

Lower Bounds for Quantile Estimation in Random-Order and Multi-Pass Streaming

Sudipto Guha^{*1} and Andrew McGregor²

¹ University of Pennsylvania
sudipto@cis.upenn.edu

² University of California, San Diego
andrewm@ucsd.edu

Abstract. We present lower bounds on the space required to estimate the quantiles of a stream of numerical values. Quantile estimation is perhaps the most studied problem in the data stream model and it is relatively well understood in the basic single-pass data stream model in which the values are ordered adversarially. Natural extensions of this basic model include the *random-order model* in which the values are ordered randomly (e.g. [21, 5, 13, 11, 12]) and the *multi-pass model* in which an algorithm is permitted a limited number of passes over the stream (e.g. [6, 7, 1, 19, 2, 6, 7, 1, 19, 2]). We present lower bounds that complement existing upper bounds [21, 11] in both models. One consequence is an exponential separation between the random-order and adversarial-order models: using $\Omega(\text{polylog } n)$ space, exact selection requires $\Omega(\log n)$ passes in the adversarial-order model while $O(\log \log n)$ passes are sufficient in the random-order model.

1 Introduction

One of the principal theoretical motivations for studying the data stream model is to understand the impact of the order of the input on computation. While an algorithm in the RAM model can process the input data in an arbitrary order, a key constraint of the data stream model is that any algorithm must process (in small space) the input data in the order in which it arrives. Parameterizing the number of passes that an algorithm may have over the data establishes a spectrum between the RAM model and the one-pass data stream model. How does the computational power of the model change along this spectrum? Furthermore, what role is played by the ordering of the stream?

These issues date back to one of the earliest papers on the data stream model in which Munro and Paterson considered the problems of sorting and selection in limited space [21]. They showed that $\tilde{O}(n^{1/p})$ space was sufficient to find the exact median of a sequence of n numbers given p passes over the data. However, if the data was randomly ordered, $\tilde{O}(n^{1/(2p)})$ space sufficed. They

* This research was supported by in part by an Alfred P. Sloan Research Fellowship and by NSF Awards CCF-0430376, and CCF-0644119.

also showed lower bounds for deterministic algorithms that stored the stream values as indivisible objects and uses a comparison based model. Specifically, they showed that all such algorithms required $\Omega(n^{1/p})$ space in the adversarial-order model and that single-pass algorithms that maintain a set of “elements whose ranks among those read thus far are consecutive and as close to the current median as possible” require $\Omega(\sqrt{n})$ space in the random-order model. They also conjectured the existence of an algorithm in the random-order model that used $O(\log \log n)$ passes and $O(\text{polylog } n)$ space to compute the median exactly. Median finding or quantile estimation has since become one of the most extensively studied problems in the data stream model [17, 18, 10, 9, 14, 4, 23, 3]. However, it was only recently shown that there does indeed exist an algorithm which uses $O(\log \log n)$ passes and $O(\text{polylog } n)$ space in the random-order model [11]. This result was based on a single-pass algorithm in the random-order model that, with high probability, returned an element of rank $n/2 \pm O(n^{1/2+\epsilon})$ and used $\text{poly}(\epsilon^{-1}, \log n)$ space. In contrast, any algorithm in the adversarial-order model requires $\Omega(n^{1-\delta})$ space to find an element of rank $n/2 \pm n^\delta$. These two facts together showed that the random-order model is strictly more powerful than the adversarial-order model.

Based on the algorithms of Munro and Paterson, it seemed plausible that any p pass algorithm in the random order stream model can be simulated by a $2p$ pass algorithm in the adversarial streaming model. This was conjectured by Kannan [15]. Further support for this conjecture came via work initiated by Feigenbaum et al. [8] that considered the relationship between various property testing models and the data-stream model. It was shown in Guha et al. [13] that several models of property testing can be simulated in the single-pass random-order stream model while it appeared that a similar simulation in the adversarial-model required two passes. While this appeared to support the conjecture, the conjecture remained unresolved.

In this paper we show that the conjecture is false. In fact, the separation between the random-order model and the adversarial-order model can be exponential. We show that using p passes, $\Omega(n^{1/p} p^{\Theta(1)})$ -space is required to compute the median exactly. This is a fully general lower bound as opposed to the lower bound for a restricted class of algorithms presented in [21]. Our proof is information-theoretic and uses a reduction from the communication complexity of a generalized form of pointer-chasing for which we prove the first lower-bound. It is also possible to establish a weaker lower bound using our reduction combined with the round-elimination lemma of Miltersen et al. [20] or the standard form of pointer-chasing considered by Nisan and Wigderson [22] as opposed our new lower bound for generalized pointer-chasing. We omit the details but stress that our communication complexity result for generalized pointer chasing is necessary to prove the stronger bound. Furthermore, we believe that this result may be useful for obtaining improved lower-bounds for other streaming problems.

