$. It follows that in every interval that we delete, the fraction of red elements is at least $\frac{1}{2}-2\epsilon$ and hence in total we delete at most $2+O(\epsilon)|R'|$ elements from the sequence and we get a sorted subsequence. This proves the lower bound.

Improving the running time. The running time of the above algorithm is $\tilde{O}(n)$ per-item because the algorithm checks every interval. An observation shows that for some small enough $\epsilon_1 < 1$, checking $O(\frac{1}{\epsilon_1}\log n)$ number of intervals is enough. To see this, note that an ϵ_1-approximate ϕ-quantile of an interval with length $|I|$ is also an $(\epsilon_1+\epsilon_2)$-approximate ϕ-quantile for all intervals with lengths $|I|+1, ..., (1+\epsilon_2)|I|$. Hence with the appropriate choice of ϵ_1 and ϵ_2 (where $\epsilon_1+\epsilon_2 < \epsilon$) and by checking only intervals of length $1, 2, ..., (1+\epsilon')^i, (1+\epsilon')^{i+1}, ..., n$, we can obtain an ϵ-approximate quantile for every interval. Given this, we can state the following theorem.

Theorem 3.2. *Given a sequence σ of length n, there a deterministic streaming algorithm that outputs a $2+O(\epsilon)$ approximation of $ED(\sigma)$ and uses space $O(\frac{1}{\epsilon^2}\log^2(\epsilon n))$ and $O(\frac{1}{\epsilon^2}\log^3 n)$ per-item running time.*

4 Space lower bound for approximating LIS

In this section we prove the conjecture of Gopalan *et al.* [3] which states a lower bound of $\Omega(\sqrt{n})$ for the space complexity of streaming algorithms that approximate $LIS(\sigma)$ within a small constant factor. Our proof is based on multi-player communication complexity combined with a direct-sum approach that is suggested in [3]. Before we begin with detail of the proof, we briefly describe the idea of [3]. Gopalan *et al.* defined a specific distribution (μ) of sequences of length t^2 such that for each sequence $\sigma \in \mu$ either $LIS(\sigma) = t$ or $LIS(\sigma) = (1+\epsilon)t$ where ϵ is a positive constant less than one. Gopalan *et al.* showed that the problem of distinguishing between the two types of the sequences in μ can be reduced to function g which is defined as the disjunction (OR) of t instances of identical primitive functions $(g = \vee_{i=1}^{t} h_i(\sigma_i))$. The input to the primitive functions, $\sigma_1, ..., \sigma_t$, are disjoint interleaving subsequences of σ which means that $\sigma_i = \sigma(j_1)\sigma(j_2)...\sigma(j_t)$ where $j_k = i+(k-1)t$. This definition helps that one can use a t-player one-way game to compute g where i-th player holds the i-th elements of the subsequences $\sigma_1, ..., \sigma_t$. The idea behind this reduction is that computing a lower bound for communication complexity of

g through a direct-sum approach would give us a lower bound for streaming algorithms that distinguishes the inputs in μ and hence it applies to approximating LIS in general.

Gopalan *et al.* used the above setting but they were not succeeded to prove a bit-communication complexity lower bound, instead they obtained a lower bound of $\Omega(t^2)$ for the total communication complexity of g through a restricted class of protocols. This result implies a lower bound of $\Omega(t)$ for approximating LIS through the so called Natural algorithms which are one-pass streaming algorithms that can only store some subset of input symbols and nothing else with additional few auxiliary bits.

Here we use the framework of Gopalan *et al.* [3] and we derive a bit-communication complexity lower bound. The idea behind our proof is defining a new primitive function h which admits a large fooling set. We show the existence of such fooling set through probabilistic method. Finally we show that the fooling set for h can be extended to create a large fooling set for g.

4.1 Some definitions and basic facts Let U be some finite universe. Let $f : U^t \rightarrow \{0,1\}$ and let X be a $k \times t$ matrix. Let $X_{ij} \in U$ denote the element at coordinate (i,j) in X. Let X_j be the set of elements that appear in the j^{th} column of X. We define $Y(X) = \{y \in U^t | \forall i \in [t] \;\; y_i \in X_i\}$ and we call it the span of the matrix X. [1] Now we define the notion of a *k-fooling set* for function f which is a generalization of a standard fooling set definition (see [5]).

