A polylogarithmic space deterministic streaming algorithm for approximating distance to monotonicity *

Timothy Naumovitz† Michael Saks‡

October 8, 2014

Abstract

The *distance to monotonicity* of a sequence of n numbers is the minimum number of entries whose deletion leaves an increasing sequence. We give the first deterministic streaming algorithm that approximates the distance to monotonicity within a $1+\varepsilon$ factor for any fixed $\varepsilon > 0$ and runs in space polylogarithmic in the length of the sequence and the range of the numbers. The best previous deterministic algorithm achieving the same approximation factor required space $\Omega(\sqrt{n})$ [9]. Previous polylogarithmic space algorithms were either randomized [10], or had approximation factor no better than 2 [8].

We also present space lower bounds for this problem: Any deterministic streaming algorithm that gets a $1+\varepsilon$ approximation requires space $\Omega(\frac{1}{\varepsilon}\log^2(n))$ and any randomized algorithm requires space $\Omega(\frac{1}{\varepsilon}\frac{\log^2(n)}{\log\log(n)})$.

1 Introduction

In the Longest Increasing Subsequence (LIS) problem the input is a function (array) $f : [n] \to [m]$ (where $[n] = \{1, \ldots, n\}$) and the problem is to determine $\mathbf{LIS}(f)$, the size of the largest $I \subseteq [n]$ such that the restriction of f to I is an increasing function.

The distance to monotonicity of f, $\mathbf{DM}(f)$ is defined to be $n - \mathbf{LIS}(f)$, which is the number of entries of f that must be changed to make f an increasing function. Clearly the algorithmic problems of computing $\mathbf{DM}(f)$ and $\mathbf{LIS}(f)$ are essentially equivalent as are the problems of approximating these quantities within a specified additive error. However, there is no such obvious correspondence between the problems of approximating $\mathbf{DM}(f)$ and $\mathbf{LIS}(f)$ to within a constant *multiplicative* factor. In fact we see from this paper that there is a significant difference in the difficulty of approximating these two problems, as least in some settings.

These problems, both the exact and approximate versions, have attracted attention in several different computational models, such as sampling, streaming, and communication models. Following several recent papers, we study this problem in the streaming model, where we are allowed one sequential pass over the input sequence, and our goal is to minimize the amount of space used by the computation.

Previous Results. The exact computation of $\mathbf{LIS}(f)$ and $\mathbf{DM}(f)$ can be done in $O(n\log(n))$ time using a clever implementation of dynamic programming [1; 2; 3], which is known to be optimal [4]. In the streaming setting, it is known that exact computation of **LIS** and **DM** require $\Omega(n)$ space even when randomization is used [9].

The most space efficient multiplicative approximation for $\mathbf{LIS}(f)$ is the deterministic $O(\sqrt{n})$ space algorithm [9] for computing a $(1+\varepsilon)$-multiplicative approximation. This space is essentially optimal [8; 7] for deterministic algorithms. Whether randomization helps significantly for this problem remains a very interesting open question.

In contrast, $DM(f)$ has very space efficient approximations algorithms. A randomized multiplicative $(4+\varepsilon)$-approximation using $O(\log^2(n))$ space was found by [9]. This was improved upon by [10] with a $(1+\varepsilon)$-multiplicative approximation using $O(\frac{1}{\varepsilon}\log^2(n))$ space. In the deterministic case, [8] gave a polylogarithmic space algorithm giving a $2+o(1)$ factor approximation, but prior to the present paper the only deterministic algorithm known that gave a $(1+\varepsilon)$-factor approximation for arbitrary $\varepsilon > 0$ was an $O(\sqrt{n})$-space multiplicative approximation given by [9]. There have been no significant previous results with regard to lower bounds for this problem in either the randomized or deterministic case.

Our Contributions. We give the first deterministic streaming algorithm for approximating $DM(f)$ to within an $1+\varepsilon$ factor using space polylogarithmic in n and m. More precisely, our algorithm uses space $O(\frac{1}{\varepsilon^2}\log^5(n)\log(m))$.

The improvement in the approximation factor from $2+o(1)$ to $1+\varepsilon$ is qualitatively significant because a factor 2 approximation algorithm to $\mathbf{DM}(f)$ can't necessarily distinguish between the case that $\mathbf{LIS}(f) = 1$ and $\mathbf{LIS}(f) = n/2$, while a $1+\varepsilon$ approximation can approximate $\mathbf{LIS}(f)$ to within an additive εn term.

*Supported in part by NSF under grants CCF-1218711 and CCF-0832787.

†Department of Mathematics, Rutgers University.

‡Department of Mathematics, Rutgers University.

Our algorithm works by maintaining a small number of small *sketches* at different scales during the streaming process. The main technical challenge in the analysis is to show that the size of the sketches can be controlled while maintaining the desired approximation quality.

We also establish lower bounds for finding $1+\varepsilon$ multiplicative approximations to **DM**. Using standard communication complexity techniques we establish an $\Omega(\frac{1}{\varepsilon}\log^2(n))$ space lower bound for deterministic algorithms and an $\Omega(\frac{1}{\varepsilon}\frac{\log^2(n)}{\log\log(n)})$ space lower bound for randomized algorithms. The reduction maps the streaming problem to the one-way communication complexity of the "Greater Than" function.

2 Preliminaries

For a positive integer r, $[r]$ denotes the set $\{1,...r\}$. Throughout the paper, f denotes a fixed function from $[n]$ to $[m]$, which we refer to as the *input sequence*. We will also refer to an element of the domain of f as an *index*, and an element of the range of f as a *value*.

- A subset J of $[n]$ is *f-monotone* if for all $j, j' \in J$, $j < j'$ implies $f(j) < f(j')$.
- The *distance to monotonicity of f* is n minus the size of the largest f-monotone subset.
- For $l, r \in [m] \cup \{0\}$, an *(l,r)-monotone* subset J is an f-monotone subset satisfying $f(j) \in (l, r]$ for all $j \in J$.
- For $I \subseteq [n]$, the $(m+1) \times (m+1)$ matrix DM_I is defined by $DM_I(l,r)$ is equal to $|I|$ minus the size of the largest (l,r)-monotone subset of I. Observe that if $l \geq r$, then $DM_I(l,r) = |I|$.

Our streaming algorithm will try to approximate $DM_{[n]}(0,m)$ (i.e. the distance to monotonicity of the entire sequence). To do this, it will maintain a small set of small matrices that each provide some approximate information about the matrices DM_I for various choices of I. This motivates the next definitions:

- A *DM-sketch* is a triple (L,R,D) where $L, R \subseteq [m] \cup \{0\}$ and D is a nonnegative matrix with rows indexed by L and columns indexed by R. We sometimes refer to the matrix D as a DM-sketch, leaving L and R implicit.
- A DM sketch (L,R,D) is *well behaved* if for any $l, l' \in L$ and $r, r' \in R$ with $l \leq l'$ and $r' \leq r$, it holds that $D(l,r) \leq D(l',r')$.
- A DM-sketch is said to be *valid for interval I* if $|I| \geq D(l,r) \geq DM_I(l,r)$ for all $l \in L$ and $r \in R$.
- For $i \in [n]$, the *trivial sketch* for i is the DM-sketch with $L = \{f(i)-1\}, R = \{f(i)\}$, and $D = [0]$. Note that the trivial sketch for i is trivially well behaved and valid for i.
- The size of a DM sketch (L,R,D) is $\max(|L|,|R|)$.

