STREAM ORDER AND ORDER STATISTICS: QUANTILE ESTIMATION IN RANDOM-ORDER STREAMS*

SUDIPTO GUHA† AND ANDREW MCGREGOR‡

Abstract. When trying to process a data stream in small space, how important is the order in which the data arrive? Are there problems that are unsolvable when the ordering is worst case, but that can be solved (with high probability) when the order is chosen uniformly at random? If we consider the stream as if ordered by an adversary, what happens if we restrict the power of the adversary? We study these questions in the context of quantile estimation, one of the most well studied problems in the data-stream model. Our results include an $O(\text{polylog } n)$-space, $O(\log \log n)$-pass algorithm for exact selection in a randomly ordered stream of n elements. This resolves an open question of Munro and Paterson [*Theoret. Comput. Sci.*, 23 (1980), pp. 315–323]. We then demonstrate an exponential separation between the random-order and adversarial-order models: using $O(\text{polylog } n)$ space, exact selection requires $\Omega(\log n / \log \log n)$ passes in the adversarial-order model. This lower bound, in contrast to previous results, applies to fully general randomized algorithms and is established via a new bound on the communication complexity of a natural pointer-chasing style problem. We also prove the first fully general lower bounds in the random-order model: finding an element with rank $n/2 \pm n^\delta$ in the single-pass random-order model with probability at least $9/10$ requires $\Omega(\sqrt{n^{1-3\delta}/\log n})$ space.

Key words. communication complexity, stochastically generated streams, stream computation

AMS subject classifications. 68Q05, 68Q17, 68Q25, 68W20, 68W25

DOI. 10.1137/07069328X

1. Introduction. One of the principal theoretical motivations for studying the data-stream model is to understand the role played by the order in which a problem is revealed. While an algorithm in the RAM model can process the input data in an arbitrary order, the key constraint of the data-stream model is that the 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 vary along this spectrum? To what extent does it matter how the stream is ordered?

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 were randomly ordered, $\tilde{O}(n^{1/(2p)})$ space sufficed. Based on this result and other observations, it seemed plausible that any p-pass algorithm in the random-order model could be simulated by a $2p$-pass algorithm in the adversarial-order model. This was posed as an open problem by Kannan [17], and further support for this conjecture came via work initiated by Feigenbaum et al. [6] that considered the relationship

between various property testing models and the data-stream model. It was shown by Guha, McGregor, and Venkatasubramanian [14] that several models of property testing can be simulated in the single-pass random-order data-stream model, while it appeared that a similar simulation in the adversarial-order model required two passes.

In this paper we resolve the conjecture and demonstrate the important role played by the stream order in the context of exact selection and quantile estimation. Before detailing our results, we first motivate the study of the random-order model. We believe that this motivation, coupled with the array of further questions that naturally arise, may establish a fruitful area of future research.

1.1. Motivation. In the literature to date, it is usually assumed that the stream to be processed is ordered by an omnipotent adversary that knows the algorithm and the set of elements in the stream. In contrast to the large body of work on adversarially ordered streams, the random-order model has received little explicit attention to date. Aside from the aforementioned work by Munro and Paterson [21] and Guha, McGregor, and Venkatasubramanian [14], the only other results were given by Demaine, López-Ortiz, and Munro [5] in a paper about frequency estimation. However, there are numerous motivations for considering this model.

First, the random-order model gives rise to a natural notion of *average-case analysis* which explains why certain data-stream problems may have prohibitive space lower bounds while being typically solvable in practice. When evaluating a permutation invariant function f on a stream, we observe that there are two orthogonal components to an instance: the set of data items in the stream, $\mathcal{O} = \{x_1, x_2, \ldots, x_n\}$, and π, the permutation of $\{1, 2, \ldots, n\}$ that determines the ordering of the stream. Since f is permutation invariant, $\mathcal{O}$ determines the value of f. One approach when designing algorithms is to make an assumption about $\mathcal{O}$ such as that the set of items is distributed according to a Gaussian distribution. While this approach has its merits, because we are trying to compute something about $\mathcal{O}$ it is often difficult to find a suitable assumption that would allow small-space computation while not directly implying the value of the function. We take an alternative view and, rather than making assumptions about $\mathcal{O}$, we consider which problems can be solved, with high probability, when the data items are ordered randomly. This approach is an average-case analysis, where π is chosen uniformly from all possible permutations while $\mathcal{O}$ is chosen worst case.

