

# Tight Lower Bounds for Multi-Pass Stream Computation via Pass Elimination

Sudipto Guha\*

Andrew McGregor<sup>†</sup>

May 1, 2008

## Abstract

There is a natural relationship between lower bounds in the multi-pass stream model and lower bounds in multi-round communication. However, this connection is less understood than the connection between single-pass stream computation and one-way communication. In this paper, we consider data-stream problems for which reductions from natural multi-round communication problems do not yield tight bounds or do not apply. While lower bounds are known for some of these data-stream problems, many of these only apply to deterministic or comparison-based algorithms, whereas the lower bounds we present apply to any (possibly randomized) algorithms. Our results are particularly relevant to evaluating functions that are dependent on the ordering of the stream, such as the longest increasing subsequence and a variant of tree pointer jumping in which pointers are revealed according to a post-order traversal.

Our approach is based on establishing “pass-elimination” type results that are analogous to the round-elimination results of Miltersen et al. [23] and Sen [29]. We demonstrate our approach by proving tight bounds for a range of data-stream problems including finding the longest increasing sequences (a problem that has recently become very popular [12, 15, 16, 22, 30] and we resolve an open question of [30]), constructing convex hulls and fixed-dimensional linear programming (generalizing results of [8] to randomized algorithms), and the “greater-than” problem (improving results of [9]). These results will also clarify one of the main messages of our work: sometimes it is necessary to prove lower bounds directly for stream computation rather than proving a lower bound for a communication problem and then constructing a reduction to a data-stream problem.

## 1 Introduction

A natural connection between communication complexity and small-space computation in the data-stream model has been widely exploited since the early papers on the data-stream model [2, 19]. For example, a stream computation can be conceptualized as a communication problem between players who know different segments of the input stream. Consequently, a lower-bound on the total amount of communication necessary yields a lower-bound on the amount of memory required by the streaming algorithm. This approach has been very successful and many lower bounds have been proved in this manner (e.g., [2, 3, 6, 18].) However, interesting issues arise when trying to prove lower bounds for multi-pass algorithms<sup>1</sup> and these have not been explored as thoroughly as in the single-pass case. Many of the lower bounds that do exist only apply to deterministic algorithms, e.g., recent results on approximating the length of the longest increasing subsequence [12, 15] and solving geometric problems [8], or only apply to restricted classes of algorithm that may only use memory in a certain way [10, 16, 24]. Given that randomization is necessary for many classic data-stream problems (e.g.,  $\ell_p$  norms and frequency moments [2, 20]) we consider it natural to seek lower-bounds for fully general randomized algorithms. We are particularly interested in space/pass trade-offs.

The goal of this paper is to construct a framework for proving multi-pass lower bounds for certain types of data-stream problems. Our approach is based on establishing pass-elimination results which tailor ideas from the

---

\*Dept. of Computer and Information Sciences, University of Pennsylvania. Philadelphia, PA 19104. Email: sudipto@cis.upenn.edu

<sup>†</sup>Information Theory & Applications Center, University of California, San Diego. San Diego, CA 92093. Email: andrewm@ucsd.edu.

<sup>1</sup>We consider input to be on a read-only tape rather than a read-write tape as in, e.g., [4, 11].

round-elimination technique in communication complexity [23, 29] specifically to data-stream computation. This allows us to prove results on tree pointer jumping when the data is revealed by a *post-order traversal* rather than level-by-level. This will play a crucial role in proving lower bounds for order-dependent functions in the data-stream model and will exemplify a distinction we wish to make between multi-party communication and data-stream computation. With this technology we prove results for a range of problems including finding the longest increasing subsequence, determining the larger of two values, constructing convex hulls, and fixed dimensional linear programming. The longest increasing subsequence (LIS) problem is to find the increasing sub-stream of maximum length. This problem, and the related problem of estimating the length of the longest increasing subsequence has become very popular in recent years [12, 15, 16, 22, 30]. We prove a tight multi-pass lower bound for LIS, thereby resolving an open question of Sun and Woodruff [30]. We note that this problem exhibits an interesting doubly-exponential space/pass trade-off. The greater-than (GT) problem is to determine which of two integers is larger given the bits of their binary expansion. It is often used as an example of the utility of round-elimination in the communication setting. A lower bound for a data-stream version was given by Chang and Kannan [9] by appealing to the communication result. We show that this lower bound can be substantially improved using the pass-elimination framework and this will further highlight the round-elimination/pass-elimination and data-stream/communication distinctions that are important in this paper. Finally, we generalize bounds for deterministic algorithms for geometric problems including constructing convex hulls and fixed dimensional linear programming [8] to apply to randomized algorithms.

**Previous Approaches and Limitations:** There do exist multi-pass lower bounds for randomized algorithms for some problems in the data-stream model. However, almost all of these bounds are based on reductions from a relatively small set of communication problems. Many bounds are based on the multi-round communication of multi-player set-disjointness (e.g., [3, 13, 16, 22, 28]) and these bounds achieve lower-bounds on the product of the number of passes and the amount of space. Hence, lower bounds from set-disjointness only exhibit space bounds that scale with the reciprocal of the number of passes. An approach that yields more interesting space/pass trade-offs is to consider pointer jumping problems (e.g., [14, 18, 25]) and related problems such as the greater-than problem (e.g., [9, 23, 29]). There are two issues with reductions from pointer-jumping and greater-than: the difficulty in achieving tight results in the data-stream model (rather than a communication model) and applying these techniques to order-dependent functions in the data-stream model. To elaborate upon these issues, it is necessary to review the round-elimination technique that underlies many communication lower bounds [1, 5, 7, 26, 27]. In round-elimination we wish to evaluate a function  $f(x, y)$  where  $x$  is known to Alice and  $y$  to Bob. A “meta-problem”  $P_f(x_1, \dots, x_t, i, y) = f(x_i, y)$  is defined where  $x_1, \dots, x_t$  is known to Alice and  $i, y$  is known to Bob. It was shown that the existence of a “good”  $k$  round protocol for  $P_f$  with Alice starting, implied that there exists a “good”  $k - 1$  round protocol for  $f$  where Bob communicates first. The result holds true even if public coins are allowed and Bob is given  $x_1, \dots, x_{i-1}$ . Essentially, the good protocol for  $P_f$ , stripped of its “not-so-useful” first round, gives a good protocol for  $f$ . If  $P_f$  is a “self-reducible” problem, i.e.,  $P_f$  and  $f$  can be thought of as the same problem on different size inputs, this gives a beautiful inductive way of bounding the communication of  $k$ -round protocols for  $f$  given a lower bound for 1 round protocols.

For example, the approach implies that if Alice and Bob have  $n$ -bit binary strings  $x$  and  $y$  respectively and are permitted to communicate a total of  $r$  messages, then to determine if  $x < y$  requires  $\tilde{\Omega}(n^{1/r})$  bits of communication. However, if we consider the same problem in the stream setting we deduce that any  $p$ -pass algorithm requires  $\tilde{\Omega}(n^{1/(2p-1)})$  space because  $p$  passes over the data corresponds to  $2p - 1$  messages. We show that this is not tight and that a  $\tilde{\Omega}(n^{1/p})$  lower-bound exists. Note that if  $p = 1$ ,  $2p - 1 = p$  and hence tight one-pass lower bounds are sometimes achieved.