Given a valid DM-sketch (L,R,D) for I, we want to obtain an estimate for the $(m+1)\times(m+1)$ matrix DM_I. Observe that, for any I, $([0,m],[0,m],DM_I)$ is a well behaved and valid DM sketch for I. For $l, r \in [m] \cup \{0\}$ and $l' \in L$ and $r' \in R$ with $l \leq l'$ and $r' \leq r$, we have $DM_I(l,r) \leq DM_I(l',r') \leq D(l',r')$. This motivates the following definitions:

- For $l, r \in [m] \cup \{0\}$, the *L-ceiling* of l, denoted by $\bar{l}^L$ is the smallest element $l' \in L \cup \{m\}$ such that $l \leq l'$. Similarly, the *R-floor* of r, denoted by $\underline{r}_R$ is the largest element $r' \in R \cup \{0\}$ such that $r \geq r'$.
- Given the DM-sketch (L,R,D) for I, the *natural estimator of DM_I induced by D* is the matrix D^* given by:
$$D^*(l,r) = D(\bar{l}^L, \underline{r}_R)$$
Observe that $([m] \cup \{0\}, [m] \cup \{0\}, D^*)$ is a DM-sketch.
- (L,R,D) is *$(1+\delta)$-accurate for interval I* if for every $l, r \in [m] \cup \{0\}$, $D^*(l,r) \leq (1+\delta)DM_I(l,r)$.

PROPOSITION 2.1. *Let D be a DM-sketch, I an interval, and D^* be the natural estimator of DM_I induced by D. If D is well behaved and valid for I, then so is D^*.*

Proof. First, note that the well-behavedness of D^* follows from the fact that if $l \leq l'$, $r' \leq r$, then $\bar{l}^L \leq \bar{l'}^L$ and $\underline{r'}_R \leq \underline{r}_R$. Let $l, r \in [m] \cup \{0\}$. The fact that $D^*(l,r) \leq |I|$ follows from the validity of (L,R,D), so it remains to show $D^*(l,r) \geq DM_I(l,r)$. We know that $\bar{l}^L \geq l$ and $\underline{r}_R \leq r$ by definition. As a result, any $(\bar{l}^L, \underline{r}_R)$-monotone subset of I is an (l,r)-monotone subset of I, so we have $DM_I(l,r) \leq DM_I(\bar{l}^L, \underline{r}_R)$. Since $D^*(l,r) = D(\bar{l}^L, \underline{r}_R) \geq DM_I(\bar{l}^L, \underline{r}_R)$ by the validity of (L,R,D), we are done. □

3 A Polylogarithmic Space Streaming Algorithm

As mentioned, at each step j, our streaming algorithm will maintain a small number of small sketches for various subintervals of $[1,j]$. Our algorithm involves the repeated use of two main building blocks: an algorithm MERGE and an algorithm SHRINK.

The algorithm MERGE takes as input an interval I of even size split into its two halves I_1 and I_2 and DM-sketches (L_1,R_1,D_1) for I_1 and (L_2,R_2,D_2) for I_2 and

outputs a DM-sketch (L, R, D) for I. It does this in the following very simple way:

- $L = L_1 \cup L_2$
- $R = R_1 \cup R_2$
- D is defined, for $l \in L$ and $r \in R$ by:

$$D(l,r) = \min_{l \le z \le r} D_1^*(l,z) + D_2^*(z,r),$$

where D_1^* is the natural estimator for $DM_{I_1}(\cdot,\cdot)$ induced by D_1 and D_2^* is the natural estimator for $DM_{I_2}(\cdot,\cdot)$ induced by D_2.

The algorithm SHRINK takes as input a DM-sketch (L, R, D) and outputs a DM-sketch (L', R', D') where $L' \subseteq L$, $R' \subseteq R$ and D' is the restriction of D to $L' \times R'$. It takes a parameter $\gamma > 0$.

The goal of the algorithm SHRINK is to choose (L', R', D') as small as possible while ensuring that, for any $l, r \in [m] \cup \{0\}$, $D'^*(l,r)$ is not too much bigger than $D^*(l,r)$. To find $L' \subseteq L$ and $R' \subseteq R$, our algorithm greedily omits values from L and R without destroying the property

$$\forall l, r \in [m] \cup \{0\}, D^*(l,r) \le D'^*(l,r) \le (1+\gamma)^2 D^*(l,r).$$

The algorithm SHRINK first determines L' and then determines R'. Let $l_1 < \cdots < l_{|L|}$ be the values in L. We construct a sequence $x_1 \le \hat{x}_1 \le x_2 \le \hat{x}_2 \le x_3, ..., \hat{x}_{s-1} \le x_s$ iteratively as follows. Let $x_1 = l_1$. For $k \ge 1$, having defined $x_1, \hat{x}_1, ..., \hat{x}_{k-1}, x_k$, if $x_k = l_{|L|}$, stop. Otherwise, let $\hat{x}_k = l_i$ where i is the largest index less than $|L|$ such that

$$\forall r \in R, D(l_i, r) \le (1+\gamma) D(x_k, r) \tag{3.1}$$

and let $x_{k+1} = l_{i+1}$. Set $L' = \{x_1, \hat{x}_1, x_2, \hat{x}_2, x_3, ..., \hat{x}_{s-1}, x_s\}$. Now let D'' be the submatrix of D induced by the rows of L', giving us an intermediate sketch (L', R, D''). Starting from D'', we perform an analogous construction for R', defining y_1 to be the largest value of R, and working our way downwards (so y_t will be the smallest value of R). We get $R' = \{y_1, \hat{y}_1, y_2, \hat{y}_2, y_3, ..., \hat{y}_{t-1}, y_t\}$, and let D' be the submatrix of D'' induced by the columns labeled by R'. This yields another DM sketch (L', R', D') for I. The DM sketch (L', R', D') will be the sketch that SHRINK outputs.

Armed with the procedures MERGE and SHRINK, we can now describe our deterministic streaming algorithm DMAPPROX for approximating distance to monotonicity. DMAPPROX requires a parameter $\gamma > 0$. (The choice of γ will be $\ln(1+\varepsilon)/(2\log(n))$ where ε is the desired approximation factor.)

We first describe a version of our algorithm that is not in the streaming model, and then convert it into a streaming algorithm, which will be called DMAPPROX. Assume without loss of generality that $n = 2^d$ for an integer d. Consider the rooted binary tree whose nodes are subintervals of $[n]$ with $[n]$ at the root, and for each interval I of length greater than 1, its left and right children will be the first and second halves of I, respectively. This will yield a full binary tree of depth $\log(n)$, where the i^{th} leaf (read from left to right) is the singleton $\{i\}$.

Our algorithm assigns to every node I a DM sketch for I as follows. To each leaf $\{i\}$ we assign the trivial sketch for i. For a non-leaf I with children I_1 and I_2, we take the DM-sketches (L_1, R_1, D_1) and (L_2, R_2, D_2) for I_1 and I_2 respectively, and apply MERGE followed by SHRINK with parameter γ to these sketches to get a DM-sketch (L', R', D') for I. We assign these DM-sketches inductively until we reach the root, yielding a DM sketch (L, R, D) for $[n]$. The output of the algorithm is $D^*(0, m)$.