Second, if we consider π to be determined by an adversary, a natural complexity question is the relationship between the power of the adversary and the resources required to process a stream. If we impose certain computational constraints on the adversary, a popular idea in cryptography, how does this affect the space and time required to process the stream?

Lastly, there are situations in which it is reasonable to assume the stream is not ordered adversarially. These include the following scenarios, where the stream order is random either by design, by definition, or because of the semantics of data:

1. *Random by definition:* A natural setting in which a data stream would be ordered randomly is if each element of the stream is a sample drawn independently from some unknown distribution. Regardless of the source distribution, given the set of n samples, each of the $n!$ permutations of the sequence of samples was equally likely. Density estimation algorithms that capitalized on this were presented by Guha and McGregor [13].

2. *Random by semantics:* In other situations the semantics of the data in the stream may imply that the stream is randomly ordered. For example, consider

a database of employee records in which the records are sorted by surname. We wish to estimate some property of the employee salaries given a stream of ⟨surname, salary⟩ tuples. If there is no correlation between the lexicographic ordering of the surnames and the numerical ordering of salaries, then the salary values are ordered uniformly at random. We note that several query optimizers make such assumptions.

3. *Random by design:* Lastly, there are some scenarios in which we dictate the order of the stream. Naturally we can therefore ensure it is nonadversarial! An example is the "backing sample" architecture proposed by Gibbons and coworkers [7,8] for maintaining accurate estimates of aggregate properties of a database. A large sample is stored on the disk and this sample is used to periodically correct estimates of the relevant properties.

1.2. Our contributions. We start with the following algorithmic results which are proved in section 3:

1. A single-pass algorithm using $O(\log n)$ space that, given any k, returns an element of rank $k \pm O(k^{1/2} \log^2 n \log \delta^{-1})$ with probability at least $1 - \delta$ if the stream is randomly ordered. The algorithm does not require prior knowledge of the length of the stream.

2. An algorithm using $O(\text{polylog } n)$ space that performs exact selection in only $O(\log \log n)$ passes. This was conjectured by Munro and Paterson [21] but has been unresolved for over 30 years.

In section 4, we introduce two notions of the order of the stream being "semi-random." The first is related to the computational power of an adversary ordering the stream, and the second is related to the random process that determines the order. We show how the performance of our algorithms degrades as the randomness of the order decreases according to either notion. These notions of semirandomness will also be critical for proving lower bounds in the random-order model. In sections 5 and 6, we prove the following lower bounds:

1. Any algorithm that returns an n^δ-approximate median, i.e., an element with rank $n/2 \pm n^\delta$, in the single-pass random-order model with probability at least $9/10$ requires $\Omega(\sqrt{n^{1-3\delta}/\log n})$ space. This is the first unqualified lower bound in this model. Previously, all that was known was that a single-pass algorithm that maintained a set of elements whose ranks (among the elements read thus far) are consecutive and as close to the current median as possible, required $\Omega(\sqrt{n})$ space to find the exact median in the random-order model [21]. Our result, which is fully general, 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. We believe the techniques used will be useful in proving average-case lower bounds for a variety of data-stream problems.

2. Any algorithm that returns an n^δ-approximate median in p passes of an adversarially ordered stream requires $\Omega(n^{(1-\delta)/p} p^{-6})$ space. In particular, this implies that in the adversarial-order model any $O(\text{polylog } n)$-space algorithm for exact selection must use $\Omega(\log n / \log \log n)$ passes. This is established via a new bound on the communication complexity of a natural pointer-chasing style problem. The best previous result showed that any deterministic, comparison-based algorithm for exact selection required $\Omega(n^{1/p})$ space for constant p [21]. This resolves the conjecture of Kannan and establishes that existing multipass algorithms are optimal up to terms polynomial in p.