A final question is whether it is possible to significantly improve upon the algorithm presented in [11] for the random-order model. In particular, does there exist a one-pass sub-polynomial approximation in $O(\text{polylog } n)$ -space? We show

that this is not the case and, in particular, a single-pass algorithm returning the exact median requires $\Omega(\sqrt{n})$ space in the random-order model. This result is about fully general algorithms in contrast to the result by Munro and Paterson [21]. We note that this is the first unqualified lower bound in the random-order model. The proof uses a reduction from communication complexity but deviates significantly from the usual form of such reductions because of the novel challenges arising when proving a lower bound in the random-order model as opposed to the adversarial-model.

1.1 Summary of Results and Overview

Our two main results of this paper are lower-bounds for approximate median finding in the random-order stream model and the multi-pass stream models.

In Section 3, we prove that any algorithm that returns an n^δ -approximate median of a randomly ordered stream with probability at least $3/4$ requires $\Omega(\sqrt{n^{1-3\delta}/\log n})$ space. This rules out sub-polynomial approximation using poly-logarithmic space.

In Section 4, we prove that any algorithm that returns an n^δ -approximate median in k passes of an adversarially ordered stream requires $\Omega(n^{(1-\delta)/k} k^{-6})$ space. This disproves the conjecture that stated that any problem that could be solved in $k/2$ passes of a randomly ordered stream could be solved in at most k passes of an adversarially ordered stream [15].

We also simplify and improve the upper bound in [11] and show that there exists a single pass algorithm using $O(1)$ words of space that, given any k , returns an element of rank $k \pm O(k^{1/2} \log^2 k)$ if the stream is randomly ordered. This represents an improvement in terms of space use and accuracy. However, this improvement is not the focus of the paper and can be found in Appendix A.

2 Preliminaries

We start by clarifying the definition of an approximate quantile of a multi-set.

Definition 1 (Rank and Approximate Selection). *The rank of an item x in a set S is defined as, $\text{RANK}_S(x) = |\{x' \in S \mid x' < x\}| + 1$. Assuming there are no duplicate elements in S , we say x is an \mathcal{T} -approximate k -rank element in S if, $\text{RANK}_S(x) = k \pm \mathcal{T}$. If there are duplicate elements in S then we say x is an \mathcal{T} -approximate k -rank element if there exists some way of ordering identical elements such that x is an \mathcal{T} -approximate k -rank element.*

3 Random Order Lower-Bound

In this section we will prove a lower bound of the space required to n^δ -approximate the median in a randomly ordered stream. Our lower-bound will be based on a reduction from the communication complexity of indexing [16]. However, the reduction is significantly more involved than typical reductions because different

segments of a stream can not be determined independently by different players if the stream is in random order.

Let Alice have a binary string σ of length $s' = \epsilon n^{-\delta} \sqrt{n_2} / (100 \ln(2/\epsilon))$ and let Bob have an index $r \in [s']$ where ϵ and n_2 will be specified shortly. It is known that for Bob to learn σ_r with probability at least $3/4$ after a single message from Alice then the message Alice sends must be $\Omega(s')$ bits. More precisely,

Theorem 1. *There exists a constant c^* such that $R_{1/4}^1(\text{INDEX}) \geq c^* s'$.*

The basic idea of our proof is that if there exists an algorithm \mathcal{A} that computes the median of a randomly ordered stream in a single pass then this gives rise to a 1-way communication protocol that solves INDEX. The protocol is based upon simulating \mathcal{A} on a stream of length n where Alice determines the first $n_1 = n - c^* n^{1-\delta} / (4 \log n)$ elements and Bob determines the remaining $n_2 = c^* n^{1-\delta} / (4 \log n)$ elements. The stream consists of the following sets of elements:

1. S : A set of $s = n^\delta s'$ elements $\bigcup_{j \in [n^\delta]} \{2i + \sigma_i : i \in [s']\}$. Note that each of the s' distinct elements occurs n^δ times. We refer to S as the “special” elements.
2. X : $x = (n + 1)/2 - r$ copies of 0.
3. Y : $y = (n - 1)/2 - s + r$ copies of $2s + 2$.

Note that any n^δ -approximate median of $U = S \cup X \cup Y$ is $2r + \sigma_r$.

The difficulty in the proof comes from the fact that the probability that \mathcal{A} finds an n^δ -approximate median depends on the random ordering of the stream. Hence, it would seem that Alice and Bob need to ensure that the ordering of U in the stream is chosen at random. Unfortunately that is not possible without excessive communication between Alice and Bob. Instead we will show that it is possible for Alice and Bob to generate a stream in “semi-random” order according to the following notion of semi-random.