We now convert this bottom up procedure into a streaming algorithm. We say that a node (interval) I is *completed* if we have reached the end of I in our stream, and we call a node (interval) *complemented* if its parent's other child is also completed. At any point during the stream, we maintain a DM sketch for every completed uncomplemented node I, creating a trivial DM sketch for each leaf as it is streamed. At step i, we look at the i^{th} value in the stream, and we find the largest interval in the binary tree for which i is the right endpoint of that interval. Call this interval I_k, where k is such that the size of this interval is 2^k. Define a sequence of intervals $I_k, I_{k-1}, ..., I_0$, where I_j is the right child of I_{j+1}. Note that i is the right endpoint of each I_j, so each I_j becomes completed at step i. As a result, our algorithm first creates the trivial sketch for i (Note that $I_0 = \{i\}$) and then performs a (possibly empty) sequence of merges and shrinks as follows. For $0 \le j < k$, given a DM sketch for I_j, the algorithm applies MERGE to the sketch for I_j and the sketch stored for its sibling, and then applies SHRINK with parameter γ to the output of MERGE to get a DM sketch for I_{j+1} (at which point it forgets the sketches for the children of I_{j+1}). The algorithm repeats this process k times, obtaining a sketch for I_k which it stores, as I_k is not yet complemented at step i. Once we reach the end of the stream, we will have our DM sketch for the root. We will prove:

THEOREM 3.1. (Main Theorem) *Let $\varepsilon > 0$ and consider the algorithm* DMAPPROX *with parameter $\gamma = \ln(1+\varepsilon)/(2\log(n))$. On input a sequence f of n integers,* DMAPPROX *outputs an approximation to the distance to*

monotonicity that is between DM_f and $(1+\varepsilon)DM_f$. The algorithm uses $O(\frac{1}{\varepsilon^2}\log^5(n)\log(m))$ space and runs in $O(\frac{1}{\varepsilon^3}n\log^6(n))$ time.

When accounting for time, we assume that arithmetic operations (additions and comparisons) can be done in unit time.

4 Proof of the Main Theorem

In this section we state some basic properties about the procedures MERGE and SHRINK, and use them to prove the main theorem. Some of these properties of MERGE and SHRINK are proved in this section, and others are proved in the next section.

LEMMA 4.1. (MERGE) *Suppose* MERGE *is run on input I,I_1,I_2, D_1,D_2 as described above and let (L,R,D) be the output DM-sketch.*

1. *The size of D is at most the sum of the sizes of D_1 and D_2.*
2. *If D_i is well-behaved for $i \in \{1,2\}$ then so is D.*
3. *If D_i is valid for I_i for $i \in \{1,2\}$ then D is valid for I.*
4. *If D_i is $(1+\delta)$-accurate for $i \in \{1,2\}$ then D is $(1+\delta)$-accurate.*
5. *The algorithm* MERGE *runs in space $O(\log(m)|L||R|)$ and time $O(|L||R|(|L|+|R|))$.*

The proof of this lemma is routine and unsurprising.

Proof. We prove each item of the claim sequentially.

First, we need to show that the size of D is at most the sum of the sizes of D_1 and D_2. The size of (L,R,D) is given by

$$\begin{aligned} max(|L|,|R|) &= max(|L_1 \cup L_2|, |R_1 \cup R_2|) \\ &\leq max(|L_1|+|L_2|, |R_1|+|R_2|) \\ &\leq max(|L_1|,|R_1|) + max(|L_2|,|R_2|). \end{aligned}$$

which is the sum of the sizes of (L_1,R_1,D_1) and (L_2,R_2,D_2).

Next, to show that D is well-behaved, we need to show that for $l,l' \in L$ and $r,r' \in R$ with $l \leq l'$ and $r' \leq r$, $D(l,r) \leq D(l',r')$. According to the definition of D, let z be such that $D(l',r') = D_1^*(l',z) + D_2^*(z,r')$. Since D_1 and D_2 are well-behaved, D_1^* and D_2^* are well-behaved by Proposition 2.1. This gives:

$$\begin{aligned} D_1^*(l',z) + D_2^*(z,r') &\geq D_1^*(l,z) + D_2^*(z,r) \\ &\geq D(l,r). \end{aligned}$$

where the last inequality follows from the definition of D. This shows that D is well-behaved.

Next, to show that (L,R,D) is valid, we need to show that for $x \in L$, $y \in R$,

(1) $D(x,y) \leq |I|$

(2) $D(x,y) \geq DM_I(x,y)$

Let z be such that $D(x,y) = D_1^*(x,z) + D_2^*(z,y)$. By Proposition 2.1,

$$|I| = |I_1| + |I_2| \geq D_1^*(x,z) + D_2^*(z,y) = D(x,y)$$

establishing (1).

For (2), let z be such that $D(x,y) = D_1^*(x,z) + D_2^*(z,y)$. We have $D_1^*(x,z) = D_1(\bar{x}^{L_1}, \underline{z}_{R_1})$ and $D_2^*(z,y) = D_2(\bar{z}^{L_2}, \underline{y}_{R_2})$ (Note that $\underline{z}_{R_1} \leq z \leq \bar{z}^{L_2}$). By the validity of (L_1,R_1,D_1) and (L_2,R_2,D_2),

$$\begin{aligned} D(x,y) &= D_1(\bar{x}^{L_1}, \underline{z}_{R_1}) + D_2(\bar{z}^{L_2}, \underline{y}_{R_2}) \\ &\geq DM_{I_1}(\bar{x}^{L_1}, \underline{z}_{R_1}) + DM_{I_2}(\bar{z}^{L_2}, \underline{y}_{R_2}) \\ &\geq DM_I(x,y) \end{aligned}$$

the last inequality following from the definition of DM. This shows that (L,R,D) is valid.

To prove the $(1+\delta)$-accuracy of D, let $l,r \in I$ and let J be an (l,r)-monotone subset of I of maximum size. We need to show that $D^*(l,r) \leq (1+\delta)DM_I(l,r)$. Let h be the value associated to the largest index of $J \cap I_1$. We see that $DM_{I_1}(l,h) + DM_{I_2}(h,r) = DM_I(l,r)$, so for the (L,R,D) sketch for I,

$$\begin{aligned} D^*(l,r) &= min_{l \leq k \leq r}(D_1^*(l,k) + D_2^*(k,r)) \\ &\leq D_1^*(l,h) + D_2^*(h,r) \\ &\leq (1+\delta)(DM_{I_1}(l,h) + DM_{I_2}(h,r)) \\ &\leq (1+\delta)DM_I(l,r). \end{aligned}$$

by the $(1+\delta)$-accuracy of (L_1,R_1,D_1) and (L_2,R_2,D_2). This shows that (L,R,D) is $(1+\delta)$-accurate.

We now analyze the amount of time that MERGE takes. Getting L and R from (L_1,R_1,D_1) and (L_2,R_2,D_2) is trivial, and getting $D(x,y)$ for each pair $(x,y) \in L \times R$ requires taking a minimum over at most $|L|+|R|$ choices of z (values of z outside of $L \cup R$ will not be helpful). Since the D_1^* and D_2^* values here can be computed in constant time (by looking at appropriate values in D_1 and D_2), each of these $|L|+|R|$ choices takes time $O(1)$. This yields the desired time bound of $O(|L||R|(|L|+|R|))$.