The second issue relates to proving lower bounds for evaluating functions, such as finding the longest increasing subsequence, which are dependent on the ordering of the stream. Here the round elimination lemma has a fundamental problem: the fact that  $x_i$  and  $y$  must interact (because  $f$  encodes order), implies that  $x_i, x_j$  interacts (through  $y$ ). But the round elimination framework requires independence of  $x_i, x_j$  ( $i \neq j$ ) and the framework does not apply as is; while we cannot describe all failed attempts, the inherent difficulty can be seen by trying to prove even a two pass result.

**Our Approach and Results:** We prove “Pass Elimination” lemmas that allow us to prove results related to round elimination *directly* for data streams rather than the usual two-step process of proving a communication lower bound and then using this to imply a data-stream lower bound. One intuitive (but technically inaccurate) way of viewing this

result is to consider round-elimination in which Bob has an index  $i$  and no other information. Hence, after the first pass Bob has nothing else to say if he reveals  $i$ . This will solve the problem of doubling the rounds and avoiding the issue of the interactions. However, in the communication complexity framework, evaluating  $f(x_i)$  is trivial once  $i$  is known since one player knows  $x_i$ ! However, in the data-stream setting, evaluating  $f(x_i)$  remains a meaningful problem. We prove the pass-elimination results using an information theoretic approach similar to that used by Sen [29]. However, we emphasize that the main contribution of our work is understanding and quantifying the exact problem which allows us to prove a variety of tight lower bounds. While in most cases the stronger model of communication complexity makes it simpler to prove lower bounds, our tighter lower bounds are achievable by exploiting the weakness of the data stream model. We believe that these results provide a natural, direct, and intuitive framework for proving multi-pass lower bounds, which is likely to encourage its use.

We demonstrate the utility of the pass-elimination lemmas by applying them to a variety of problems. We consider the resulting bounds interesting in their own right as they are either the first known lower bounds for the specific problem, or they significantly improve previous known bounds (either by establishing better bounds or by being more general, e.g., applying to all algorithms rather than just deterministic algorithms.)

1. *Post-Order Traversal Tree Pointer Jumping*: In Section 2, we consider the data-stream problem of tree pointer jumping in which the pointer values for each node are revealed according to a post-order traversal of the tree. We prove that any  $p$ -pass algorithm for the  $(p + 1)$ -level,  $t$ -ary problem requires  $\Omega(t/2^p)$  space. Note that this really is a data-stream bound in the sense that the only natural way to conceptualize the problem as a communication problem is if there is a player for each of the  $O(t^p)$  nodes in the tree. Consequently, the usual approach of bounding maximum message length (or space in the data-stream setting) by averaging the total communication over each message sent fails. A main achievement in [15] was circumventing such an argument but this was only achieved in the deterministic setting.
2. *Longest Increasing Subsequence*: In Section 3, we prove that any  $p$ -pass (randomized) algorithm that finds the elements of the longest increasing subsequence requires  $\tilde{\Omega}(k^{1+1/(2^p-1)})$  space where  $k$  is the length of the longest increasing subsequence. This matches the upper bound of [22] and resolves an open question of [30]. This doubly-exponential space/pass trade-off is unusual in the data-stream literature but we expect it may arise for other problems for which the natural way to attack the problem is by dynamic programming, e.g., for time-series histograms [17].
3. *Promise Minimum Missing Element and Greater-Than*: The greater-than (GT) problem is to determine which of two integers,  $x, y \in 2^n$ , is larger given a length  $2n$  stream whose elements are the bits of the binary expansion of  $x$  and  $y$  (in some order.) In Section 4, we show that any  $p$ -pass (randomized) algorithm for this problem requires  $\tilde{\Omega}(n^{1/p})$  space. Our result also applies to the related problem of finding the minimum missing element (MME) of a stream where all elements smaller than this element occur with multiplicity 1.<sup>2</sup> The best previous bound was  $\tilde{\Omega}(n^{1/(2^p-1)})$  [9].
4. *Convex Hulls and Fixed-Dimensional Linear Programming*: In Section 5, we show that any  $p$ -pass algorithm for fixed-dimensional linear programming requires  $\tilde{\Omega}(n^{1/p})$  space where the stream consists of  $n$  constraints and the objective function is known. In the full version, we also show bounds for  $p$ -pass algorithms for constructing the convex hull of a stream of  $n$  points in  $\mathbb{R}^2$ . If the points are sorted on their  $x$ -coordinates we show that  $\tilde{\Omega}(\sqrt{n})$  space is required for  $p = O(1)$ . These bounds generalize previous bounds for a restricted class of deterministic algorithms [8].

**Important Note on the Model:** It will be convenient to consider a more powerful variant of the usual space-bounded data-stream model. Specifically, we allow algorithms to do an unbounded amount of work, using an unbounded amount of space between the arrival of each data element. We only insist that the amount of space in use when the next element arrives satisfies the appropriate bound. Of course, lower bounds for this more powerful model also apply to the usual model. Also, throughout we will assume that the number of passes is constant although many of the results generalize.

<sup>2</sup>Without the promise, MME is related to 2-party set disjointness and stronger bounds exist.

## 2 Pass Elimination (When the First Pass is Passé...)

In this section we present two new general lower-bounds for multi-pass algorithms in the data-stream model. These results are established by taking ideas from the round-elimination results in communication complexity and applying them directly to data streams. We start by defining the following “meta problems.”

**Definition 1.** For a problem  $f$  defined on a set of streams  $\mathcal{X}$ , we define:

1.  $P_{t,f,g}$ : Given a stream  $\langle x_1, \dots, x_t, i, g(x_1, x_2, \dots, x_{i-1}) \rangle \in \mathcal{X}^t \times [t] \times \mathcal{X}$ , solve  $f(x_i)$  where  $g$  is an arbitrary function  $g : \mathcal{X}^{i-1} \rightarrow \mathcal{X}$ .
2.  $Q_{t,f}$ : Given a stream  $\langle x_1, \dots, x_t \rangle \in \mathcal{X}^t$ , output  $\{\langle i, f(x_i) \rangle : i \in [t]\}$  in any order. We do not require the output to be stored in working memory.<sup>3</sup>

We will prove the following lemma in Section 2.1.

**Lemma 2** (Pass Elimination for  $P_{t,f,g}$ ). Assume  $t \geq 5000s$ . If there exists a  $k$ -pass,  $s$ -space algorithm for  $P_{t,f,g}$  with prob. at least  $1 - \delta$ , then there exists a  $(k - 1)$ -pass,  $2s \log(\delta^{-1})$ -space algorithm for  $f$  with prob. at least  $1 - \delta$ .

It is reasonable to ask if it is possible to prove a version of the above lemma whether the space requirement remains the same while the error probability only increases additively. This was the case in the communication setting [29]. Unfortunately in the data-stream setting the state must be encoded in the current memory whereas, in the communication setting, the state is determined by all messages that have been sent (note that 2-player communication can be trivially assumed to be in the blackboard model.) The proof of Lemma 3 is similar to that of Lemma 2 and can be found in the full version.

**Lemma 3** (Pass Elimination for  $Q_{t,f}$ ). Assume  $t \geq 2s\delta^{-1}$ . If there exists a  $k$ -pass,  $s$ -space algorithm for  $Q_{t,f}$  with prob. at least  $1 - \delta$ , then there exists a  $(k - 1)$ -pass,  $2s$ -space algorithm for  $Q_{t/2,f}$  with prob. at least  $1 - 10\delta$ .

Note that the algorithm for  $f$  whose existence is proven in Lemma 3 does not succeed with the same probability as  $Q_{t,f}$ . Unfortunately this can not be fixed by parallel repetition because the algorithm for  $Q_{t,f}$  may write the solution to a write-only tape.