Definition 2 (ϵ -Generated Random Order). *Consider a set of elements $\{x_1, \dots, x_n\}$. Then $\sigma \in \text{Sym}_n$ defines a stream $\langle x_{\sigma(1)}, \dots, x_{\sigma(n)} \rangle$ where Sym_n is the set of all permutations on $[n]$. We say the ordering of this stream is ϵ -Generated Random if σ is chosen according to some distribution ν such that $\|\mu - \nu\|_1 \geq \epsilon$ where μ is the uniform distribution over all possible orderings.*

The importance of this definition is captured in the following simple lemma.

Lemma 1. *Let \mathcal{A} be a randomized algorithm that estimates some property of a randomly ordered stream such that the estimate satisfies some guarantee with probability at least p . Then the estimate returned by running \mathcal{A} on a stream in ϵ -generated random order satisfies the same guarantees with probability at least $p - \epsilon$.*

Proof. We say the \mathcal{A} succeeds if the estimate returns satisfies the required guarantees. Let $\Pr_{\mu, \text{coin}}(\cdot)$ denote the probability of an event over the internal coin

tosses of \mathcal{A} and the ordering of the stream when the stream order is chosen according to distribution μ . Similarly define $\Pr_{\nu, \text{coin}}(\cdot)$ where ν is any distribution satisfying $\|\mu - \nu\|_1 \leq \epsilon$.

$$\Pr_{\mu, \text{coin}}(\mathcal{A} \text{ succeeds}) = \sum_{\sigma \in \text{Sym}_n} \Pr_{\mu}(\sigma) \Pr_{\text{coin}}(\mathcal{A} \text{ succeeds}|\sigma) \leq \Pr_{\nu, \text{coin}}(\mathcal{A} \text{ succeeds}) + \epsilon .$$

Consequently, if we can show that Alice and Bob can generate a stream that is in $O(\epsilon)$ -generation random order then by appealing to Lemma 1 we can complete the proof.

Let A be a set of n_1 elements in U and $B = U \setminus A$ be a set of n_2 elements. A will be chosen randomly according to one of two distributions. We consider the following families of events.

$$E_t = \{a : |A \cap X| = |A \cap Y| + t\} \text{ and } S_{s_1} = \{a : |A \cap S| = s_1\}.$$

We define two distributions μ and μ' . Let μ be the distribution where A is chosen uniformly at random from all subsets of size n_1 of U . Note that,

$$\begin{aligned} \Pr_{\mu}(S_{s_1}) &= \binom{n_1}{s_1} \binom{n_2}{s_2} / \binom{n}{s} \\ \Pr_{\mu}(E_t | S_{s_1}) &= \binom{n_1 - s_1}{(n_1 - s_1)/2 - t/2} \binom{n_2 - s_2}{x - (n_1 - s_1)/2 + t/2} / \binom{n - s}{x} \\ \Pr_{\mu}(\{a\} | E_t, S_{s_1}) &= \begin{cases} \frac{1}{|E_t \cap S_{s_1}|} & \text{if } a \in E_t \cap S_{s_1} \\ 0 & \text{otherwise} \end{cases} \end{aligned}$$

where $s_1 + s_2 = s$. Note that the above three equations fully specify μ since

$$\Pr_{\mu}(\{a\}) = \sum_{t, s_1} \Pr_{\mu}(\{a\} | E_t, S_{s_1}) \Pr_{\mu}(E_t | S_{s_1}) \Pr_{\mu}(S_{s_1}).$$

Let μ' be a distribution on A where $\Pr_{\mu'}(S_{s_1}) = \Pr_{\mu}(S_{s_1})$, $\Pr_{\mu'}(\{a\} | E_t, S_{s_1}) = \Pr_{\mu}(\{a\} | E_t, S_{s_1})$ and,

$$\Pr_{\mu'}(E_t | S_{s_1}) = \binom{n_1 - s_1}{(n_1 - s_1)/2 - t/2} \binom{n_2 - s_2}{(n_2 - s_2)/2 + t/2} / \binom{n - s}{(n - s)/2}$$

where $s_1 + s_2 = s$. Note that $\mu' = \mu$ if $r = s/2$. The crux of the proofs is that μ and μ' are closely related even if r is as small as 1 or as large as s .