Finally, the amount of space that this algorithm uses is just the amount of space required to store L, R, and D. Since each element uses $\log(m)$ bits, this yields the desired space bound of $O(\log(m)|L||R|)$. This completes the proof of Lemma 4.1. □

LEMMA 4.2. (SHRINK) *On input an a sketch (L, R, D) that is valid for I and $(1+\delta)$-accurate,* SHRINK *with parameter γ outputs a sketch (L', R', D') that is well behaved and valid for I and is $(1+\gamma)^2(1+\delta)$-accurate. This algorithm runs in space $O(\log(m)|L||R|)$ and time $O(|L||R|)$.*

Proof. First, we see that SHRINK produces a matrix D' which is a submatrix of D for the same interval I, and as a result, the well behavedness and validity of (L', R', D') follows trivially from the definitions.

Next, we need to show that for $l, r \in [m] \cup \{0\}$, $D'^*(l,r) \leq (1+\gamma)^2(1+\delta)DM_I(l,r)$. We do this by showing that $D''^*(l,r) \leq (1+\gamma)D^*(l,r)$, and $D'^*(l,r) \leq (1+\gamma)D''^*(l,r)$. The two arguments are analogous, so we show the proof for the first case only. If $\bar{l}^{L'} = m$ (with $m \notin L'$), then since the largest value of L is in L', $\bar{l}^{L} = m$ also, and $D''^*(l,r) = D^*(l,r) \leq (1+\delta)DM_I(l,r)$ by hypothesis. Otherwise, $\bar{l}^{L'} = x_k$ or $\bar{l}^{L'} = \hat{x}_k$ for some k. If $\bar{l}^{L'} = x_k$, then for $x_k = l_{i+1}$ as in the description of SHRINK, $l > l_i$, so $\bar{l}^{L} = l_{i+1} = x_k$. This means that again, $D''^*(l,r) = D^*(l,r) \leq (1+\delta)DM_I(l,r)$ by hypothesis.

If instead, $\bar{l}^{L'} = \hat{x}_k$,

$$\begin{aligned} D''^*(l,r) &= D''(\hat{x}_k, \underline{r}_R) \\ &= D(\hat{x}_k, \underline{r}_R) \\ &\leq (1+\gamma)D(x_k, \underline{r}_R) \\ &\leq (1+\gamma)D(\bar{l}^{L}, \underline{r}_R) \\ &= (1+\gamma)D^*(l,r) \\ &\leq (1+\gamma)(1+\delta)DM_I(l,r) \end{aligned}$$

where the third line follows from the definition of SHRINK, and the fourth line follows from the well behavedness of D, as $x_k \leq l \leq \bar{l}^{L}$. This shows that (L', R, D'') is $(1+\gamma)(1+\delta)$-accurate. As mentioned earlier, an analogous argument with the shrinking of R shows that (L', R', D') is within a $(1+\gamma)$ factor of (L', R, D''), so (L', R', D') is $(1+\gamma)^2(1+\delta)$-accurate.

To analyze the amount of time this algorithm takes, we see that our algorithm involves constructing the sequence $x_1, \hat{x}_1, x_2, \hat{x}_2, x_3, ..., \hat{x}_{s-1}, x_s$. Recall that the elements of L are enumerated as $l_1 < l_2 < \cdots < l_{|L|}$. To determine, $x_{k+1} = l_{i'}$ from $x_k = l_i$, we need to compute the difference between rows l_j and l_i of D starting with $j = i+1$ and continuing until we reach $j = i'$ for which some entry of the difference vector exceeds $(1+\gamma)$ times the corresponding entry of row l_i (or l_j reaches $l_{|L|}$). When this happens we set $x_{k+1} = l_{i'}$ and $\hat{x}^k = l_{i'-1}$. If $x_{k+1} = l_{|L|}$ we stop otherwise we continue to determine x_{k+2} in the same way. Notice that throughout the algorithm, we consider each row only once as l_j so we compute the difference of at most $|L|$ pairs of rows. Each such difference is computed in $O(|R|)$ arithmetic operations so the overall running time is $O(|L||R|)$.

Looking at the amount of space used, we see that since (L', R', D') is not larger than (L, R, D) and none of the intermediate computations require any significant amount of space, we will need at most the space required to store the (L, R, D) sketch, which will be at most $O(\log(m)|L||R|)$ for the D matrix, as it consists of $|L||R|$ elements, each using at most $\log(m)$ bits. Note that if (L, R, D) has size $O(\log_{1+\gamma}(n))$, then this becomes space $O(\frac{1}{\gamma^2}\log^2(n)\log(m))$. □

We will also crucially need to control the size of the sketch that is output by SHRINK. Without an additional hypothesis on the input sketch (L, R, D) we can't bound the size of the sketch (L', R', D') (better than the trivial bound given by the size of (L, R, D)). To obtain the desired bound we will impose a technical condition called *coherence* on (L, R, D). We defer the definition of coherence until Section 5, but the reader can understand the structure of the argument in this section without knowing this definition. In section 5, we'll prove two Lemmas:

LEMMA 4.3. *If (L, R, D) is coherent, then the output (L', R', D') of* SHRINK *with parameter γ is coherent and satisfies* $\max(|L'|, |R'|) \leq 2\log_{1+\gamma} n + 3$.

In order to carry out the appropriate induction argument we'll need:

LEMMA 4.4. *For $i \in [n]$, the trivial sketch for i is coherent. Furthermore in* MERGE *if (L_1, R_1, D_1) and (L_2, R_2, D_2) are both coherent then so is (L, R, D).*

Using these pieces, we can now prove Theorem 3.1.

Proof. First, we aim to show that DMAPPROX approximates DM_f to within a $1+\varepsilon$ factor. To do this, it suffices to show that the DM sketch for $[n]$ computed by DMAPPROX is valid and $1+\varepsilon$-accurate. Let $\gamma = \ln(1+\varepsilon)/(2\log(n))$. If we run DMAPPROX on f, we have a binary tree of depth $\log(n)$, with a DM sketch for each node. Using Lemmas 4.1 and 4.2, for a node I with children I_1 and I_2, if the DM sketches for I_1 and I_2 are $(1+\delta)$-accurate, then the DM sketch for I is $(1+\gamma)^2(1+\delta)$-accurate. Furthermore, it is trivial to see that the trivial sketch for i is 1-accurate. By a simple induction on the depth of the tree, our final DM sketch (L, R, D) for $[n]$ is $(1+\gamma)^{2\log(n)}$-accurate. In addition, since the trivial sketch is valid and MERGE and SHRINK preserve validity, the DM sketch for $[n]$ is valid. We see that $(1+\gamma)^{2\log(n)} \leq (e^{2\gamma})^{\log(n)} = 1+\varepsilon$.

Next, we need to show that, at any point during the stream, the algorithm DMAPPROX uses $O(\frac{1}{\varepsilon^2}\log^5(n)\log(m))$ space. First, we note that the trivial sketch is coherent by Lemma 4.4, and since MERGE and SHRINK preserve coherence by Lemmas 4.4 and 4.3, every sketch computed by DMAPPROX is coherent by induction. Now, it is clear that the trivial sketch has size 1, and by Lemma 4.3, for any interval I, the DM sketch for I has size $O(\log_{1+\gamma}(n))$. As a result, the intermediate sketches resulting from applications of MERGE will also have size $O(\log_{1+\gamma}(n))$. Note also that for each of these sketches, the constant out in front is uniformly bounded by a small, fixed constant. As a result, it remains to show that the number of sketches stored by DMAPPROX at any given time is sufficiently small.