**Post-Order-Traversal:** We now define a data-stream problem related to pointer-chasing. The significant difference in our work is the order in which we encode a pointer-chasing instance in the data-stream. This will be essential in proving lower-bounds for order-dependent functions such as finding the elements of the longest increasing subsequence.

**Definition 4.** Consider a  $(p + 1)$ -level,  $t$ -ary tree  $T$  rooted at  $v_{root}$  and a function  $f : V(T) \rightarrow [t]$  where  $f(v) \in \{0, 1\}$  if  $v$  is a leaf of  $T$ . Define  $g(v)$  to be the  $f(v)$ -th child of  $v$  if  $v$  is an internal node and  $f(v)$  if  $v$  is a leaf. The POT problem to evaluate  $\text{POT}(f) = g^{p+1}(v_{root})$  given the stream  $\langle f(v_{\sigma(1)}), f(v_{\sigma(2)}), \dots, f(v_{\sigma(|T|)}) \rangle$ , where  $\sigma$  is a post-order-traversal of  $V(T)$ .

The next theorem follows by induction from Lemma 2.

**Theorem 5.** Any  $p$ -pass algorithm that solves the  $(p + 1)$ -level,  $t$ -ary POT problem (w/p.  $9/10$ ) requires  $\Omega(t/2^p)$  space.

**Strong-Post-Order-Traversal:** A strong-post-order-traversal of a tree is a variant of the post-order-traversal in which a partial pre-order traversal is made between the visit of each node on the post-order-traversal. While this may seem like a very artificial construction, it will be very important to some of our other lower bounds.

**Definition 6.** Consider a  $(p + 1)$ -level,  $t$ -ary tree  $T$  rooted at  $v_{root}$  and a function  $f : V(T) \rightarrow [t]$  where  $f(v) \in \{0, 1\}$  if  $v$  is a leaf of  $T$ . Define  $g(v)$  to be the  $f(v)$ -th child of  $v$ . The SPOT problem is to evaluate  $\text{SPOT}(f) = g^{p+1}(v_{root})$

<sup>3</sup>Note that  $Q_{t,f}$  can be solved in one pass if there is sufficient memory to solve  $f(x)$  in one pass as the  $f(x_i)$  can be computed sequentially. However, if  $f(x)$  can be solved in much less space given multiple passes, it is not clear if  $Q_{t,f}$  can be solved in less space.

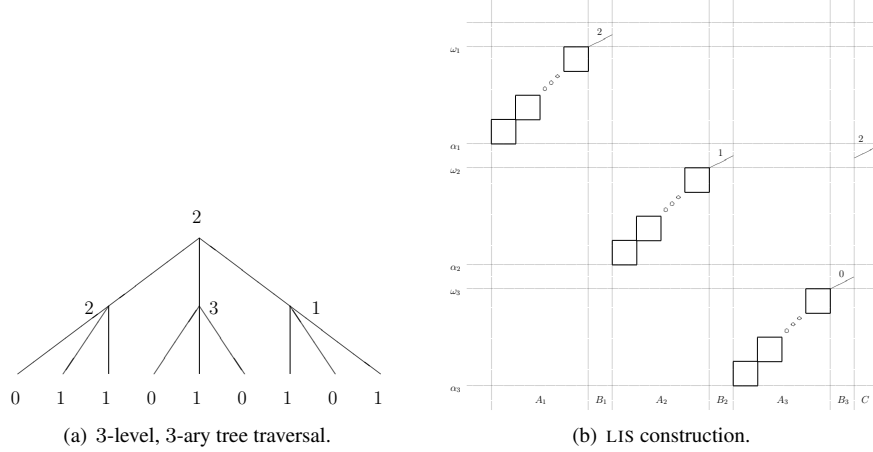


Figure 1: Part (a) shows an instance of  $(p + 1)$ -level,  $t$ -ary tree traversal with  $t = 3$ ,  $p = 2$ , and  $g^3(v_{\text{root}}) = 0$ . The POT stream is  $\langle 0, 1, 1, 2, 0, 1, 0, 3, 1, 0, 1, 1, 2 \rangle$ . The SPOT stream is  $\langle 0, 1, 1, 2, 0, 0, 1, 0, 3, 0, 1, 1, 0, 1, 1, 2, 2, 0, 1, 1 \rangle$ . Part (b) is referenced in Section 3.

given the stream  $\langle S(v_{\sigma(1)}), S(v_{\sigma(2)}), \dots, S(v_{\sigma(|T|)}) \rangle$ , where  $\sigma$  is a post-order-traversal of  $V(T)$  and  $S(v)$  is defined as follows:

$$S(v) = \begin{cases} \langle f(v) \rangle & \text{if } v \text{ is a leaf} \\ \langle f(v), R(u_1), \dots, R(u_{f(v)-1}) \rangle & \text{if } v \text{ has children } u_1, \dots, u_t \end{cases}$$

where  $R(u)$  is a pre-order traversal of the nodes in the sub-tree rooted at  $u$ .

The next theorem follows by induction from Lemma 2.

**Theorem 7.** Solving the  $(p + 1)$ -level  $t$ -ary SPOT problem (w/p. 9/10) in  $p$  passes requires  $\Omega(t/2^p)$  space.

The next theorem follows from Lemma 3 by induction.

**Theorem 8.** If solving  $f(x)$  in one pass (w/p. 9/10) requires  $s$  space then any  $p$ -pass algorithm for  $Q_{t,f}$  (w/p.  $1 - 1/10^p$ ) requires  $\Omega(s)$ -space if  $t > s(20)^{(p-1)p/2}$ .

## 2.1 Proof of Lemma 2

Consider an arbitrary distribution  $D$  over  $\mathcal{X}$ . By assumption and Yao's lemma there exists a deterministic  $k$ -pass  $s$ -space algorithm  $\mathcal{A}$  that, with probability at least  $1 - \delta$ , solves  $P_{t,f,g}$  correctly on the stream  $\langle X_1, \dots, X_t, i, g(X_1, \dots, X_{i-1}) \rangle$  where each  $X_j \sim D$  are independent and  $i \in_{\mathcal{R}} [t]$ . We will show that such an algorithm can be used to construct a deterministic  $(k - 1)$ -pass  $2s$ -space algorithm that solves  $f$  on  $\langle X_i \rangle$  with probability at least 9/10. Since  $D$  was arbitrary, by Yao's lemma, this shows that there is a randomized algorithm that solves  $P_{t,f}$  with probability at least 4/5 over an arbitrary input. Running  $O(\log \delta^{-1})$  copies in parallel and taking the median value results in an algorithm that is successful with probability at least  $1 - \delta$ .

First we introduce some further notation. Let the memory states of  $\mathcal{A}$  be  $\mathcal{M} = \{m_1, \dots, m_{2^s}\}$  where  $m_1$  is the initial memory state. Let  $\mathcal{A}(S; m_u)$  denote the memory state of  $\mathcal{A}$  after being initialized with memory state  $m_u$  and reading stream  $S$ . If  $m_u = m_1$  we omit it. Let  $\mathcal{C}$  be the set of inputs upon which  $\mathcal{A}$  is successful. By assumption,  $\Pr[\langle X_1, \dots, X_t, r, g(X_1, \dots, X_{r-1}) \rangle \in \mathcal{C}] \geq 1 - \delta$ .