DEFINITION 4.1. *Let $S \subset U^t$. For some positive integer k, S is a k-fooling set for f iff $f(x) = 0$ for each $x \in S$ but for each subset of S' of S with cardinality k, the span of S' contains a member y where $f(y) = 1$.*

We define the t-player one-way game $\mathcal{G}(f,t)$ which consists of players $P_1,...,P_t$ such that P_i holds the element $a_i \in U$ and the goal of the game is to compute $f(a)$ where $a = (a_1, a_2, ..., a_t)$. The course of the communication is from P_i to P_{i+1} (here $t+1 = 1$) where the players speak in turn. We emphasize that our lower bounds apply to the stronger model where the players speak in turn - in natural order of $1,...,n$- and they write exactly one message on a board that is visible to all. The game ends when a player announces the answer.

We define $CC_t^{tot}(f)$ (total communication complexity of f) as the minimum number of bits required to be sent by the players (or the total length of the messages written on the shared board) in every deterministic communication protocol that always outputs a correct answer for the game $\mathcal{G}(f,t)$. The following fact is folklore and it is an easy consequence of the fact that a k-fooling set requires at least $\log(|S|/(k-1))$ monochromatic rectangles to be covered (see chapters 1 and 6 from [5]).

FACT 4.1. *Let S be a k-fooling set for f. We have $CC_t^{tot}(f) \geq \log(|S|/(k-1))$.*

Let P be a correct protocol for $\mathcal{G}(f,t)$ and let $max(P)$ be the maximum number of bits sent by a player in P taken over all inputs of f. We define $CC_t^{max}(f)$ (maximum communication complexity of f) as $\min\{max(P_i)\}$ where P_i ranges over all correct protocols for $\mathcal{G}(f,t)$. It is clear that $C_t^{max}(f) \geq \frac{1}{t}CC_t^{tot}(f)$.

REMARK 4.1. *In this section, we assume that an increasing subsequence does not contain repetition.*

4.2 The proof of the conjecture We start with the definition of the primitive function h. Let t be an even integer. Let σ be a sequence of length t where $\forall i, \sigma(i) \in T = \{0,...,t\}$ and nonzero elements of σ form a strictly increasing sequence. In other words, σ represents the characterization vector of some subset of $[t]$. Here we assume that the elements $\{1,t\}$ are always appear in σ. We define $h(\sigma) = 0$ if no consecutive nonzero elements appear in σ. It is clear that in this case $LIS(\sigma) \leq \frac{t}{2} + 1$. We define $h(\sigma) = 1$ when σ has an increasing subsequence of length at least αt where α is some constant greater than $\frac{1}{2}$. Note that we restrict the inputs of h to the sequences having one of the above properties.

Now similar to [3], we define the function g which is the OR of t parallel and independent copies of h. Let $h_1, h_2, ..., h_t$ be t identical copies of h. [2] Let B be a $t \times t$ matrix where B_i is the i^{th} row of B and it represents an input for function h_i. We assume that the universe of the rows of B are disjoint except for the zero element which is shared by all of them. In another words, we assume that B_i is a sequence that represents a subset of $[(i-1)t+1, it]$ in the manner we described above. Now we define $g(B) = h_1(B_1) \vee h_2(B_2) \vee ... \vee h_t(B_t)$.

Let s be the sequence of length t^2 which is formed by the concatenation of the columns of B in the natural order. Formally speaking let $C_j = B_{1j}B_{2j}...B_{tj}$; we let $s = C_1C_2...C_t$. Now consider an increasing sequence in s and trace the elements of this sequence in B.

[1] In some places we use the span operator for sets of tuples (or sequences); in these situation the set of tuples or sequences are regarded as a matrix where rows are the tuples (sequences).

[2] Their description are the same though their domain are different.

For now lets assume that the increasing subsequence should not contain the zero element (we later remove this assumption). It is easy to see that any increasing sequence in s can only take a right or downward direction in the matrix B. This implies that the length of the longest increasing sequence in S is at most $2t-1$. For the case when $g(B) = 0$, since we do not have two consecutive nonzero elements in a row, the length of each step along a row is at least two. This implies that $LIS(s)$ is at most $\frac{3}{2}t$ in this case. Now consider the case when $g(B) = 1$. Here we have a row, say B_j, that contains an increasing subsequence of length at least αt. In this case, the increasing subsequence can take the first column down to the row B_j and then take the increasing subsequence in B_j up to the last column and then go down the rest of the last column. Therefore when $g(B) = 1$, we have that $LIS(s) \geq (1+\alpha)t$ (recall that we defined $\alpha > \frac{1}{2}$).