According to our algorithm description, we maintain DM sketches only for nodes which are both completed and uncomplemented. Since, for any given level of the tree, the sketches for the nodes of this level are obtained sequentially from left to right, at most one node from any level can be both completed and uncomplemented at any point during the stream. As a result, our algorithm stores $O(\log(n))$ sketches at any point in time. This means that the total amount of space needed to store these sketches is $O(\frac{1}{\gamma^2}\log^3(n)\log(m)) = O(\frac{1}{\varepsilon^2}\log^5(n)\log(m))$. Since none of the intermediate computations require more space than this, the desired result is achieved.

Lastly, we need to show that DMAPPROX runs in time $O(\frac{1}{\varepsilon^3}n\log^6(n))$. We start with n intervals of size 1 and we finish with 1 interval of size n, so our procedure performs $n-1$ applications of MERGE and SHRINK, each of which take $O(\frac{1}{\gamma^3}\log^3(n))$ time. Since our entire procedure consists of constructing our DM-sketches for the leaves (each of which takes $O(1)$ time), performing these applications of MERGE and SHRINK, and outputting a value from our final D matrix, the entire procedure runs in time $O(\frac{1}{\gamma^3}n\log^3(n)) = O(\frac{1}{\varepsilon^3}n\log^6(n))$. □

5 Sequence Matrices

In this section, we give the definition of the term *coherence* that appears in Lemmas 4.3 and 4.4, and we prove the lemmas.

At a high level, the goal of this section is to show that our SHRINK procedure yields a sketch which is sufficiently small. In order to do so, it will be necessary to keep track of not only the lengths of the increasing sequences represented by our D matrices, but also the sequences themselves. We have the following definitions:

- A *sequence matrix* S of an interval I is a matrix with rows and columns indexed by values of f, whose entries are f-monotone subsets of I.
- A sequence matrix is said to *represent* a DM sketch (L,R,D) if the rows of S are indexed by L, the columns of S are indexed by R, and for each $l \in L$ and $r \in R$ the entry $S(l,r)$ is an (l,r)-monotone subset of size $|I| - D(l,r)$.

Looking at SHRINK, we see that an element is added to L' each time condition (3.1) in the shrinking procedure is violated. We would like to show that each violation of this condition can be associated to a set of witnesses to the violation (which we call *irrelevant elements* below) of sufficient size. To illustrate the idea, consider $l_1, l_2 \in L$, $r \in R$ such that $D(l_2,r) > (1+\gamma)D(l_1,r)$ (a violation of condition (3.1)). If we set $k = D(l_2,r) - D(l_1,r)$, then if S represents (L,R,D), $S(l_1,r)$ has k more elements than $S(l_2,r)$, so it is clear that $S(l_1,r)$ contains at least k elements which do not appear in $S(l_2,r)$. We need for our argument that none of these elements appear in any entry of S in any row at or above l_2, but unfortunately this is not true for an arbitrary sequence matrix representative of (L,R,D). However, it will be possible for us to guarantee that this condition (which we call *coherence*) is satisfied by the sequence matrices that we consider. This motivates the following definitions.

- Given a sequence matrix S, an index i is said to be *left irrelevant* (henceforth we will refer to this simply as *irrelevant*) to $l \in [m] \cup \{0\}$ if for all $l' \in L, r \in R$ such that $l' \geq l$, $S(l',r)$ does not contain i. Analogously, an index i is said to be *right irrelevant* to $r \in [m] \cup \{0\}$ if for all $r' \in R, l \in L$ such that $r' \leq r$, $S(l,r')$ does not contain i.
- A DM sketch (L,R,D) is said to be *left-coherent* for I if there exists a representative sequence matrix S for this sketch such that for any two values $l_1, l_2 \in L$, $r \in R$, $S(l_1,r)$ contains at least $D(l_2,r) - D(l_1,r)$ indices which are left irrelevant (irrelevant) to l_2. Analogously, a DM sketch is said to be *right-coherent* for I if there exists a representative sequence matrix S for this sketch such that for any two values $r_1, r_2 \in R$, $l \in L$, $S(l,r_2)$ contains at least $D(l,r_1) - D(l,r_2)$ indices which are right irrelevant to r_1. Call (L,R,D) *coherent* if it is both left-coherent and right-coherent.
- For S a sequence matrix which represents a DM sketch (L,R,D), the *sequence estimator induced by* S is the $(m+1)\times(m+1)$ sequence matrix S^* given by:

$$S^*(l,r) = S(\bar{l}^L, \underline{r}_R)$$

For the purposes of our analysis, we will build up these sequence matrices in the same way we build up our distance matrices. For $i \in [n]$, the *trivial sequence matrix* for i is the 1×1 matrix $[\{f(i)\}]$. Note that the trivial sequence matrix for i represents the trivial sketch for i.

Let (L_1, R_1, D_1) and (L_2, R_2, D_2) be valid DM sketches for consecutive intervals I_1 and I_2 respectively, and let (L, R, D) be the output sketch obtained by applying MERGE to these two sketches. Given sequence matrices S_1 and S_2 which represent (L_1, R_1, D_1) and (L_2, R_2, D_2) respectively, we construct a sequence matrix S which represents (L, R, D) as follows. Recall that the matrix D constructed in our algorithm had entries $D(l, r)$, where $D(l, r) = min_{l \le z \le r}(D_1^*(l, z) + D_2^*(z, r))$. Let z_0 be the smallest z value achieving this minimum. Now let $S(l, r) = S_1^*(l, z_0) \cup S_2^*(z_0, r)$. It is clear that this union is an (l, r)-monotone subset of I. Furthermore, its size by the representativity of S_1 and S_2 is

$$\begin{aligned}&(|I_1| - D_1^*(l, z_0)) + (|I_2| - D_2^*(z_0, r))\\ &= |I| - (D_1^*(l, z_0) + D_2^*(z_0, r))\\ &= |I| - D(l, r)\end{aligned}$$

This shows that S is representative of (L, R, D). Call S the *merged sequence matrix* of S_1 and S_2.

We now state and prove a proposition which will help us prove Lemma 4.4.

PROPOSITION 5.1. *Let I be an interval split into two halves I_1 and I_2, and let S_1, S_2 be sequence matrices which represent I_1, I_2 respectively. Let S be the merged sequence matrix of S_1 and S_2. For $l \in L$ and any index $i \in I_1$, if i is irrelevant to l in S_1, then i is irrelevant to l in S. Similarly, for $r \in R$ and any index $i \in I_2$, if i is right irrelevant to r in S_2, then i is right irrelevant to r in S.*

Proof. We prove the first statement, the proof of the second part of the proposition is analogous. Let $l' \in L$ such that $l' \ge l$, and let $r \in R$. We aim to show that $S(l', r)$ does not contain i. We have that

$$\begin{aligned}S(l', r) &= S_1^*(l', z_0) \cup S_2^*(z_0, r)\\ &= S_1(\bar{l'}^{L_1}, \underline{z_0}_{R_1}) \cup S_2(\bar{z_0}^{L_2}, \underline{r}_{R_2})\end{aligned}$$

Since i is irrelevant to l in S_1, $S_1(\bar{l'}^{L_1}, \underline{z_0}_{R_1})$ does not contain i. Furthermore, $S_2(\bar{z_0}^{L_2}, \underline{r}_{R_2})$ does not contain i, as i lies in I_1, and $S_2(\bar{z_0}^{L_2}, \underline{r}_{R_2}) \subseteq I_2$. This shows that $S(l', r)$ does not contain i, proving the claim. □

Using this tool, we now prove Lemma 4.4.