**Lemma 9.** There exists  $r \in [t]$  and  $\langle x_1, x_2, \dots, x_{r-1} \rangle \in \mathcal{X}^{r-1}$  such that,

$$I(M^r; M^t) \leq \frac{9s}{t} \text{ and } \Pr[\langle x_1, \dots, x_{r-1}, X_r, \dots, X_t, r, g(x_1, \dots, x_{r-1}) \rangle \in \mathcal{C}] \geq 1 - 9\delta$$

where  $M^r = \mathcal{A}(x_1, \dots, x_{r-1}, X_r)$ ,  $M^t = \mathcal{A}(x_1, \dots, x_{r-1}, X_r, \dots, X_t)$ , and  $I(\cdot; \cdot)$  is the mutual information.

See the full version for the proof of the above lemma and the rest of the lemmas in this section. Next we consider the distribution of the memory states of the algorithm after processing various prefixes of the stream. Define the distributions,  $p_{u,v} = \Pr[M^r = m_u, M^t = m_v]$ ,  $p_u = \Pr[M^r = m_u]$ ,  $q_v = \Pr[M^t = m_v]$ , and  $p_u^v = p_{u,v}/q_v$ . Note that  $p^v$  is the distribution  $M^r$  conditioned on the event that  $\{M^t = v\}$ . Intuitively, if the mutual information between  $M^r$  and  $M^t$  is low then  $p^v$  is similar to  $p$ . The next lemma, a variant of the Average Encoding Theorem [21], substantiates this.

**Lemma 10.** *If  $v$  is chosen with probability  $q_v$  then the variational distance between the distribution of  $M^r$  and  $M^r$  conditioned on  $\{M^t = v\}$  is small if  $I(M^r; M^t)$  is small. Specifically,  $\Pr_{v \sim q_v} \left[ \sum_u |p_u - p_u^v| \leq 2\sqrt{3I(M^r; M^t)} \right] \geq 2/3$ .*

The next lemma shows that there exists a state  $m_v$  such that we may choose a continuation of the stream  $\langle x_1, \dots, x_{r-1}, X_r \rangle$  that a) is only a function of  $M^r$ , b) yields a stream upon which the algorithm is likely to succeed, and c) guarantees  $M^t = m_v$ .

**Lemma 11.** *There exists  $m_v \in \mathcal{M}$  and  $Z : \mathcal{M} \rightarrow \mathcal{X}^{t-r}$  such that  $\mathcal{A}(Z(m_u); m_u) = m_v$  and  $\Pr[\langle x_1, \dots, x_{r-1}, X_r, Z(M^r), r, g(x_1, \dots, 9/10, \delta < 1/1000 \text{ and } s < 50000t$ .*

Let  $m_w = \mathcal{A}(r, g(x_1, \dots, x_{r-1}); m_v)$ . We construct a  $(k-1)$ -pass algorithm  $\mathcal{A}'$  by running  $\mathcal{A}$  on the stream  $\langle x_1, \dots, x_{r-1}, X_r, Z(M^r), r, g(x_1, \dots, x_{r-1}) \rangle$  and essentially collapsing the first two passes of  $\mathcal{A}$  into a single pass. We compute  $m_u = \mathcal{A}(x_1, x_2, \dots, x_{r-1}, X_r)$  and  $m_x = \mathcal{A}(x_1, x_2, \dots, x_{r-1}, X_r; m_w)$  in parallel using a single pass. We store  $M^r$  for the rest of the algorithm and use it to construct  $Z(m_u)$  when necessary. Note that if  $Z(\cdot)$  is not defined on  $m_u$ , we may output an arbitrary value. We then compute the first pass of  $\mathcal{A}'$  by computing,  $\mathcal{A}(Z(m_u), r, g(x_1, \dots, x_{r-1}); m_x)$ . The remaining  $k-2$  passes of  $\mathcal{A}'$  emulate the remaining  $k-2$  passes of  $\mathcal{A}$  where we generate  $Z(m_u)$  from the stored value of  $m_u$  when necessary.

### 3 Longest Increasing Subsequence

An instance of the longest increasing subsequence (LIS) problem is, given a stream  $S = \langle x_1, \dots, x_m \rangle \in [n]^m$ , find the elements of a sub-stream of increasing values  $\langle x_{i_1}, \dots, x_{i_j} \rangle$  of maximum length. In a promise variant, the PLIS problem, we are promised  $|\text{LIS}(S)| \leq k$ . The best algorithmic result for this problem is due to Liben-Nowell et al. [22]: there exists a deterministic algorithm for PLIS that uses  $p$  passes and  $\tilde{O}(k^{1+1/(2^p-1)})$  space. Sun and Woodruff [30] established that Liben-Nowell et al.'s algorithm was essentially optimal for one-pass algorithms. In particular they showed that any one-pass algorithm for PLIS (with prob. 9/10) requires  $\Omega(k^2)$  space. It was left as an open question whether the doubly exponential trade-off between space and passes in Liben-Nowell et al.'s result was the true trade-off. We show that this is the case and present a matching lower-bound for PLIS by a reduction to the POT problem.

**Theorem 12.** *Solving PLIS (w/p. 9/10) in  $p$  passes requires  $\Omega(k^{1+\frac{1}{2^p-1}})$  space.*

*Proof.* Let  $t = m^{1/p}$ . We show a reduction from the  $(p+1)$ -level  $t$ -ary POT problem to computing the elements of a PLIS problem over universe  $[2^p mt]$  where the LIS is promised to have length  $k = \Theta(t^{1-1/2^p})$ . Let  $f$  be an instance of  $(p+1)$ -level  $t$ -ary POT. Let  $h_1 = 1$  and then recursively define

$$r_j := \left\lceil \sqrt{t/h_{j-1}} \right\rceil, \quad t_j := \lceil t/r_j \rceil, \quad \text{and} \quad h_j := r_j h_{j-1} + t_j.$$

By induction it can be shown that  $t^{1-1/2^j}/6 \leq h_{j+1} < 6t^{1-1/2^j}$ .

With each node  $v$  we associate a sequence of numbers,  $I(v)$ . If  $v$  is leaf then  $I(v) = (f(v))$ . If  $v$  is an internal node we will define  $I(v)$  inductively such that if  $v$  is at level  $j$ ,  $\text{LIS}(I(v)) = h_j$ . Let  $\text{children}(v) = \{u_0, \dots, u_{t-1}\}$ . We group the children  $(v)$  into  $t_j$  sets  $S_i = \{u_{(i-1)r_j + \ell} : \ell = 0, \dots, r_j - 1\}$ . Note that  $|S_1| = \dots = |S_{t_j-1}| = r_j$  but  $|S_{t_j}| = t - (t_j - 1)r$  may be smaller. The  $I(v)$  we construct will have the following form:  $I(v) = (A_1 : B_1 : \dots : A_{t_j} : B_{t_j} : C)$  where “:” denotes concatenation.  $A_i$  will be constructed from  $I(v)$  for  $v \in S_i$  and  $B_i$  will be an increasing sequence.  $C$  will be an increasing sequence of length  $a := \lceil f(v)/r_j \rceil$ . Together  $A_i$  and  $B_i$  and  $C$  will satisfy:

1.  $\text{LIS}(A_i : B_i) = r_j h_{j-1} + t_j - i$  and  $\max(A_i : B_i) < \min(A_{i-1} : B_{i-1})$
2.  $\text{LIS}(A_a : B_a : C) = h_j$
3.  $\max(A_a : B_a) < \min(C) < \max(C) < \min(A_{a-1} : B_{a-1})$

It follows that the elements of the longest increasing subsequence of  $I(v)$  include the elements of the longest increasing subsequence of  $A_a$ . See Figure 1(b).