1.3. Related work on quantile estimation. Quantile estimation is perhaps the most extensively studied problem in the data-stream model [3,4,9,10,15,19,20,23]. Manku, Rajagopalan, and Lindsay [19,20] showed that we can find an element of rank $k \pm \epsilon n$ using $O(\epsilon^{-1} \log^2 \epsilon n)$ space, where n is the length of the stream and k is user specified. This was improved to a deterministic, $O(\epsilon^{-1} \log \epsilon n)$-space algorithm by Greenwald and Khanna [10]. Gilbert et al. [9] gave an algorithm for the model in which elements may also be "deleted" from the stream. Shrivastava et al. [23] presented another deterministic algorithm for insert-only streams that uses $O(\epsilon^{-1} \log U)$ space, where U is the size of the domain from which the input is drawn. Gupta and Zane [15] and Cormode et al. [3] presented algorithms for estimating *biased quantiles*, i.e., algorithms that return an element of rank $k \pm \epsilon k$ for any $k \in \{1, \ldots, n\}$. We note that all these algorithms are for the adversarial-order model and therefore are not designed to take advantage of a weak, or absent, adversary.

1.4. Recent developments. Since the initial submission of this paper, there has been follow-up work that presents lower bounds on the space required for multi-pass algorithms for randomly ordered data streams [1,2]. In particular, it was shown that any $O(\text{polylog}\, n)$-space algorithm that returns the median of a randomly ordered stream of length n with probability at least $9/10$ requires $\Omega(\log \log n)$ passes. Other problems were also considered in the random-order model, including estimating frequency moments, graph connectivity, and measuring information divergences [1].

2. Notation and preliminaries. Let $[n] = \{1, \ldots, n\}$. Let Sym_n be the set of all $n!$ permutations of $[n]$. We say $a = b \pm c$ if $|a - b| \le c$ and write $a \in_R S$ to indicate that a is chosen, uniformly at random, from the set S. The next definition clarifies the rank of an element in a multiset.

DEFINITION 2.1 (rank and approximate selection). *The rank of an item x in a set S is defined as*

$$\text{RANK}_S(x) = |\{x' \in S | x' < x\}| + 1 \ .$$

Assuming there are no duplicate elements in S, we say x is an Υ-approximate k-rank element if $\text{RANK}_S(x) = k \pm \Upsilon$. If there are duplicate elements in S, we say x is an Υ-approximate k-rank element if there exists some way of breaking ties such that x is an Υ-approximate k-rank element.

At various points we will appeal to less common variants of Chernoff–Hoeffding bounds that pertain to sampling without replacement.

THEOREM 2.2 (Hoeffding [16]). *Consider a population C consisting of N values $\{c_1, \ldots, c_N\}$. Let the mean value of the population be $\mu = N^{-1} \sum_{i=1}^{N} c_i$ and let $c_{\max} = \max_{i \in N} c_i - \min_{i \in N} c_i$. Let $X_1, \ldots, X_n$ be a sequence of independent samples* without *replacement from C and $\bar{X} = n^{-1} \sum_{i=1}^{n} X_i$. Then,*

$$\Pr\left[\bar{X} \notin (\mu - a, \mu + b)\right] \le \exp\left(-2na^2/c_{\max}^2\right) + \exp\left(-2nb^2/c_{\max}^2\right) \ .$$

The following corollary will also be useful. If $c_i = 1$ for $i \in [k]$ and 0 otherwise, then

$$\Pr\left[\bar{X} \notin (\mu - a, \mu + b)\right] \le \exp\left(-2a^2n^2/k\right) + \exp\left(-2b^2n^2/k\right) \ .$$

3. Algorithms for random-order streams. In this section we show how to perform approximate selection of the kth smallest element in a single pass over a

Selection Algorithm:
1. Let $a = -\infty, b = +\infty$, and

$$\Upsilon = 20 \ln^2(n) \ln(\delta^{-1}) \sqrt{k}$$
$$p = 4(\log_{4/3}(n/\Upsilon) + (\ln(3/\delta) \log_{4/3}(n/\Upsilon))^{1/2})$$
$$l_1 = n\Upsilon^{-1} \ln(3n^2 p/\delta)$$
$$l_2 = 2(n-1)\Upsilon^{-1} \sqrt{(k+\Upsilon) \ln(6np/\delta)}$$

2. Partition stream as $S = \langle S_1, E_1, \ldots, S_p, E_p \rangle$ where $|S_i| = l_1$ and $|E_i| = l_2$
3. Phase $i \in [p]$:
   (a) *Sample*: If $S_i \cap (a,b) = \emptyset$, return a, else let u be first element in $S_i \cap (a,b)$
   (b) *Estimate*: Compute $r = \text{RANK}_{E_i}(u)$ and let $\tilde{r} = (n-1)(r-1)/l_2 + 1$
   (c) *Update*: If $\tilde{r} < k - \Upsilon/2$, $a \leftarrow u$, $\tilde{r} > k + \Upsilon/2$, $b \leftarrow u$, else return u

FIG. 3.1. *The selection algorithm.*

randomly ordered stream of length n. As we are interested in massive data streams, we consider the space complexity and accuracy guarantees of the algorithm as n becomes large.