Proof. First, it is clear that the trivial sequence matrix for i exhibits the coherence of the trivial sketch for i, as L and R both contain 1 element, making the condition for coherence trivially satisfied.

It remains to show that the resultant sketch (L, R, D) from the algorithm MERGE is coherent, given that the input sketches are coherent. We prove that (L, R, D) is left-coherent, the proof that it is right-coherent is analogous and left to the reader. Let I_1, I_2, I be as defined in Lemma 4.1, and let (L_1, R_1, D_1) and (L_2, R_2, D_2) be coherent DM sketches for I_1 and I_2 respectively. Let S_1 and S_2 be the representative sequence matrices for these sketches given by the left-coherent condition, and let S be the merged sequence matrix of S_1 and S_2. Let $l_1 < l_2$ be values in L, and let $r \in R$ (Note that the statement is trivial if $l_1 = l_2$, so we only consider $l_1 \ne l_2$). Our goal will be to find $D(l_2, r) - D(l_1, r)$ elements in $S(l_1, r)$ which are irrelevant to l_2. Let z_0 be the minimum value such that

$$D(l_1, r) = D_1^*(l_1, z_0) + D_2^*(z_0, r)$$

We break the argument into two cases:

Case 1: $z_0 \ge l_2$

In this case, we have that

$$\begin{aligned}D(l_2, r) &= \min_{l_2 \le z \le r}(D_1^*(l_2, z) + D_2^*(z, r))\\ &\le D_1^*(l_2, z_0) + D_2^*(z_0, r)\end{aligned}$$

so defining $k = D_1^*(l_2, z_0) - D_1^*(l_1, z_0)$, we have

$$D(l_2, r) - D(l_1, r) \le D_1^*(l_2, z_0) - D_1^*(l_1, z_0) = k$$

Since (L_1, R_1, D_1) is left-coherent, $S_1^*(l_1, z_0)$ contains at least k indices which are irrelevant to l_2. These indices are in $S(l_1, r)$ by definition of the merged sequence matrix, and they are irrelevant to l_2 in S by proposition 5.1. As such, we find k indices in $S(l_1, r)$ which are irrelevant to l_2 proving the claim in this case.

Case 2: $z_0 < l_2$

In this case, we have that

$$\begin{aligned}D(l_2, r) &= \min_{l_2 \le z \le r}(D_1^*(l_2, z) + D_2^*(z, r))\\ &\le D_1^*(l_2, l_2) + D_2^*(l_2, r)\end{aligned}$$

$$\begin{aligned}D(l_2, r) - D(l_1, r) \le\ & D_1^*(l_2, l_2) - D_1^*(l_1, z_0)\\ &+ D_2^*(l_2, r) - D_2^*(z_0, r)\end{aligned}$$

Let k_1, k_2 be such that

$$\begin{aligned}D_1^*(l_2, l_2) - D_1^*(l_1, z_0) &= k_1\\ D_2^*(l_2, r) - D_2^*(z_0, r) &= k_2\end{aligned}$$

By definition of D^*, we have that $D_1^*(l_2, l_2) = |I_1| = D_1^*(l_2, z_0)$, so $k_1 = D_1^*(l_2, z_0) - D_1^*(l_1, z_0)$. Again, since (L_1, R_1, D_1) is left-coherent, $S_1^*(l_1, z_0)$ contains at least k_1 indices which are irrelevant to l_2. These indices are in $S(l_1, r)$ by definition of the merged sequence matrix, and they are irrelevant to l_2 in S by proposition 5.1.

Furthermore, since (L_2, R_2, D_2) is left-coherent, $S_2^*(z_0, r)$ contains at least k_2 indices which are irrelevant to l_2. These indices are in $S(l_1, r)$ by definition of the merged sequence matrix, and they are irrelevant to l_2 in S by proposition 5.1. Lastly, note that $S_1^*(l_1, z_0) \subseteq I_1$ and $S_2^*(z_0, r) \subseteq I_2$, so these two sets of indices are disjoint. As such, we find $k_1 + k_2$ indices in $S(l_1, r)$ which are irrelevant to l_2 proving the claim in this case as well.

This exhausts all cases, proving the lemma. □

We now prove Lemma 4.3.

Proof. First, the reader should note that if (L, R, D) is a coherent sketch (with sequence matrix S) and (L', R', D') is any shrinking of (L, R, D) (i.e. $L' \subseteq L$, $R' \subseteq R$, and D' is the associated submatrix of D induced by L' and R'), then it is clear that (L', R', D') is also coherent, as we can just take S' to be the appropriate submatrix of S. As a result, we have that SHRINK preserves coherence.

Consider the sequence $x_1, x_2, x_3, ..., x_s = l_{|L|}$ (without the $\hat{x}$'s) described in the SHRINK procedure. For each i, let r_i be an element $r \in R$ that maximizes $D(x_{i+1}, r) - D(x_i, r)$ and let $k_i = D(x_{i+1}, r_i) - D(x_i, r_i)$. Let S be a coherent sequence matrix representative of (L, R, D). By the definition of left-coherent, for each i between 1 and $s-1$ there are k_i elements of $S(x_i, r_i)$ that are irrelevant to x_{i+1} (and thus also irrelevant to $x_{i+2}, \ldots, x_s$). Thus these sets of irrelevant elements are disjoint and so $k_1 + \ldots + k_{s-1} \leq n$.

We now prove by induction on j between 1 and $s-1$ that $k_1 + \ldots + k_j \geq (1+\gamma)^{j-1}$. For the basis, $k_1 \geq 1$, and for the induction step suppose $j > 1$. There are $k_1 + \cdots + k_{j-1}$ indices that are irrelevant to x_j so all entries of row x_j are at least this sum which is at least $(1+\gamma)^{j-2}$ by induction. Since k_j is at least γ times the smallest entry of row x_j by condition (3.1) in SHRINK, we have $k_1 + \cdots + k_j \geq (1+\gamma)(k_1 + \cdots + k_{j-1}) \geq (1+\gamma)^{j-1}$.

On the other hand, $k_1 + \cdots + k_{s-1} \leq n$ which implies $s \leq \log_{1+\gamma}(n) + 2$ so $|L'| \leq 2\log_{1+\gamma}(n) + 3$. Similarly $|R'|$ is bounded above by the same quantity. □

6 Algorithm for unknown input length

In streaming algorithms, the question of what values are known to the algorithm is frequently asked. The reader should note that, in our previous algorithm, m was not needed, however the algorithm did require a priori knowledge of the value of n. A closer look at the algorithm shows that, apart from the value of γ, knowledge of n was not needed (note that the way we progress through the binary tree allows us to build to it as we go, continuing the procedure in the same way regardless of the size of n).