**Constructing  $A_i, B_i$  and  $C$ :** We construct  $A_i$  and  $B_i$  as follows. First, let  $R_{j-1} \geq \max(I(u_0), \dots, I(u_{t-1}))$ . Let  $I_1(u_{ir_j+\ell})$  be the sequence formed by adding  $\ell R_{j-1}$  to each element of  $I(u_{ir_j+\ell})$ . Therefore for all  $i$ ,  $\max(I_1(u_{ir_j+\ell})) \leq |S_{i+1}| R_{j-1}$  and

$$\text{LIS}(I_1(u_{ir_j}) : \dots : I_1(u_{ir_j+|S_{i+1}|-1})) = |S_{i+1}| h_{j-1}.$$

Let  $I_2(u_{ir_j+\ell})$  be the sequence formed by adding  $(t_j - i)(r_j R_{j-1} + t_j)$  to each element of  $I_1(u_{ir_j+\ell})$ . We now define  $A_i = (I_2(u_{ir_j}) : \dots : I_2(u_{ir_j+|S_{i+1}|-1}))$ . For  $i \in [t_j - 1]$ ,

$$\begin{aligned} \min(A_i) &\geq (t_j - i)(r_j R_{j-1} + t_j) =: \alpha_i \\ \max(A_{i+1}) &\leq r_j R_{j-1} + (t_j - 1 - i)(r_j R_{j-1} + t_j) =: \omega_{i+1}. \end{aligned}$$

and therefore  $\alpha_i - \omega_{i+1} \geq t_j$ . Hence, letting

$$B_i = (\omega_i + 1, \omega_i + 2, \dots, \omega_i + (r_j h_{j-1} + t_j - i - |S_{i+1}| h_{j-1}))$$

ensures  $\text{LIS}(A_i : B_i) = r_j h_{j-1} + t_j - i$  and  $\max(A_i : B_i) < \min(A_{i-1} : B_{i-1})$ . Letting  $C$  be the sequence  $C = (\alpha_{a-1} - a, \dots, \alpha_{a-1} - 1)$  ensures all the necessary properties. One issue remains: one of the elements of longest increasing subsequence of the stream encodes the answer to the POT problem but it is not clear which one because the subsequence only encodes  $\lceil f(v)/r_j \rceil$  rather than  $f(v)$ . However, this can be fixed by encoding  $f(v)$  in the lower order bits of  $C$  with a factor  $t$  increase in universe size.  $\square$

## 4 Promise Min. Missing Element and Greater-Than

In this section, we consider the related problems of MME and GT. An instance of MME consists of  $n$  non-negative integers  $A$  and the goal to identify  $\text{MME}(A) = \min\{i \in \mathbb{N}_0 : i \notin A\}$ . There exists an  $\Omega(n/p)$  space lower-bound for any  $p$ -pass algorithm that solves this problem that can be proved using a reduction from SET-DISJOINTNESS. We are interested in a promise version of the problem in which we assume that all elements less than  $\text{MME}(A)$  occur only once in  $A$ . This changes the complexity of the problem considerably.

An instance of the GT consists of two  $n$ -bit strings  $x = x_1 \dots x_n$  and  $y = y_1 \dots y_n$  and the goal is to determine if  $x < y$ , i.e., if there exists  $i \in [n]$  such that  $x_i < y_i$  and  $x_j = y_j$  for all  $j < i$ . In the data-stream setting we assume that the stream consists of the elements  $\{(x_i, i), (y_i, i) : i \in [n]\}$  in some arbitrary order. Note that MME and GT are related. For example, if  $A = \{2(i-1) + x_i, 2(i-1) + 1 - y_i : i \in [n]\}$  then the parity of  $\text{MME}(A)$  is odd iff  $x < y$ . Note that a stream of such elements can be transduced from a stream of elements from  $\{(x_i, i), (y_i, i) : i \in [n]\}$  in constant space. Hence any lower-bound for MME also yields a lower-bound for GT.

Both problems were considered by Chang and Kannan [9] for the purposes of proving lower bounds on the ‘‘generalized histogram learning problem’’ (see [9] for details). Using the round-elimination lemma and results from [23, 29] they proved a space lower bound of  $\Omega(n^{1/(2p-1)})$  for any constant  $p$ -pass algorithm that solved either GT or MME. We improve this by a polynomial factor improvement when  $p > 1$ .

**Theorem 13.** *Solving GT or MME (w/p. 9/10) in  $p$  passes requires  $\Omega(n^{1/p})$  space.*

*Proof.* We reduce SPOT to MME. Let the function  $f$  be an instance of SPOT and let  $\{u_1, \dots, u_{t^p}\}$  be the leaves of a  $(p+1)$ -level,  $t$ -ary tree. With each leaf  $u_i$  we associate  $c(u_i) = 2(i-1) + f(u_i)$  and  $d(u_i) = 2(i-1) + 1 - f(u_i)$ .

The idea behind the reduction is that if  $g^p(v_{\text{root}}) = u_j$  then the smallest  $2j - 1$  elements of the stream will be  $\{c(u_i), d(u_i) : i < j\} \cup \{c(u_j)\}$  and the stream will not contain  $d(u_j)$ . Consequently  $d(u_j)$  will be the minimum missing element and hence,  $1 - (d(u_j) \bmod 2) = g^{p+1}(v_{\text{root}})$ .

For a leaf  $u$ , the reduction replaces  $f(u)$  by  $c(u)$  the first time the leaf is visited in the strong post-order traversal. If  $u$  is visited a second time then we replace  $f(u)$  by  $d(u)$  on this second visit. On all subsequent visits to  $u$ ,  $f(u)$  is ignored. For example, the SPOT instance in Figure 1(a) would become  $\langle 0, 3, 5, 1, 6, 9, 10, 7, 8, 13, 14, 17, 2, 4 \rangle$ . Note that the minimum missing element is 11 and  $f(g^p(v_{\text{root}})) = 1 - (11 \bmod 2) = 0$ .

We need to show that this reduction can be performed in a streaming fashion. In particular, we need to establish that it is possible to determine if a node in the strong post-order traversal has already been visited once or at least twice. We can recognize the first time we visit a leaf because the first visit to  $u_i$  is before the first visit to  $u_j$  if  $i < j$ . Hence, it suffices to remember the smallest leaf that has not been visited.

Leaf  $u$  is only revisited during a pre-order traversal of a sub-tree containing  $u$ . Consider a pre-order traversal of the  $j$ -level,  $t$ -ary sub-tree that contains  $u$ . Let  $v_j$  be the root of this sub-tree and let  $v_j, v_{j-1}, \dots, v_1$  be the path from root to leaf  $v_1 = u$ . Let  $v_{i-1}$  be the  $a_i$ -th child of  $v_i$ . Then  $u$  is being revisited for the first time if  $a_i < f(v_i)$  for all  $i = 2, \dots, j$ . Note that the set of relevant values of  $f$  can be maintained in  $O(p)$  space because the sub-tree is visited in pre-order.  $\square$

## 5 Fixed-Dimensional Linear Programming

In this section, we consider the problem of linear programming in  $\mathbb{R}^d$  where  $d$  is assumed constant. An instance of this problem consists of an objective function known to the algorithm and a set of  $n$  constraints each of which are specified by an element of the stream. In a recent paper, Chan and Chen [8] showed that any  $p$ -pass deterministic algorithm “of a certain type” for finding the lowest point in the intersection of  $n$  upper half planes in  $2D$  requires  $\Omega(n^{1/p})$  space. The algorithms considered were for the decision tree model where “the only allowable operations on the input half-planes are testing the sign of a function evaluated at the coefficients of a subset of half-planes currently in memory.” While these test functions could be any continuous function, the algorithm is restricted to storing points of the input only.