Now consider the game $\mathcal{G}(g, t)$ where P_i holds the i^{th} column of B. Let A be a deterministic streaming algorithm that approximates the length of LIS within a factor better than $1 + \frac{1}{2}(\alpha - \frac{1}{2})$ using $p(t)$ space. We show that we can use A as a deterministic protocol for deciding g with $p(t)$ maximum communication. First suppose each player replaces the zeros in her input with distinct negative numbers in a way the negative numbers form a decreasing subsequence in the big sequence. Let s' be the new sequence. Consider that any increasing subsequence in s' can only take one of the negative numbers and hence the above argument regarding the output of g and $LIS(s')$ is still valid. It follows that A can be used to distinguish $g(B) = 0$ from $g(B) = 1$. From this we get the following lemma.

Lemma 4.1. *There exists some $\epsilon < 1$ such that any deterministic streaming algorithm for approximating LIS within a $(1+\epsilon)$ factor requires at least $CC_t^{max}(g)$ space.*

Now in order to prove a lower bound for $CC_t^{max}(g)$, we first prove a lower bound on $CC_t^{tot}(h)$ which is obtained by a fooling set argument. We then show that the product of fooling sets of h_i's is actually a fooling set for g. Finally from Observation 4.1 we derive a lower bound for $CC_t^{tot}(g)$; this value divided by t gives us a lower bound for $CC_t^{max}(g)$.

Lemma 4.2. *Let $k > 320$. There is a k-fooling set for function h of size c^t where $c > 1$.*

Proof. Note that a k-fooling set for h consists of sequences of length $t+2$ such that no consecutive nonzero elements appear in those sequences and in addition to that the span of any k sequences contains a sequence with at least αt nonzero elements. For this it is enough to find a collection of an exponential number of distinct subsets of $T = 1, ..., t$, say F, such that no $s \in F$ contains two consecutive members of T and the union of any k members of F covers at least α fraction of $[t]$ (remember that α is constant bigger than $\frac{1}{2}$). Consider that since we have assumed that elements $\{1, t\}$ appear in any input for h, for our purpose, we require to pick subsets from the set 3,...,t-2. For the sake of simplicity lets assume that we choose our subsets from $[t]$ as we can simply extend the size of the input sequences by a constant number without harming the final bounds.

We use probabilistic method to prove the existence of such collection of inputs. Consider the random process of picking subsets of $[t]$ where each element is picked independently with probability p. Suppose we pick M subsets in this manner. Consider a random subset. Let $b(t)$ be the probability that no two consecutive members of $[t]$ are picked for this subset. When a subset satisfies this property we refer to it as a good subset. Using induction we can show that $b(t) \geq (1-p^2)^t$. To see this, consider the recurrence $b(t) = (1-p)b(t-1) + p(1-p)b(t-2)$ where $b(1) = 1$ and $b(2) = 1 - p^2$. Now let the random variable Z represent the number of good subsets. It follows that $E(Z) = (1-p^2)^t M$.

Now we need to find a upper bound on the probability that the union of k random subsets cover less than α fraction of the universe. Let $J_1, ..., J_k$ be k random subsets. Note that each element is included in $J = J_1 \cup J_2 \cup ... \cup J_k$ with probability $\gamma = 1 - (1-p)^k$. Hence the expected size of J is γt. Since the inclusion of the elements are independent, we can use Chernoff bound to bound the probability that $|J| < (\gamma - \epsilon)t$ where $\epsilon \in (0, \gamma - \frac{1}{2})$.

$$Pr(|J| < (\gamma - \epsilon)t) \leq \delta = e^{(\frac{-\epsilon^2\gamma}{2})t}.$$