DEFINITION 3.1 (random order). *Consider a set of elements* $x_1, \ldots, x_n \in [\text{poly}(n)]$. *Then this set and* $\pi \in \text{Sym}_n$ *define a stream* $S = \langle x_{\pi(1)}, \ldots, x_{\pi(n)} \rangle$. *If* π *is chosen uniformly from* Sym_n, *then we say the stream is in* random order.

We will present the algorithm, assuming the exact value of the length of the stream, n, is known in advance. In a subsequent section, we will show that this assumption is not necessary. In what follows, we will assume that the stream contains distinct values. This can easily be achieved with probability at least $1 - \delta$ by attaching a secondary value $y_i \in_R [n^2 \delta^{-1}]$ to each item x_i in the stream. We say $(x_i, y_i) < (x_j, y_j)$ iff $x_i < x_j$ or $(x_i = x_j$ and $y_i < y_j)$. Note that breaking the ties arbitrarily results in a stream whose order is not random. We also may assume that $k \leq n/2$ by symmetry.

3.0.1. Algorithm overview. Our algorithm proceeds in phases and each phase is composed of the following three distinct subphases: the *sample* subphase, the *estimate* subphase, and the *update* subphase. 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 subphase finds a value $u \in (a,b)$. The estimate subphase estimates the rank of u. The update subphase replaces a or b by u depending on whether the rank of u is believed to be less than or greater than k. See Figure 3.1 for the algorithm.

3.0.2. Analysis. For the analysis we define the following quantity:

$$\Gamma(a,b) = |S \cap (a,b)| = |\{v \in S : a < v < b\}| .$$

LEMMA 3.2. *With probability* $1 - \delta/3$, *for all phases, if* $\Gamma(a,b) \geq \Upsilon$, *then there exists an element* u *in each sample subphase, i.e.,*

$$\Pr[\forall \ i \in [p] \ and \ a, b \in S \ such \ that \ \Gamma(a,b) \geq \Upsilon; S_i \cap (a,b) \neq \emptyset] \geq 1 - \delta/3 .$$

Proof. Fix $i \in [p]$ and $a, b \in S$ such that $\Gamma(a, b) \geq \Upsilon$. Then,

$$\Pr[S_i \cap (a, b) \neq \emptyset] \geq 1 - \left(1 - \frac{\Gamma(a, b)}{n}\right)^{l_1} \geq 1 - \exp\left(\frac{-\Upsilon l_1}{n}\right) = 1 - \frac{\delta}{3n^2 p} .$$

The result follows by applying the union bound over all choices of i, a, and b. $\square$

LEMMA 3.3. *With probability $1 - \delta/3$, for all phases, we determine the rank of u with sufficient accuracy, i.e.,*

$$\Pr\left[\forall i \in [p], \ u \in S; \ \begin{array}{ll} \tilde{r} = \mathrm{RANK}_S(u) \pm \Upsilon/2 & \textit{if } \mathrm{RANK}_S(u) < k + \Upsilon + 1 \\ \tilde{r} > k + \Upsilon/2 & \textit{if } \mathrm{RANK}_S(u) \geq k + \Upsilon + 1 \end{array}\right] \geq 1 - \delta/3 ,$$

where $\tilde{r} = (n - 1)(\mathrm{RANK}_{E_i}(u) - 1)/l_2 + 1$.