We show that the  $\Omega(n^{1/p})$  bound holds in  $3D$  for any algorithm, e.g., for algorithms that may be randomized or may store information other than the points themselves. For  $2D$ , the same bound holds if we allow one non-linear constraint.

**Theorem 14.** *Solving 3DLP ( $w/p, 9/10$ ) in  $p = O(1)$  passes requires  $\Omega(n^{1/p})$  space.*

*Proof.* Let  $t = n^{1/p}$ . We first reduce  $(p+1)$ -level  $t$ -ary POT to (the feasibility of)  $2D$  linear programming with the added non-linear constraint that  $x^2 + y^2 = 1$ . We then show how to remove this constraint using one extra dimension. The  $f$  be an instance of  $(p+1)$ -level  $t$ -ary POT. We consider constraints specified by chords on the bottom half of the unit circle. It will be convenient to use polar representation of points and represent every half-space by a chord through a pair of points. The reduction is as follows:

1. For each node  $v \in V(T)$  define two “end-points”  $s_v$  and  $t_v$ . If  $v$  is the root of  $T$  let  $s_v = (-1, 0)$  and  $t_v = (1, 0)$ . For any other node  $v$ , let  $v$  be the  $i$ -th child of node  $w$ . Consider partitioning the arc from  $s_w$  to  $t_w$  into  $t$  equi-length arcs and define  $s_v$  and  $t_v$  to be the end points of the  $i$ -th such arc. For each leaf, also define a “mid-point”  $m_v$  that is halfway along the arc from  $s_v$  to  $t_v$ .
2. For each internal node  $v \in V(T)$ : Add chords through  $s_v$  and  $s_{u_{f(v)}}$  and through  $t_{u_{f(v)}}$  and  $t_v$  where  $\{u_i : i \in [t]\}$  are the children of  $v$ .
3. For each leaf  $v \in V(T)$ : If  $f(v) = 0$ , add the chord through  $s_v$  and  $t_v$ .
4. Lastly, add “dead-zone” constraints. Let  $\{u_1, \dots, u_{t^p}\}$  be the leaves of the tree. For  $i \in [t^p]$ , add the chord between  $m_{u_i}$  and  $m_{u_{i+1}}$  where  $m_{u_{m^p+1}} = m_{u_1}$ .

The constraint at an internal node  $v$  ensures that any feasible points lies on the arc  $(s_v, t_v)$ . The constraints at a leaf node  $v$  determine if there exists a feasible point on the arc  $(s_v, t_v)$  given that the dead-zone constraints ensure that any feasible point is a mid-point. Consequently, there exists a feasible point on the unit circle iff  $\text{POT}(f) = 1$ . It is immediate that the number of constraints is  $O(n)$  and the smallest angle (at the origin) we construct in this process is  $\Theta(1/n)$ . Thus in a polar format the input is polynomially bounded. To remove the constraint  $x^2 + y^2 = 1$ , consider lifting the problem to the cone  $z = x^2 + y^2$ , and add the constraint  $z = 1$ . While the constraint  $z = x^2 + y^2$  is not linear we observe that we are only interested in  $O(n)$  predetermined points on the cone. Thus we can approximate the cone by  $O(n)$  planes.  $\square$

**Acknowledgments:** Thanks to Kook Jin Ahn and Matei David for helpful comments.

## References

- [1] M. Adler, E. D. Demaine, N. J. A. Harvey, and M. Patrascu. Lower bounds for asymmetric communication channels and distributed source coding. In *ACM-SIAM Symposium on Discrete Algorithms*, pages 251–260, 2006.
- [2] N. Alon, Y. Matias, and M. Szegedy. The space complexity of approximating the frequency moments. *Journal of Computer and System Sciences*, 58(1):137–147, 1999.
- [3] Z. Bar-Yossef, T. S. Jayram, R. Kumar, and D. Sivakumar. An information statistics approach to data stream and communication complexity. In *IEEE Symposium on Foundations of Computer Science*, pages 209–218, 2002.
- [4] P. Beame, T. S. Jayram, and A. Rudra. Lower bounds for randomized read/write stream algorithms. In *ACM Symposium on Theory of Computing*, pages 689–698, 2007.
- [5] A. Chakrabarti. Lower bounds for multi-player pointer jumping. In *IEEE Conference on Computational Complexity*, pages 33–45, 2007.
- [6] A. Chakrabarti, S. Khot, and X. Sun. Near-optimal lower bounds on the multi-party communication complexity of set disjointness. In *IEEE Conference on Computational Complexity*, pages 107–117, 2003.
- [7] A. Chakrabarti and O. Regev. An optimal randomised cell probe lower bound for approximate nearest neighbour searching. In *IEEE Symposium on Foundations of Computer Science*, pages 473–482, 2004.
- [8] T. M. Chan and E. Y. Chen. Multi-pass geometric algorithms. *Discrete & Computational Geometry*, 37(1):79–102, 2007.
- [9] K. L. Chang and R. Kannan. The space complexity of pass-efficient algorithms for clustering. In *ACM-SIAM Symposium on Discrete Algorithms*, pages 1157–1166, 2006.
- [10] M. Chu, S. Kannan, and A. McGregor. Checking and spot-checking of heaps. In *International Colloquium on Automata, Languages and Programming*, pages 728–739, 2007.
- [11] C. Demetrescu, I. Finocchi, and A. Ribichini. Trading off space for passes in graph streaming problems. In *ACM-SIAM Symposium on Discrete Algorithms*, pages 714–723, 2006.
- [12] F. Ergun and H. Jowhari. On the distance to monotonicity and longest increasing subsequence of a data stream. In *ACM-SIAM Symposium on Discrete Algorithms*, 2008.
- [13] J. Feigenbaum, S. Kannan, A. McGregor, S. Suri, and J. Zhang. Graph distances in the streaming model: the value of space. In *ACM-SIAM Symposium on Discrete Algorithms*, pages 745–754, 2005.
- [14] J. Feigenbaum, S. Kannan, A. McGregor, S. Suri, and J. Zhang. On graph problems in a semi-streaming model. *Theoretical Computer Science*, 348(2-3):207–216, 2005.
- [15] A. Gal and P. Gopalan. Lower bounds on streaming algorithms for approximating the length of the longest increasing subsequence. In *IEEE Symposium on Foundations of Computer Science*, 2007.
- [16] P. Gopalan, T. Jayram, R. Krauthgamer, and R. Kumar. Estimating the sortedness of a data stream. In *ACM-SIAM Symposium on Discrete Algorithms*, 2007.
- [17] S. Guha, N. Koudas, and K. Shim. Approximation and streaming algorithms for histogram construction problems. *ACM Trans. Database Syst.*, 31(1):396–438, 2006.
- [18] S. Guha and A. McGregor. Lower bounds for quantile estimation in random-order and multi-pass streaming. In *International Colloquium on Automata, Languages and Programming*, pages 704–715, 2007.