Lemma 2. *If $s_1 \leq \frac{\epsilon \sqrt{n_2}}{100 \ln(2/\epsilon)}$ and $t < t^*$ where $t^* = \sqrt{2n_2 \ln(2/\epsilon)} + s$ then,*

$$\frac{1}{1 + \epsilon} \leq \frac{\Pr_{\mu}(E_t | S_{s_1})}{\Pr_{\mu'}(E_t | S_{s_1})} \leq 1 + \epsilon .$$

We omit the proof of this lemma and subsequent lemmas whose proofs, while detailed, do not require any non-standard ideas. Next, we ascertain that it is sufficient to consider only values of $t < t^*$.

Lemma 3. $E^* := \bigcup_{|t| < t^*} E_t$ is a high probability event under μ' and μ , i.e., $\min(\Pr_\mu(E^*), \Pr_{\mu'}(E^*)) \geq 1 - \epsilon$.

Let S^{**} be the event that the number of distinct special items in the suffix of the stream is at most $s^{**} := c^* s' / (2 \log(n))$, i.e., $S^{**} = \{|\{i \in [s'] : 2i + \sigma_i \in B\}| < s^{**}\}$.

Lemma 4. S^{**} is a high probability event, i.e. $\Pr_{\mu'}(S^{**}) = \Pr_\mu(S^{**}) \geq 1 - \exp(-s' c^* / (13 \log n))$. This is greater than $1 - \epsilon$ for sufficiently large n .

Let ν be the distribution μ' conditioned on the events S^{**} and E^* . Alice and Bob can easily determine the prefix and suffix of a stream according to this distribution:

1. Alice randomly places the special items such that at most $c^* s' / (2 \log n)$ distinct elements occur in the suffix, and chooses a value t with probability $\Pr_{\mu'}(E_t | S^{**}) / (1 - \Pr_{\mu'}(E^* | S^{**}))$. She then randomly places $(n_1 - s_1 - t) / 2$ “0”’s and $(n_1 - s_1 + t) / 2$ “2s+2”’s and the special items she assigned to the suffix. She then sends $S' = \{(i, \sigma_i) : 2i + \sigma_i \notin \text{prefix of stream}\}$ (note that this is a multi-set in general) to Bob along with the value of t .
2. Bob randomly places $x - (n_1 - s - t) / 2$ “0”’s and $y - (n_1 - s + t) / 2$ “2s+2”’s and $\{2i + \sigma_i : (i, \sigma_i) \in S'\}$ in the suffix of the stream.

To prove our result we need to show that ν is sufficiently close to μ . This can be shown by appealing to Lemmas 3 and 4.

Lemma 5. ν is 5ϵ -near to random, i.e., $\|\mu - \nu\|_1 \leq 5\epsilon$.

Theorem 2. *Computing an n^δ -approximate median of a random order stream with probability at least 9/10 requires $\Omega(\sqrt{n^{1-3\delta}} / \log(n))$ space.*

Proof. Let \mathcal{A} be an algorithm using M bits of memory that returns the median of a randomly ordered stream with probability 9/10 (over both the ordering and the private coin tosses). Assume Alice and Bob generate the stream as described above. In addition, Alice runs \mathcal{A} on the prefix of the stream and sends the memory state to Bob when she is done. Bob then continues running \mathcal{A} , initialized with the transmitted memory state, on the suffix of the stream. Bob then returns 1 if the output of the algorithm is odd and 0 otherwise. By Lemma 1 and Lemma 5 this protocol is correct with probability $9/10 - 5\epsilon$.

We now bound the number of bits required to follow the above protocol. The number of bits required to specify a sub-set of the unique elements of S of size at most s^{**} is

$$\lg \sum_{0 \leq s_2 \leq s^{**}} \binom{s'}{s_2} \leq \lg(s^{**} + 1) + s^{**} \lg \left(\frac{s' e}{s^{**}} \right) .$$

For each unique element occurring in the suffix of the stream we need to specify how many time it occurs and the associated bit value of σ . This takes at most

$s^{**}(1 + \lg n)$ bits. Hence the number of bits transmitted in the protocol is at most

$$\lg s^{**} + s^{**} \lg \left(\frac{s'e}{s^{**}} \right) + s^{**}(1 + \lg n) + \lg n + M,$$

Assuming that $s^{**} = \omega(\lg n)$ this is bounded above by $\frac{3}{4}c^*s + M$ and hence we conclude that $M = \Omega(s')$ by appealing to Theorem 1.

4 Adversarial Order Lower-Bound

In this section we will prove that any k pass algorithm that returns the median of an adversarially ordered stream must use $\tilde{\Omega}(n^{1/k})$ space. This, coupled with the upper bound of Munro and Paterson [21], will resolve the space complexity of multi-pass algorithms for median finding up to poly-logarithmic terms. The proof will use a reduction from the communication complexity of a generalized form of pointer chasing that we now describe.