Proof. Fix $i \in [p]$ and $u \in S$. First, we consider u such that $\mathrm{RANK}_S(u) < k + \Upsilon + 1$. Let $X = \mathrm{RANK}_{E_i}(u) - 1$ and note that $E[X] = l_2(\mathrm{RANK}_S(u) - 1)/(n - 1)$. Appealing to the second part of Theorem 2.2,

$$\begin{aligned} \Pr[\tilde{r} \neq \mathrm{RANK}_S(u) \pm \Upsilon/2] &= \ \Pr\left[|X - E[X]| \geq \frac{l_2 \Upsilon}{2(n - 1)}\right] \\ &\leq \ 2\exp\left(\frac{-2(l_2\Upsilon/(2(n - 1)))^2}{\mathrm{RANK}_S(u) - 1}\right) \\ &\leq \ \frac{\delta}{3np} , \end{aligned}$$

where the last inequality follows because $(l_2\Upsilon/(2(n - 1)))^2 = (k + \Upsilon)\ln(6np/\delta)$ (by definition of l_2) and $\mathrm{RANK}_S(u) - 1 < k + \Upsilon$ (by assumption). Now assume that $\mathrm{RANK}_S(u) \geq k + \Upsilon + 1$ and note that $\Pr[\tilde{r} \geq k + \Upsilon/2]$ is minimized for $\mathrm{RANK}_S(u) = k + \Upsilon + 1$. Hence,

$$\begin{aligned} \Pr[\tilde{r} > k + \Upsilon/2] &= \ 1 - \Pr\left[E[X] - X \geq \frac{l_2 \Upsilon}{2(n - 1)}\right] \\ &\geq \ 1 - \exp\left(-\frac{(l_2\Upsilon)^2}{4(k + \Upsilon)(n - 1)^2}\right) \\ &= \ 1 - \frac{\delta}{6np} . \end{aligned}$$

The result follows by applying the union bound over all choices of i and u. $\square$

We now give the main theorem of this section.

THEOREM 3.4. *For $k \in [n]$, there exists a single-pass, $O(\log n)$-space algorithm in the random-order model that returns u such that $\mathrm{RANK}_S(u) = k \pm 20\ln^2(n)\ln(\delta^{-1})\sqrt{k}$ with probability at least $1 - \delta$.*

Proof. Consider $\Gamma(a, b) = |\{v \in S : a < v < b\}|$ in each phase of the algorithm. By Lemmas 3.2 and 3.3, with probability at least $1 - 2\delta/3$, in every phase, if we do not terminate, then $\Gamma(a, b)$ decreases and $\mathrm{RANK}_S(a) \leq k \leq \mathrm{RANK}_S(b)$. In particular, in each phase, with probability $1/4$, either we terminate or $\Gamma(a, b)$ decreases by at least a factor of $3/4$. Let Y be the number of phases in which $\Gamma(a, b)$ decreases by a factor of $3/4$. If the algorithm does not terminate, then $Y < \log_{4/3}(n/\Upsilon)$ since $\Gamma(a, b)$ is initially n and the algorithm will terminate if $\Gamma(a, b) < \Upsilon$. But,

$$\Pr\left[Y < \log_{4/3}(n/\Upsilon)\right] = \Pr\left[Y < E[Y] - \sqrt{\ln(3/\delta)\log_{4/3}(n/\Upsilon)}\right] \leq \delta/3 .$$

Hence with probability at least $1 - \delta$ the algorithm returns a value with rank $k \pm \Upsilon$.

The space bound follows immediately from the fact that the algorithm only stores a constant number of polynomially sized values and maintains a counter that stores values in the range $[n]$. Finally, for sufficiently large n,

$$p(l_1 + l_2) \leq 20 \ln^2(n) \ln(\delta^{-1}) n \Upsilon^{-1} \sqrt{k} = n,$$

and hence the stream is sufficiently long for all the phases to complete. $\square$

3.1. Generalizing to unknown stream lengths. The algorithm in the previous section assumed prior knowledge of n, the length of the stream. We now discuss a simple way to remove this assumption. First we argue that, for our purposes, it is sufficient to look at only half the stream.