- [19] M. R. Henzinger, P. Raghavan, and S. Rajagopalan. Computing on data streams. *External memory algorithms*, pages 107–118, 1999.
- [20] P. Indyk. Stable distributions, pseudorandom generators, embeddings, and data stream computation. *J. ACM*, 53(3):307–323, 2006.
- [21] H. Klauck, A. Nayak, A. Ta-Shma, and D. Zuckerman. Interaction in quantum communication and the complexity of set disjointness. In *ACM Symposium on Theory of Computing*, pages 124–133, 2001.
- [22] D. Liben-Nowell, E. Vee, and A. Zhu. Finding longest increasing and common subsequences in streaming data. *J. Comb. Optim.*, 11(2):155–175, 2006.
- [23] P. B. Miltersen, N. Nisan, S. Safra, and A. Wigderson. On data structures and asymmetric communication complexity. *J. Comput. Syst. Sci.*, 57(1):37–49, 1998.
- [24] J. I. Munro and M. Paterson. Selection and sorting with limited storage. *Theor. Comput. Sci.*, 12:315–323, 1980.
- [25] N. Nisan and A. Wigderson. Rounds in communication complexity revisited. *SIAM J. Comput.*, 22(1):211–219, 1993.
- [26] M. Patrascu and M. Thorup. Time-space trade-offs for predecessor search. In *ACM Symposium on Theory of Computing*, pages 232–240, 2006.
- [27] M. Patrascu and M. Thorup. Randomization does not help searching predecessors. In *ACM-SIAM Symposium on Discrete Algorithms*, pages 555–564, 2007.
- [28] A. A. Razborov. On the distributional complexity of disjointness. *Theor. Comput. Sci.*, 106(2):385–390, 1992.
- [29] P. Sen. Lower bounds for predecessor searching in the cell probe model. In *IEEE Conference on Computational Complexity*, pages 73–83, 2003.
- [30] X. Sun and D. Woodruff. The communication and streaming complexity of computing the longest common and increasing subsequences. In *ACM-SIAM Symposium on Discrete Algorithms*, pages 336–345, 2007.

## A Misc. Proofs

*Proof of Lemma 9.* Let  $Y = \mathcal{A}(X_1, \dots, X_t)$ . By the chain rule for mutual information,

$$s \geq I(X_1, X_2, \dots, X_t; Y) = \sum_{r \in [t]} I(X_r; Y | X_1, \dots, X_{r-1}) .$$

Therefore, for at least  $2t/3$  values of  $r$ ,  $I(X_r; Y | X_1, X_2, \dots, X_{r-1}) \leq 3s/t$ . Similarly, for at least  $2t/3$  values of  $r$  such that

$$\Pr [\langle X_1, \dots, X_t, r, g(X_1, \dots, X_{r-1}) \rangle \in \mathcal{C}] \geq 1 - 3\delta.$$

Therefore there exists an  $r$  for which both are true. Now for  $2/3$  fraction of  $X_1, \dots, X_{r-1}$  we must have  $I(X_r; Y | X_1 = x_1, \dots, X_{r-1} = x_{r-1}) \leq 9s/t$  since  $I(X_r; Y | X_1, \dots, X_{r-1})$  is an expectation. Likewise,

$$\Pr_{\substack{x_1 = X_1, \dots, \\ x_{r-1} = X_{r-1}}} [\Pr [\langle x_1, \dots, x_{r-1}, X_r, \dots, X_t, r, g(x_1, \dots, x_{r-1}) \rangle \in \mathcal{C}] \geq 1 - 9\delta] \geq \frac{2}{3}.$$

Hence, the required  $\langle x_1, \dots, x_{r-1} \rangle$  exists. □

*Proof of Lemma 10.* We rewrite the mutual information as

$$I(M^r; M^t) = \sum_{u,v} p_{u,v} \log(p_{u,v}/p_u q_v) = \sum_v q_v \text{KL}(p^v, p) .$$

where KL is the Kullback-Leibler divergence. Therefore, noting that KL is always positive, by Markov's inequality,

$$\Pr_{v \sim q_v} [\text{KL}(p^v, p) \leq 3I(M^r; M^t)] \geq 2/3 .$$

However, by Pinsker's inequality,  $\text{KL}(p^v, p) \geq L_1^2(p^v, p)/4$ . □

*Proof of 11.* Denote the success probability of the algorithm conditioned on  $M^r = m_u$  and  $M^t = m_v$  as

$$s_{u,v} := \Pr [\langle x_1, \dots, x_{r-1}, X_r, \dots, X_t, r, g(x_1, \dots, x_{r-1}) \rangle \in \mathcal{C} | M^r = m_u, M^t = m_v] .$$

By Lemma 9, we know that  $\sum_{u,v} q_v p_u^v s_{u,v} = \sum_{u,v} p_{u,v} s_{u,v} \geq 1 - 9\delta$ . Therefore, by Markov's inequality,  $\Pr_{v \sim q_v} [\sum_u p_u^v s_{u,v} \geq 1 - 27\delta] \geq 2/3$ . Hence, by Lemma 10 and the union bound, with probability at least  $1/3$ ,

$$\Pr_{v \sim q_v} \left[ \sum_u p_u^v s_{u,v} \geq 1 - 27\delta \quad \text{and} \quad \sum_u |p_u - p_u^v| \leq 2\sqrt{27s/t} \right] \geq 1/3 .$$

Therefore there exists a  $v$  such that  $\sum_u p_u s_{u,v} \geq 1 - 27\delta - 2\sqrt{27s/t} \geq 9/10$ . Note that  $s_{u,v}$  is defined with respect to a random  $X_{r+1}, \dots, X_t$  subject to the condition  $\mathcal{A}(X_{r+1}, \dots, X_t; m_u) = m_v$ . Hence there exists a  $Z(\cdot)$  with the required properties. □

*Proof of Lemma 3.* The proof follows along very similar lines to the proof of Lemma 2. The main difference is that we do not condition the relevant events on specific values for a prefix of  $\langle X_1, \dots, X_t \rangle$ . There is also the issue that with an algorithm that may write-out to an output tape it can not be argued that the success probability can be boosted by repeating the algorithm many times and taking the median. We consider a deterministic algorithm  $\mathcal{A}$  acting on a stream  $\langle X_1, \dots, X_t \rangle$  where each  $X_i$  is distributed independently according to a given distribution  $D$ . We let  $\hat{\mathcal{C}}$  denote that set of input upon which the algorithm is successful, i.e.,

$$\hat{\mathcal{C}} := \{ \langle x_1, \dots, x_t \rangle : \mathcal{A} \text{ correctly solves } Q_{t,f} \text{ on } \langle x_1, \dots, x_t \rangle \} .$$

The proofs of arguments below are omitted as they are almost identical to the analogous results in the proof Lemma 2.