Definition 3 (Generalized Pointer Chasing). For $i \in [k]$, let $f_i : [m] \rightarrow [m]$ be an arbitrary function. Then g_k is defined by

$$g_k(f_1, f_2, \dots, f_k) = f_k(f_{k-1}(\dots f_1(1)\dots)) .$$

Let the i -th player, P_i , have function f_i and consider a protocol in which the players must speak in the reverse order, i.e., $P_k, P_{k-1}, \dots, P_1, P_k, \dots$. We say the protocol has r rounds if P_k speaks r times. Let $R_\delta^r(g_k)$ be the total number of bits that must be communicated in an r -round (randomized) protocol for P_1 to learn g_k with probability at least $1 - \delta$.

Note that $R_0^k(g_k) = O(k \log m)$. We will be looking at k round protocols. The proof of the next result follows along similar lines to [22] and will be proved in the Appendix B.

Theorem 3. $R_{1/10}^{k-1}(g_k) = \Omega(m/k^4 - k^2 \log m)$.

The next theorem is shown by reducing generalized pointer-chasing to approximate selection.

Theorem 4. Finding an n^δ -approximate median in k passes of an adversarially ordered stream requires $\Omega(n^{\frac{1-\delta}{k}} k^{-6})$.

Proof. We will show how a k -pass algorithm \mathcal{A} that computes a t -approximate median of a length n stream gives rise to a k -round protocol for computing g_{k+1} when $m = (n/((k+1)(2t+1)))^{1/k}/2$. If \mathcal{A} uses M bits of space then the protocol uses at most $(k(k+1) - 1)M$ bits. Hence by Theorem 3, this implies that $M = \Omega(m/k^6) = \Omega((n/t)^{1/k} k^{-6})$.

The intuition behind the proof is that any t -approximate median will correspond to a number $g_1 g_2 g_3 \dots g_{k+1}$ written in base $m + 2$. The input of P_1 will

S_1	S_2	S_3
$(0, 0, 0) \times 5(3 - f_A(1))$	$(1, 0, 0) \times (3 - f_B(1))$	$(1, 1, f_C(1)), (1, 2, f_C(2)), (1, 3, f_C(3))$
	$(1, 4, 0) \times (f_B(1) - 1)$	
	$(2, 0, 0) \times (3 - f_B(2))$	$(2, 1, f_C(1)), (2, 2, f_C(2)), (2, 3, f_C(3))$
	$(2, 4, 0) \times (f_B(2) - 1)$	
	$(3, 0, 0) \times (3 - f_B(3))$	$(3, 1, f_C(1)), (3, 2, f_C(2)), (3, 3, f_C(3))$
	$(4, 4, 0) \times (f_B(3) - 1)$	
$(4, 0, 0) \times 5(f_A(1) - 1)$		

Table 1. Reduction from Pointer Chasing to Exact Median Finding. A triple of the form (x_2, x_1, x_0) corresponds to the numerical value $x_2 \cdot 5^2 + x_1 \cdot 5^1 + x_0 \cdot 5^0$. Note that $\text{median}(S_1 \cup S_2 \cup S_3) = f_A(1) \cdot 5^2 + f_B(f_A(1)) \cdot 5^1 + f_C(f_B(f_A(1))) \cdot 5^0$

first determine the highest order ‘bit’, i.e., g_1 . Then the input of P_2 will determine the g_2 and so on. Specifically, each player P_i will determine a segment of the stream S_i : P_{k+1} determines the first $n_{k+1} = |S_{k+1}|$ elements, P_k determines the next $n_k = |S_k|$, etc. These segments are defined as follows,

$$S_1 = \left\{ \underbrace{0, \dots, 0}_{(m-f_1(1))(2t+1)(2m-1)^{k-1}}, \dots, \underbrace{(m+1)b^k, \dots, (m+1)b^k}_{(f_1(1)-1)(2t+1)(2m-1)^{k-1}} \right\}$$

and for $j \in \{2, \dots, k\}$,

$$S_j = \bigcup_{x_{k+2-j}, \dots, x_k \in [m]} \left\{ \underbrace{\sum_{i=k+2-j}^k x_i b^i, \dots, \sum_{i=k+2-j}^k x_i b^i}_{(m-f_j(x_{k+2-j}))(2t+1)(2m-1)^{k-j}}, \right. \\ \left. \underbrace{(m+1)b^{k+1-j} + \sum_{i=k+2-j}^k x_i b^i, \dots, (m+1)b^{k+2-j} + \sum_{i=k+2-j}^k x_i b^i}_{(f_j(x_{k+2-j})-1)(2t+1)(2m-1)^{k-j}} \right\},$$

and finally,

$$S_{k+1} = \bigcup_{x_1, \dots, x_k \in [m]} \left\{ \underbrace{f_{k+1}(x_1) + \sum_{i=1}^k x_i b^i, \dots, f_{k+1}(x_1) + \sum_{i=1}^k x_i b^i}_{2t+1} \right\},$$

where $b = m + 2$. See Table 1 for the an example when $k = 2$ and $m = 3$. Note that $n_{k+1} = (2t + 1)m^k$ and for $j \geq k$, $n_j = (2t + 1)(m - 1)(2m -$