LEMMA 3.5. *Given a randomly ordered stream S of length n, let S' be a contiguous substream of length $\tilde{n} \geq n/2$. Then, with probability at least $1 - \delta$, if u is the $\tilde{k}$th smallest element of S', then $\mathrm{RANK}_S(u) = \tilde{k}n/\tilde{n} \pm 2(8\tilde{k} \ln \delta^{-1})^{0.5}$.*

Proof. Let $a = \tilde{k}/\tilde{n}$. Let the elements in the stream be $x_1 \leq \cdots \leq x_n$. Let $X = |\{x_1, \ldots, x_{an+b}\} \cap S'|$ and $Y = |\{x_1, \ldots, x_{an-b-1}\} \cap S'|$, where $b = 2(8\tilde{k} \ln \delta^{-1})^{0.5}$. The probability that the element of rank $\tilde{k} = a\tilde{n}$ in S' has rank in S outside the range $[an - b, an + b]$ is less than

$$\Pr[X < a\tilde{n} \text{ or } Y > a\tilde{n}] \leq \Pr[X < E[X] - b/2 \text{ or } Y > E[Y] + b/2]$$

$$\leq 2\exp\left(\frac{-(b/2)^2}{3(a\tilde{n} + b)}\right) \leq \delta \ .$$

The lemma follows. $\square$

To remove the assumption that we know n, we make multiple instantiations of the algorithm. Each instantiation corresponds to a guess of n. Let $\beta = 1.5$. Instantiation i guesses a length of $\lceil 4\beta^i \rceil - \lfloor \beta^i \rfloor + 1$ and is run on the stream starting with the $\lfloor \beta^i \rfloor$th data item and ending with the $\lceil 4\beta^i \rceil$th data item. We remember the result of the algorithm until the $2(\lceil 4\beta^i \rceil - \lfloor \beta^i \rfloor + 1)$th element arrives. We say the instantiation has been canceled at this point.

LEMMA 3.6. *At any time, there is only a constant number of instantiations. Furthermore, when the stream terminates, at least one instantiation has run on a substream of at least $n/2$.*

Proof. Consider the tth element of the data stream. By this point there have been $O(\log_\beta t)$ instantiations made. However, $\Omega(\log_\beta t/6)$ instantiations have been canceled. Hence $O(\log_\beta t - \log_\beta t/6) = O(1)$ instantiations are running. We now show that there always exists an instantiation that has been running on at least half the stream. The ith instantiation gives a useful result if the length of the stream $n \in U_i = \{\lfloor 4\beta^i \rfloor + 1, \ldots, 2(\lceil 4\beta^i \rceil - \lfloor \beta^i \rfloor + 1)\}$. But $\bigcup_{i \geq 0} U_i = \mathbb{N} \setminus \{0, 1, 2, 3, 4\}$ since for all $i > 1$, $\lfloor 4\beta^i + 1 \rfloor \leq 2(\lceil 4\beta^{i-1} \rceil - \lfloor \beta^{i-1} \rfloor + 1)$. $\square$

We can therefore generalize Theorem 3.4 as follows.

THEOREM 3.7. *For $k \in [n]$, there exists a single-pass, $O(\log n)$-space algorithm in the random-order model that returns u such that $\mathrm{RANK}_S(u) = k \pm 11 \ln^2(n) \ln(\delta^{-1}) \sqrt{k}$ with probability at least $1 - \delta$. The algorithm need not know n in advance.*

3.2. Multipass exact selection. In this section we consider the problem of exact selection of an element of rank $k = \Omega(n)$. We will later show that this requires $\Omega(\sqrt{n})$ space if an algorithm is permitted only one pass over a stream in random order. However, if $O(\log \log n)$ passes are permitted, we now show that $O(\text{polylog } n)$

space is sufficient. We will again assume that the elements in the stream are distinct but we note that it is not difficult to avoid this assumption.

We use a slight variant of the single-pass algorithm in section 3 as a building block. Rather than returning a single candidate, we output the pair a and b. Using the analysis in section 3, it can be shown that, with probability $1 - \delta$, $\text{RANK}_S(a) < k < \text{RANK}_S(b)$ and that

$$|\text{RANK}_S(a) - \text{RANK}_S(b)| \leq O(\sqrt{n} \log^2 n \log \delta^{-1}) \ .$$

In one additional pass, $\text{RANK}_S(a)$ and $\text{RANK}_S(b)$ can be computed exactly. Hence, after two passes, by ignoring all elements outside the range (a, b), we have reduced the problem to that of finding an element of rank $k - \text{RANK}_S(a)$ in a stream of length $O(\sqrt{n} \log^3 n)$ if we assume that $\delta^{-1} = \text{poly}(n)$. If we repeat this process $O(\log \log n)$ times and then select the desired element by explicitly storing the remaining $O(\text{polylog}\, n)$-length stream, it would appear that we can perform exact selection in $O(\text{polylog}\, n)$ space and $O(\log \log n)$ passes. However, there is one crucial detail that needs to be addressed.