Seeking this, we look at the role of γ in our approximation, and we see that one property it had was that $\prod_{i=1}^{\log(n)}(1+\gamma)^2 \leq 1+\varepsilon$. We replace γ with a quantity that depends on the current level of the binary tree, call it $a(i)$ (Here level is counted from the bottom up, i.e. the leaves are at level 1, the parents of the leaves are at level 2, etc). If n is not known beforehand then in principle it could be arbitrarily large, meaning that if we replace γ with $a(i)$, we require $\prod_{i=1}^{\infty}(1+a(i))^2 \leq 1+\varepsilon$. Taking $a(i) = \frac{c}{i^{1+\beta}}$ for any fixed $\beta > 0$ we can choose $c = c(\beta)$ so that this product is at most $1+\varepsilon$. This will yield the desired accuracy of approximation, so it remains to determine the amount of space that this modified algorithm would require.

This modification will result in DM sketches of size $O(\log_{1+a(i)}(n))$ after i merges. We see that $a(i) \leq \frac{c}{\log^{1+\beta}(n)}$, since we have $\log(n)$ levels in our tree, so this yields DM sketches of size at most $O(\frac{1}{\varepsilon}\log^{2+\beta}(n))$. As a result, our D matrices have at most $O(\frac{1}{\varepsilon^2}\log^{4+2\beta}(n))$ entries, resulting in an algorithm that runs in space $O(\frac{1}{\varepsilon^2}\log^{5+2\beta}(n)\log(m))$, for any $\beta > 0$.

7 Lower bounds for approximating distance to monotonicity

In this section we use standard communication complexity arguments to prove lower bounds for the space complexity of approximating distance to monotonicity for both randomized and deterministic algorithms. We apply a reduction from an appropriate one-way communication problem, a common technique which has been used frequently to establish streaming lower bounds [11].

Let $A(n, \varepsilon)$ be the problem of approximating the distance to monotonicity of n integers taking on values in $[m]$ (where $m = poly(n)$) to within a factor of $(1+\varepsilon)$. Now consider the one-way communication problem where Alice is given a list of k r-bit integers $x_1, x_2, ..., x_k$, Bob is given an index i between 1 and k, as well as an r-bit integer y, and the goal is to compute $GT(x_i, y)$, where GT is the "greater than" function ($GT(x, y) = 1$ iff $x > y$). Denoting this problem by $B(k, r)$, we show that for appropriate choices of parameters, $B(k, r)$ can be reduced to $A(n, \varepsilon)$.

THEOREM 7.1. *Let* $k = \lfloor \frac{1}{2}\log_{1+\varepsilon}(\frac{\varepsilon n}{2\lceil 1/\varepsilon \rceil}) - \frac{1}{2} \rfloor$, $r = \lceil \log(n) \rceil$, *and assume there exists a protocol to solve*

$A(n,\varepsilon)$ using $S(n,m,\varepsilon)$ bits of space. Then there is a protocol for $B(k,r)$ using $O(S(n,m,\varepsilon))$ bits.

In order to prove this theorem, we will need the following proposition.

PROPOSITION 7.1. *Let $\varepsilon > 0$, $n \in \mathbb{N}$, $k = \lfloor \frac{1}{2}\log_{1+\varepsilon}(\frac{\varepsilon n}{2\lceil 1/\varepsilon \rceil}) - \frac{1}{2} \rfloor$. There exists a sequence of positive integers $a_1, a_2, ..., a_k$ satisfying the following properties:*

1. $\forall j < k, a_j \geq \varepsilon \sum_{i=j+1}^{k} a_i$
2. $\sum_{i=1}^{k} a_i \leq \frac{n}{2}$

Proof. We construct such a sequence $a_1, a_2, ..., a_k$ as follows. Let $a_k = \lceil \frac{1}{\varepsilon} \rceil$. For $j < k$, set $a_j = \lceil (1+\varepsilon)a_{j+1} \rceil$. To establish property 1, we see inductively

$$\begin{aligned} a_j &\geq (1+\varepsilon)a_{j+1} \\ &= \varepsilon a_{j+1} + a_{j+1} \\ &\geq \varepsilon a_{j+1} + \varepsilon \sum_{i=j+2}^{k} a_i \\ &= \varepsilon \sum_{i=j+1}^{k} a_i \end{aligned}$$

For property 2, we first note trivially that for any real number $x \geq \frac{1}{\varepsilon}$, we have $(1+\varepsilon)x \geq x + 1 \geq \lceil x \rceil$. As a result, for any $j < k$, $a_j = \lceil (1+\varepsilon)a_{j+1} \rceil \leq (1+\varepsilon)^2 a_{j+1}$. This yields the following:

$$\begin{aligned} \sum_{i=1}^{k} a_i &\leq a_1 + \sum_{i=2}^{k} a_i \\ &\leq a_1 + \frac{1}{\varepsilon} a_1 \\ &= \frac{1+\varepsilon}{\varepsilon} a_1 \\ &\leq \frac{1+\varepsilon}{\varepsilon} (1+\varepsilon)^{2k} \lceil 1/\varepsilon \rceil \\ &\leq \frac{n}{2} \end{aligned}$$

□

Using this, we prove Theorem 7.1.

Proof. Assume that we have a streaming protocol P for $A(n,\varepsilon)$ using $S(n,m,\varepsilon)$ bits, and consider an instance of $B(k,r)$ where Alice receives $x_1, x_2, ..., x_k$ as input, and Bob receives i, y as input. Consider the sequence of integers $T(x_1, x_2, ..., x_k, i, y)$ defined as follows. Let $a_1, a_2, ..., a_k$ be a sequence of integers satisfying Proposition 7.1, and for any j let $g(x_j, l) = n^2(l-1) + nx_j$. $T(x_1, x_2, ..., x_k, i, y)$ will consist of $k+1$ blocks, where for $j \leq k$, the j^{th} block consists of a_j consecutive integers ending at $g(x_j, j)$, and the $(k+1)^{th}$ block will consist of $n - \sum_{j=1}^{k} a_j$ consecutive integers beginning at $g(y, i) + 1$.

Under this construction, if $x_i \leq y$, then the first i blocks along with the last block form an increasing subsequence of length greater than $n/2$, and any increasing subsequence containing any element from blocks $i+1$ through k cannot contain any element from the last block, so it will have length at most $n/2$. As a result, the increasing subsequence of the first i blocks and the last block are a longest increasing subsequence, so the distance to monotonicity of $T(x_1, x_2, ..., x_k, i, y)$ is $\sum_{j=i+1}^{k} a_j$. On the other hand, if $x_i > y$, then the same is true for the first $i-1$ blocks along with the last block, so the distance to monotonicity of $T(x_1, x_2, ..., x_k, i, y)$ is $\sum_{j=i}^{k} a_j$ in this case. By condition 1 of Proposition 7.1, these values differ by a factor of at least $(1+\varepsilon)$, so P must be able to separate these two cases. As a result, Alice can construct the first k blocks of $T(x_1, x_2, ..., x_k, i, y)$ using her input and run P on this part of the sequence. She can then communicate the current bits stored by P to Bob, at which point Bob can construct the last block of $T(x_1, x_2, ..., x_k, i, y)$ using his input and run the remainder of P to get its result. At this point, Bob can use the result of P to determine whether or not $x_i > y$, and output the result. This is a protocol for $B(k,r)$ using $O(S(n,m,\varepsilon))$ bits, so $B(k,r)$ requires $O(S(n,m,\varepsilon))$ bits. □

As a result of this reduction, any deterministic (resp. randomized) lower bound for $B(k,r)$ will translate to a deterministic (resp. randomized) lower bound for $A(n,\varepsilon)$.