$1)^{k-j+1}m^{j-1} < (2t+1)m^k$. Hence, $\sum_{j \in [k+1]} n_j \leq (2t+1)(k+1)(2m)^k = n$, and that the largest value in the stream is $(m+1)b^k = O(n)$. Note that any t -approximate median equals, $\sum_{i \in [k+1]} g_i b^{k+1-i}$ and thus if P_1 returns the t -approximate median modulo b then this is g_{k+1} . This can easily be computed by a protocol in which each player transmits the memory state of the algorithm at the appropriate juncture.

References

1. T. M. Chan and E. Y. Chen. Multi-pass geometric algorithms. In *Symposium on Computational Geometry*, pages 180–189, 2005.
2. 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.
3. G. Cormode, F. Korn, S. Muthukrishnan, and D. Srivastava. Space- and time-efficient deterministic algorithms for biased quantiles over data streams. In *PODS*, pages 263–272, 2006.
4. G. Cormode and S. Muthukrishnan. An improved data stream summary: the count-min sketch and its applications. *J. Algorithms*, 55(1):58–75, 2005.
5. E. D. Demaine, A. López-Ortiz, and J. I. Munro. Frequency estimation of internet packet streams with limited space. In *ESA*, pages 348–360, 2002.
6. P. Drineas and R. Kannan. Pass efficient algorithms for approximating large matrices. In *ACM-SIAM Symposium on Discrete Algorithms*, pages 223–232, 2003.
7. 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.
8. J. Feigenbaum, S. Kannan, M. Strauss, and M. Viswanathan. Testing and spot-checking of data streams. *Algorithmica*, 34(1):67–80, 2002.
9. A. C. Gilbert, Y. Kotidis, S. Muthukrishnan, and M. Strauss. How to summarize the universe: Dynamic maintenance of quantiles. In *International Conference on Very Large Data Bases (VLDB)*, pages 454–465, 2002.
10. M. Greenwald and S. Khanna. Space-efficient online computation of quantile summaries. In *ACM SIGMOD International Conference on Management of Data*, pages 58–66, 2001.
11. S. Guha and A. McGregor. Approximate quantiles and the order of the stream. In *ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 273–279, 2006.
12. S. Guha and A. McGregor. Space-efficient sampling. In *AISTATS*, pages 169–176, 2007.
13. S. Guha, A. McGregor, and S. Venkatasubramanian. Streaming and sublinear approximation of entropy and information distances. In *ACM-SIAM Symposium on Discrete Algorithms*, pages 733–742, 2006.
14. A. Gupta and F. Zane. Counting inversions in lists. *ACM-SIAM Symposium on Discrete Algorithms*, pages 253–254, 2003.
15. S. Kannan. Open problems in streaming. *DIMACS Workshop (Slides: dimacs.rutgers.edu/Workshops/Streaming/abstracts.html)*, 2001.
16. E. Kushilevitz and N. Nisan. *Communication Complexity*. Cambridge University Press, 1997.

17. G. S. Manku, S. Rajagopalan, and B. G. Lindsay. Approximate medians and other quantiles in one pass and with limited memory. In *ACM SIGMOD International Conference on Management of Data*, pages 426–435, 1998.
18. G. S. Manku, S. Rajagopalan, and B. G. Lindsay. Random sampling techniques for space efficient online computation of order statistics of large datasets. In *ACM SIGMOD International Conference on Management of Data*, pages 251–262, 1999.
19. A. McGregor. Finding graph matchings in data streams. In *APPROX-RANDOM*, pages 170–181, 2005.
20. 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.
21. J. I. Munro and M. Paterson. Selection and sorting with limited storage. *Theor. Comput. Sci.*, 12:315–323, 1980.
22. N. Nisan and A. Wigderson. Rounds in communication complexity revisited. *SIAM J. Comput.*, 22(1):211–219, 1993.
23. N. Shrivastava, C. Buragohain, D. Agrawal, and S. Suri. Medians and beyond: new aggregation techniques for sensor networks. In *SenSys*, pages 239–249, 2004.
24. A. C. Yao. Lower bounds by probabilistic arguments. In *IEEE Symposium on Foundations of Computer Science*, pages 420–428, 1980.