We now set $p = \frac{1}{k}$. Since $\gamma > 1 - 1/e$ setting $\alpha = \gamma - \epsilon$ for some $\epsilon \in (0, \frac{1}{2} - \frac{1}{e})$ satisfies our requirement for α. Now since we require the union of any k random subsets to cover α fraction of the universe, using the union bound, if $\binom{M}{k}\delta < 1$ then with a positive probability there are $(1-p^2)^t M$ many good subsets that form a k-fooling set for h. By plugging in the values we get $(1 - \frac{1}{k^2})^t (\frac{1}{\delta})^{\frac{1}{k}} = (e^{\frac{\epsilon^2\gamma}{2k}}(1 - \frac{1}{k^2}))^t$ many good subsets. One can inspect that for some big enough k (depending on ϵ) $e^{\frac{\epsilon^2\alpha}{2k}}(1 - \frac{1}{k^2})$ is bigger than one. (For $\epsilon = 0.1$, taking $k > 320$ works.) This completes the proof.

The following lemma shows that one can construct a large fooling set for g from a fooling set for h.

Lemma 4.3. *Let F_i be a k-fooling set for h_i. Then*

$F = (F_1 \times F_2 \times ... \times F_t)$ *is a* k^t*-fooling set for* g.

Proof. Note that the members of F are tuples of length t such that each coordinate is itself a sequence of length t (here the coordinates are rows of the input matrix for g). First of all, by the definition of fooling set for h and the definition of g, we have that $g(B) = 0$ for all $B \in F$. Now let F' be an arbitrary collection of k^t members of F. Let H_i be the set of elements from F_i that appear in F'. Note that H_i is a subset of inputs for h_i. Since the size of F' is at least k^t there exists some $j \in [t]$ such that $|H_j| \geq k$. Now let W be some subset of k members of F' that cover H_j. Here we regard W as a set of k matrices of order $t \times t$. (Note that g is defined as the OR of h's applied to the rows of those matrices). Consider $Y(W)$. Note that in each $B \in Y(W)$, the i^{th} column of B is picked from the i^{th} column of the one of the matrices in W. From the fact that F_j is a fooling set for h_j, it follows that there exists $y \in Y(H_j)$ (the span of H_j) such that $h_j(y) = 1$. It implies that there exists some $B \in Y(W)$ such that $g(B) = 1$. We conclude that F is a k^t-fooling set for g.

To finish the proof of the main theorem, consider that $CC_t^{tot}(g) \geq \log \frac{|F|}{k^t} \geq \log \frac{|F_i|^t}{k^t}$. From the result of Lemma 4.2 and the fact that $CC_t^{max}(g) \geq \frac{1}{t} CC_t^{tot}(g)$, it follows that $CC_t^{max}(g) = \Omega(t)$. Taking $|s| = t^2 = n$, we derive the following theorem.

THEOREM 4.1. *There exists some* $\epsilon < 1$ *such that any* $O(1)$*-pass deterministic streaming algorithm for approximating LIS within* $1 + \epsilon$ *factor requires at least* $\Omega(\sqrt{n})$ *space.*

5 Acknowledgement

The authors would like to thank Cenk Sahinalp, Ravi Kumar and Gabor Tardos for their useful comments and suggestions. Special thanks to Cenk Sahinalp for his help on the writing of an earlier draft of this paper.

References

[1] M. Ajtai, T.S. Jayram, Ravi Kumar, D. Sivakumar. Approximate counting of inversions in a data stream. *Proceedings on 34th Annual ACM Symposium on Theory of Computing, STOC 2002.*

[2] A. Gal, P. Gopalan. Lower bounds on streaming algorithms for approximating the length of the longest increasing subsequence. *In 48th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2007.*

[3] P. Gopalan, T. S. Jayram, R. Krauthgamer, R. Kumar, Estimating the sortedness of a data stream. *In Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2007.*

[4] A. Gupta, F. Zane. Counting inversions in lists. *In Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms. SODA 2003.*

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

[6] D. Liben-Nowell, E. Vee, and A. Zhu. Finding longest increasing and common subsequences in streaming data. *Journal of Combinatorial Optimization*, 11(2):155-175, 2006.

[7] X. Lin, H. Lu, J. Xu, J.X. Yu. Continuously maintaining quantile summaries of the most recent N elements over a data stream. *In ICDE 2004.*

[8] S. Muthukrishnan. Data Streams: Algorithms and Applications. *Now Publishers Inc., 2005.*

[9] D. Woodruff, X. Sun. The Communication and Streaming Complexity of Computing the Longest Common and Increasing Subsequences. *In Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2007.*