LEMMA 7.1. *Given $B(k,r)$ as defined above,*

1. *Any deterministic protocol for $B(k,r)$ requires $\Omega(kr)$ bits.*
2. *Any randomized protocol for $B(k,r)$ requires $\Omega(\frac{kr}{\log(r)})$ bits.*

Before proving this lemma, we note that it along with Theorem 7.1 immediately implies the following two results.

THEOREM 7.2. *Any deterministic streaming algorithm which approximates the distance to monotonicity of a sequence of n nonnegative integers to within a factor of $1+\varepsilon$ requires space $\Omega(\frac{1}{\varepsilon}\log^2(n))$.*

THEOREM 7.3. *Any randomized streaming algorithm which approximates the distance to monotonicity of a sequence of n nonnegative integers to within a factor of* $1+\varepsilon$ *requires space* $\Omega(\frac{1}{\varepsilon}\frac{\log^2(n)}{\log\log(n)})$.

We now prove Lemma 7.1

Proof. Starting with the first claim, it is a well known fact that the deterministic one-way communication complexity of a function $D(x,y)$ is just $\log(w)$, where w is the number of distinct rows in the communication matrix for D. Since any two rows of the matrix for $B(k,r)$ corresponding to distinct k-tuples $(x_1, x_2, ..., x_k)$ are distinct, it remains to count the number of such possible k-tuples. Each x_i can take on any of 2^r values, giving us 2^{kr} such k-tuples. The claim follows.

Before addressing the second claim, we first note that $B(1,r)$ is just the "Greater Than" function, $GT(r)$. It has been shown that a lower bound for the one-way communication complexity of $GT(r)$ is $\Omega(r)$ [6]. It seems plausible that this would translate to a $\Omega(kr)$ lower bound for the one-way communication complexity of $B(k,r)$, however we are unable to adapt this argument. [5] gives a simpler argument achieving a lower bound of $\Omega(\frac{r}{\log(r)})$ for the one-way communication complexity of $GT(r)$, which we are able to adapt to achieve a lower bound of $\Omega(\frac{kr}{\log(r)})$ for $B(k,r)$. Applying this technique, we show that running a randomized protocol for $B(k,r)$ $O(\log(r))$ times will yield a randomized one way protocol capable of computing the indexing function where Alice is given a kr bit string $x_1, x_2, ..., x_{kr}$, Bob is given an index $i \in [kr]$, and the goal is to output x_i, a problem that is known to require $\Omega(kr)$ bits [5].

Let P be a randomized protocol for $B(k,r)$ achieving the optimal complexity. Fix inputs $x_1, x_2, ..., x_k, i, y$ for Alice and Bob. If Alice and Bob run P on this input, they will err with probability at most $1/3$. If instead Alice and Bob run P $c\log(r)$ times for some constant c and Bob outputs the majority result, this protocol will err with probability at most $r^{-\Omega(1)}$. Note that the message sent by this protocol does not depend on Bob's input, meaning Bob can compute the output for several different choices of his input without any additional communication (though it will increase the probability of error). This means that for the set $\{y_1, y_2, ..., y_r\}$, Bob can use Alice's message to compute $GT(x_i, y_j)$ for each j. Furthermore, since this set has only r elements, the probability that all of these computations are correct is at least $1 - r^{-\Omega(1)}$. Choosing the y_j's accordingly, Bob can essentially run a binary search to determine x_i exactly with high probability. To see this, we first note that, given a fixed x_i, running a binary search to determine x_i will use a fixed sequence $y_1, y_2, ..., y_r$, assuming each output of $GT(x_i, y_j)$ is correct. Therefore, for any j, if $GT(x_i, y_t)$ gave the correct output for each $t < j$, then Bob's choice of y_j will be determined by x_i (i.e. by the previous values of $GT(x_i, y_t)$). Since the value of x_i uniquely determines the sequence $y_1, y_2, ..., y_r$ that will yield a correct binary search for x_i, we can use a union bound to bound the probability that $GT(x_i, y_j)$ outputs the correct value for all indices j. Since for any fixed j, the probability that the output for $GT(x_i, y_j)$ is incorrect is at most $r^{-\Omega(1)}$, the probability that at least one of these is incorrect is at most $r^{1-\Omega(1)} = r^{-\Omega(1)}$. This shows that the probability that all of these outputs are correct (i.e. the probability that Bob correctly computes x_i) is at least $1 - r^{-\Omega(1)}$.

As a result, this is a randomized protocol P' for the problem where Alice is given $x_1, x_2, ..., x_k$, Bob is given i, and the goal is to compute every bit of x_i. This protocol can be used as a protocol for the indexing problem mentioned earlier as follows. For an instance of this aforementioned problem, Alice is given $x'_1, x'_2, ..., x'_{kr}$, Bob is given an index $i \in [kr]$, and the goal is to output x'_i. Alice can view her input as k strings of length r and run P'. Bob can run P' using $\lfloor \frac{i-1}{n} + 1 \rfloor$, which will give him the value of x'_i (in addition to several other values x'_j) with probability at least $2/3$. This shows that $c\log(r)$ iterations of P can be used to simulate a computation known to require $\Omega(kr)$ bits [5]. The result follows. □

References

[1] D. Aldous and P. Diaconis, *Longest increasing subsequences: from patience sorting to the Baik-Deift-Johannson theorem*, Bulletin of the American Mathematical Society, 36 (1999), pp. 413–432.

[2] M. Fredman, *On computing the length of the longest increasing subsequences*, Discrete Mathematics, 11 (1975), pp. 29–35.

[3] C. Schensted, *Longest increasing and decreasing subsequences*, Canadian Journal of Mathematics, 13 (1961), pp. 179–191.

[4] P. Ramanan, *Tight* $\Omega(n \lg n)$ *lower bound for finding a longest increasing subsequence*, International Journal of Computer Mathematics, 65(3 & 4) (1997), pp. 161–164.

[5] I. Kremer and N. Nisan and D. Ron, *On Randomized One-Round Communication Complexity*, Computational Complexity, 8 (1995), pp. 596–605.

[6] P. B. Milterson and N. Nisan and S. Safra and A. Wigderson, *On data structures and asymmetric communication complexity*, J. Comp. System Sci., 57(1) (1998), pp. 37–49.

[7] A. Gál and P. Gopalan, *Lower bounds on streaming algorithms for approximating the length of the longest increasing subsequence*, Proceedings of the 48th Symposium on Foundations of Computer Science (FOCS)), 2007, pp. 294–304.

[8] F. Ergun and H. Jowhari, *On distance to monotonicity and longest increasing subsequence of a data stream*, in Proceedings of the 19th Symposium on Discrete Algorithms (SODA), 2008, pp. 730–736.

[9] P. Gopalan and T. S. Jayram and R. Krauthgamer and R. Kumar, *Estimating the sortedness of a data stream*, in Proceedings of the 18th Symposium on Discrete Algorithms (SODA), 2007, pp. 318–327.

[10] M. Saks and C. Seshadhri, *Space efficient streaming algorithms for the distance to monotonicity and asymmetric edit distance*, in Proceedings of the 24th Symposium on Discrete Algorithms (SODA), 2013, pp. 1698–1709.

[11] X. Sun and D. P. Woodruff, *The communication and streaming complexity of computing the longest common and increasing subsequences*, in Proceedings of the 18th Symposium on Discrete Algorithms (SODA), 2007, pp. 336–345.