A Selection Algorithm for Random-Order Streams

In this section we show how to perform approximate selection in a single pass. We will also assume that the stream contains distinct values. This can easily be achieved by attaching a secondary value $y_i \in_R [n^3]$ to each item x_i in the stream. We say $(x_i, y_i) < (x_j, y_j)$ iff $x_i < x_j$ or $(x_i = x_j \text{ and } y_i < y_j)$. Note that breaking the ties arbitrarily results in a stream whose order is not random.

Our algorithm proceeds in phases and each phase is composed of three distinct sub-phases; the *Sample* sub-phase, the *Estimate* sub-phase, and the *Update* sub-phase. At all points we maintain an open interval (a, b) such that we believe that the value of the element with rank k is between a and b . In each phase we aim to narrow the interval (a, b) . The *Sample* sub-phase finds a value $u \in (a, b)$. The *Estimate* sub-phase estimates the rank of u . The *Update* sub-phase replaces a or b by u depending on whether the rank of u is believed to be less or greater than u . The algorithm is presented in Fig. 1.

Theorem 5. *When presented with a randomly ordered stream, the Selection algorithm returns a value $u \in S$ such that $\text{RANK}_S(u) = k \pm 10 \ln^2(n) \ln(\delta^{-1}) \sqrt{k}$ with probability at least $1 - \delta$. The algorithm uses only $O(\log n)$ bits of space.*

The algorithm appears to need to know the length of the stream in advance but this assumption can be removed by making multiple “staggered” instantiations of the algorithm that correspond to guesses of the length as in [11]. Also, as with the algorithm presented in [11], the algorithm can be used as a sub-routine to create an algorithm that performs exact selection in $O(\log \log n)$ passes. Lastly, the algorithm can be generalized to deal with streams whose order is only “almost random” in the sense of being ϵ -generated random or t -bounded random [11].

Selection Algorithm:

1. Let $\Upsilon = 10 \ln^2(n) \ln(\delta^{-1}) \sqrt{k}$ and $p = 2(\lg(n/\Upsilon) + \sqrt{\ln(3/\delta) \lg(n/\Upsilon)})$
2. Let $a = -\infty$ and $b = +\infty$
3. Let $l_1 = n\Upsilon^{-1} \ln(3n^2p/\delta)$ and $l_2 = 2(n-1)\Upsilon^{-1} \sqrt{(k+\Upsilon) \ln(6np/\delta)}$.
4. Let the stream $S = S_1, E_1, \dots, S_p, E_p$ where $|S_i| = l_1$ and $|E_i| = l_2$
5. Phase i :
 - (a) *Sample* sub-phase: If $S_i \cap (a, b) = \emptyset$ return a , else let $u \in S_i \cap [a, b]$
 - (b) *Estimate* sub-phase: Compute $r = \text{RANK}_{E_i}(u)$ and $\tilde{r} = \frac{(n-1)(r-1)}{l_2} + 1$.
 - (c) *Update* sub-phase: If $\tilde{r} < k - \Upsilon/2$, $a \leftarrow u$, $\tilde{r} > k + \Upsilon/2$, $b \leftarrow u$ else return u

Fig. 1. An Algorithm for Computing Approximate Quantiles**B Proof of Theorem 3**

The proof is a generalization of a proof by Nisan and Wigderson [22]. We present the entire argument for completeness. In the proof we lower bound the $(k-1)$ -round distributional complexity, $D_{1/20}^{k-1}(g_k)$, i.e. we will consider a deterministic protocol and an input chosen from some distribution. The theorem will then follow by Yao's Lemma [24] since $D_{1/20}^{k-1}(g_k) \leq 2R_{1/10}^{k-1}(g_k)$.

Let T be the protocol tree of a deterministic k -round protocol. We consider the input distribution where each f_i is chosen uniformly and independently from F , the set of all m^m functions from $[m]$ to $[m]$. Note that this distribution over inputs gives rise to distribution over paths from the root of T to the leaves. We will assume that in round j , P_i 's message includes g_{j-1} if $i > j$ and g_j if $i \leq j$.

By induction this is possible with only $O(k^2 \log m)$ extra communication. Consequently we may assume that at each node at least $\lg m$ bits are transmitted. We will assume that protocol T requires at most $\epsilon m/2$ bits of communication where $\epsilon = 10^{-4}(k+1)^{-4}$ and derive a contradiction.

Consider a node z in the protocol tree of T corresponding to the j th round of the protocol when it is P_i 's turn to speak. Let g_{t-1} be the appended information in the last transmission. Note that g_0, g_1, \dots, g_{t-1} are specified by the messages so far.