**Lemma 15.** *There exists  $r \in \{t/2, \dots, t\}$  such that  $I(\hat{M}^r; \hat{M}^t) \leq 2s/t$  where  $\hat{M}^r = \mathcal{A}(X_1, \dots, X_r)$  and  $\hat{M}^t = \mathcal{A}(X_1, \dots, X_t)$ .*

We define distributions analogous to the distributions in the proof Lemma 2.

$$\begin{aligned} \hat{p}_{u,v} &:= \Pr [\hat{M}^r = m_u, \hat{M}^t = m_v], \hat{p}_u := \Pr [\hat{M}^r = m_u], \\ \hat{q}_v &:= \Pr [\hat{M}^t = m_v], \text{ and } \hat{p}_u^v := \hat{p}_{u,v}/\hat{q}_v . \end{aligned}$$

**Lemma 16.** *If  $v$  is chosen with probability  $\hat{q}_v$  then the variational distance between the distribution of  $\hat{M}^r$  and  $\hat{M}^r$  conditioned on  $\{\hat{M}^t = v\}$  is small if  $I(\hat{M}^r; \hat{M}^t)$  is small. Specifically,*

$$\Pr_{v \sim \hat{q}_v} \left[ \sum_u |\hat{p}_u - \hat{p}_u^v| \leq 2\sqrt{3I(\hat{M}^r; \hat{M}^t)} \right] \geq 2/3 .$$

**Lemma 17.** *There exists  $m_v \in \mathcal{M}$  and function  $\hat{Z} : \mathcal{M} \rightarrow \mathcal{X}^{t-r}$  such that  $\mathcal{A}(\hat{Z}(m_u); m_u) = m_v$  and  $\Pr [\langle x_1, \dots, x_{r-1}, X_r, \hat{Z}(\hat{M}^r) \rangle \in \mathcal{C}] \geq 1 - 5\delta$ , if  $s \leq \delta t/2$ .*

We can use the existence  $\hat{Z}(\cdot)$  to construct a  $(k-1)$ -pass algorithm  $\mathcal{A}'$  from  $\mathcal{A}$  that, with probability at least  $1 - 5\delta$ , writes-out  $(i, f(x_i))$  for all  $i \in [t/2]$  in some order. This follows along similar lines to Lemma 2. The one important difference is that since  $\mathcal{A}$  writes-out  $(i, f(x_i))$  for all  $i \in [t]$  we need to add an auxiliary algorithm between  $\mathcal{A}$  and the output tape. This algorithm ensures that  $(i, f(x_i))$  is only written to the output tape if  $i \in [t/2]$ . □

## B Convex Hulls

In this section, we consider the problem of constructing the convex hull of  $n$  points in  $\mathbb{R}^2$ . In the streaming setting we assume that we may write out the points that specify the convex hull as we process the stream rather than having to store these points in working memory until the entire convex hull has been determined.

We will prove lower-bounds for two variants of this problem. In the first, we are promised that the stream of these points is sorted by the first coordinate of the points and may write out the points of the convex hull in any order. Our results here generalize a result for deterministic algorithms due to Chan and Chen [8]. In the second variant we make no assumption on the ordering of the stream but insist that our algorithm outputs the points of the convex hull sorted by the  $x$ -coordinate.

### B.1 Sorted Input-Order and Arbitrary Output-Order

Chan and Chen [8] showed that any constant pass, deterministic algorithm that reads a stream of  $n$  points sorted by  $x$ -coordinate and prints out the vertices of the upper hull (in any order) requires  $\Omega(\sqrt{n/\log n})$  space. Their proof is information-theoretic, but “it is assumed that to print a point to the output stream, the point must currently reside in memory”, that is, the algorithm rules out compressing a subset of points together and reconstructing them as the next few bits are read. We reiterate that their lower-bound is only for deterministic algorithms.

We show that the bound holds for any unrestricted, randomized algorithm. The reduction we use is similar to the reduction in [8], but differs in important details.

**Theorem 18.** *Consider a set of  $2D$  points with polynomially-bounded precision in a sorted order of  $x$ -coordinate. Any  $O(1)$ -pass algorithm that outputs exactly the points on the convex hull with probability  $1 - \delta$  requires  $\Omega(\sqrt{n})$  space for sufficiently small constant  $\delta$ .*

*Proof.* Given an arc  $(s_v, t_v)$  on the upper part of the unit circle we first subdivide into 2 halves  $(s_v, b_v), (b_v, t_v)$ . The first half is further subdivided into  $r$  (which will be set later) equal pieces  $\{(s_v(i), t_v(i))\}$  of length  $\ell$  each. Given an  $r$  bit string  $\sigma \in \{0, 1\}^r$ , if  $\sigma[i] = 0$  we choose the point  $p_v(i)$  which is  $\ell/3$  from  $s_v(i)$ , otherwise if  $\sigma[i] = 1$  we chose the point  $p_v(i)$  to be  $\ell/3$  from  $t_v(i)$  (and in both cases  $p_v(i)$  is inside  $(s_v(i), t_v(i))$ ). Given an index (this will involve a final reduction from indexing)  $j \in [r]$  we push  $b_v$  radially upward to  $b'_v$  so that the tangent from  $b'_v$  to the circle is at  $t_v(j)$ . It is easy to see that the precision needed (in polar coordinates) is polynomial. Note that the pushing upward limits the other tangent from  $b'_v$  to touch the circle in the range  $(b_v, t_v)$ . This was the reason why the interval was split in  $1/2$ . The final collection of points are  $\{s_v(i)\} \cup \{p_v(i)\} \cup \{t_v(i)\} \cup \{b'_v\}$ .

Given an indexing problem where Alice has  $\sigma$  and Bob has  $j$  we can in a single pass transform the indexing problem to a convex hull problem, where the position of the *third last* point of the hull reveals  $\sigma[j]$ . Thus a single pass algorithm for this gadget  $G_v$  would require  $\Omega(r)$  space. Now let  $r = \sqrt{n}$  and consider  $m = \sqrt{n}$  copies of  $G_v$  (which we know, by construction, do not interact with each other). Suppose we wish to find all the points on the convex hull (in some arbitrary order), we will determine all the penultimate points and solve all the  $G_v$ . This defines a  $Q_{t,f}$  problem where  $f = G_v$ . Now by Theorem 8, if we use less than  $cm$  bits for some constant  $c > 0$  and  $p = O(1)$  passes and succeed  $O(1)$  probability then we can solve a problem  $G_v$  in one pass with  $O(1)$  probability, which implies an  $\Omega(r)$  bound.  $\square$

### B.2 Arbitrary Input-Order and Sorted Output-Order

In this section we prove a *new* lower bound for  $2D$  output sensitive convex hull: given that the upper hull is guaranteed to be of size  $h$ , (where  $h = o(\sqrt{n})$  because of the sorted order lower bound result) any  $O(1)$  pass randomized algorithm requires  $\Omega(h)$ . As a consequence the lower bound for  $2D$  convex hull of arbitrarily order points using  $p = O(1)$  passes is  $\Omega(h + n^{1/p})$ , almost matching the upper bounds of [8] which uses the product of these two terms and a few more passes.

**Theorem 19.** *Consider a set of  $2D$  points with polynomially bounded precision in arbitrary order such that the convex hull is of size  $h < \sqrt{n}$ . Any  $O(1)$ -pass algorithm that outputs exactly the the points on the convex hull with probability at least  $1 - \delta$  requires  $\Omega(h)$  space for some sufficiently small constant  $\delta$ .*

*Proof.* The reduction is similar to that used in Theorem 18 except we partition  $(s_v, a_v), (a_v, b_v), (b_v, t_v)$ . The  $p(i)$  are chosen in the region  $(a_v, b_v)$ . Except now, we push  $a_v$  (which breaks sorted order of input) and  $b_v$  upwards (with the knowledge of the index  $j$ ) to  $a'_v, b'_v$  such that the tangent from  $a'_v$  touches the circle at  $s_v(j)$ . The tangent from  $b'_v$  touches at  $t_v(j)$  as before. This reduces the convex hull of these set of points to size 5, and does not interfere with the circle anywhere else. This is an indexing problem of size  $r$ , denoted by  $G'_v$ . Now we can define a problem  $Q_{h, G'_v}$  which corresponds to solving  $h$  of these problems in parallel. If the space used for  $p = O(1)$  passes is smaller than  $\Omega(h)$  then we again have to solve indexing with  $O(1)$  probability. Note that this needs  $r > h$  and therefore  $h \leq \sqrt{n}$ .  $\square$