Denote the set of functions $f_1 \times \dots \times f_k$ that are consistent with the messages already sent be $F_1^z \times \dots \times F_k^z$. Note that the probability of arriving at node z is $|F|^{-k} \prod_{1 \leq j \leq k} |F_j^z|$. Also note that, conditioned on arriving at node z , $f_1 \times \dots \times f_k$ is uniformly distributed over $F_1^z \times \dots \times F_k^z$.

Let c_z be the total communication until z is reached. We say a node z in the protocol tree is *nice* if, for $\delta = \max\{4\sqrt{\epsilon}, 400\epsilon\}$, it satisfies the following two conditions:

$$|F_j^z| \geq 2^{-2c_z} |F| \text{ for } j \in [k] \quad \text{and} \quad H(f_t^z(g_{t-1})) \geq \lg m - \delta .$$

Claim. Given the protocol reaches node z and z is nice then,

$$\Pr[\text{next node visited is nice}] \geq 1 - 4\sqrt{\epsilon} - 1/m .$$

Proof. Let w be a child of z and let $c_w = c_z + a_w$. For $l \neq i$ note that $|F_l^w| = |F_l^z|$ since P_l did not communicate at node z . Hence the probability that we reach node w given we have reached z is $\prod_{1 \leq j \leq k} |F_j^w|/|F_j^z| = |F_i^w|/|F_i^z|$. Furthermore, since z is nice,

$$\Pr[|F_i^w| < 2^{-2c_w}|F|] \leq \Pr\left[\frac{|F_i^w|}{|F_i^z|} < 2^{-2a_w}\right] \leq \sum_w 2^{-2a_w} \leq \frac{1}{m} \sum_w 2^{-a_w} \leq \frac{1}{m} .$$

where the second last inequality follows from $a_w \geq \lg m$ and the last inequality follows by Kraft's inequality (the messages sent must be prefix free.) Hence, with probability at least $1 - 1/m$, the next node in the protocol tree satisfies the first condition for being nice.

Proving the second condition is satisfied with high probability is more complicated. Consider two different cases, $i \neq t$ and $i = t$, corresponding to whether or not player i appended g_t . In the first case, since P_t did not communicate, $F_t^z = F_t^w$ and hence $H(f_t^w(g_{t-1})) = H(f_t^z(g_{t-1})) \geq \lg m - \delta$.

We now consider the second case. We need to show that $H(f_{t+1}^w(g_t)) \geq \lg m - \delta$. Note that we can express f_{t+1}^w as the following vector of random variables, $(f_{t+1}^w(1), \dots, f_{t+1}^w(m))$ where each $f_{t+1}^w(v)$ is a random variables in universe $[m]$. Note there is no reason to believe that components of this vector are independent. By the sub-additivity of entropy,

$$\sum_{v \in [m]} H(f_{t+1}^w(v)) \geq H(f_{t+1}^w) \geq \lg(2^{-2c_w}|F|) = \lg(|F|) - 2c_w \geq m \lg m - \epsilon m$$

using the fact that f_{t+1}^w is uniformly distribution over F_{t+1}^w , $|F_{t+1}^w| \geq 2^{-2c_w}|F|$ and $c_w \leq \epsilon m/2$. Hence if v were chosen uniformly at random from $[m]$,

$$\Pr[H(f_{t+1}^w(v)) \leq \log m - \delta] \leq \epsilon/\delta ,$$

by Markov's inequality. However, we are not interested in a v chosen uniformly at random but rather $v = g_t = f_t^z(g_{t-1})$. However since the entropy of $f_t^z(g_{t-1})$ is large it is almost distributed uniformly. Specifically, since $H(f_t^z(g_{t-1})) \geq \lg m - \delta$ it is possible to show that (see [22]), for our choice of δ ,

$$\Pr[H(f_{t+1}^w(g_t)) \leq \log m - \delta] \leq \frac{\epsilon}{\delta} \left(1 + \sqrt{\frac{4\delta}{\epsilon/\delta}}\right) \leq 4\sqrt{\epsilon} .$$

Hence with probability at least $1 - 4\sqrt{\epsilon}$ the next node satisfies the second condition of being nice. The claim follows by the union bound.

Note that the height of the protocol tree is $k(k-1)$ and that the root of the protocol tree is nice. Hence the probability of ending at a leaf that is not nice is at most $k(k-1)(1/m + 4\sqrt{\epsilon}) \leq 1/25$. If the final leaf node is nice then then $H(g_t)$ is at least $\lg m - \delta$ and hence the probability that g_t is guessed correctly is at most $(\delta + 1)/\lg m$ using Fano's inequality. This is less than $1/100$ for sufficiently large m and hence the total probability of P_1 guessing g_k correctly is at most $1 - 1/20$.