In the first pass, by assumption we are processing a data stream whose order is chosen uniformly from Sym_n. However, because the stream order is not rerandomized between each pass, it is possible that the previous analysis does not apply because of dependencies that may arise between different passes. Fortunately, the following straightforward, but necessary, observation demonstrates that this is not the case.

FACT 3.8. *Let a and b, respectively, be the lower and upper bound returned after a pass of the algorithm on the stream $\langle x_1, \ldots, x_n \rangle$. Let $\pi \in \text{Sym}_n$ satisfy $i = \pi(i)$ for all $i \in [n]$ such that $x_i \notin (a, b)$. Then the algorithm also would return the same bounds after processing the stream $\langle x_{\pi(1)}, \ldots, x_{\pi(n)} \rangle$.*

Therefore, conditioned on the algorithm returning a and b, the substream of elements in the range (a, b) are still ordered uniformly. This leads to the following theorem.

THEOREM 3.9. *For $k \in [n]$, there exists an $O(\text{polylog}\, n)$-space, $O(\log \log n)$-pass algorithm in the random-order model that returns the kth smallest value of a stream with probability $1 - 1/\text{poly}(n)$.*

3.3. Applications to equidepth histograms. In this section we briefly overview an application to constructing B-bucket equidepth histograms. Here, the histogram is defined by B buckets whose boundaries are defined by the items of rank $in/(B+1)$ for $i \in [B]$. Gibbons, Matias, and Poosala [8] consider the problem of constructing an approximate B-bucket equidepth histogram of data stored in a backing sample. The measure of "goodness of fit" they consider is

$$\mu = n^{-1} \sqrt{B^{-1} \sum_{i \in [B]} \epsilon_i^2} \ ,$$

where ϵ_i is the error in the rank of the boundary of the ith bucket. They show that μ can be made smaller than any $\epsilon > 0$ where the space used depends on ϵ. However, in their model it is possible to ensure that the data are stored in random order. As a consequence of the algorithm in section 3, we get the following theorem.

COROLLARY 3.10. *In a single pass over a backing sample of size n stored in random order, we can compute the B quantiles of the samples using $O(B \log n)$ memory with error $\tilde{O}(n^{-1/2})$. Since the error goes to zero as the sample size increases, we have the first consistent estimator for this problem.*

4. Semirandom order. In this section we consider two natural notions of "semirandom" ordering and explain how our algorithm can be adjusted to process streams whose order is semirandom under either definition. The first notion is stochastic in nature: we consider the distribution over orders which are "close" to the uniform order in terms of the variational distance. This will play a critical role when proving lower bounds.

DEFINITION 4.1 (ϵ-generated-random order). *Given set $\{x_1, \ldots, x_n\}$, $\pi \in \mathrm{Sym}_n$ defines a stream $\langle x_{\pi(1)}, \ldots, x_{\pi(n)} \rangle$. We say the order is ϵ-generated random (ϵ-GR) if π is chosen according to a distribution ν such that $\|\mu - \nu\|_1 \leq \epsilon$, where μ is the uniform distribution on Sym_n.*

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

LEMMA 4.2. *Let $\mathcal{A}$ be a randomized algorithm that succeeds (i.e., returns an estimate of some property with some accuracy guarantee) with probability at least $1 - \delta$ in the random-order model. Then $\mathcal{A}$ succeeds with probability at least $1 - \delta - \epsilon$ when the stream order is ϵ-GR.*

Proof. Let $\Pr_{\mu,\mathrm{coin}}[\cdot]$ 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 the uniform distribution μ. Similarly, define $\Pr_{\nu,\mathrm{coin}}[\cdot]$, where ν is any distribution satisfying $\|\mu - \nu\|_1 \leq \epsilon$:

$$\Pr_{\mu,\mathrm{coin}}[\mathcal{A} \text{ succeeds}] = \sum_{\pi \in \mathrm{Sym}_n} \Pr_{\mu}[\pi] \Pr_{\mathrm{coin}}[\mathcal{A} \text{ succeeds}|\pi] \leq \Pr_{\nu,\mathrm{coin}}[\mathcal{A} \text{ succeeds}] + \epsilon \ .$$

The lemma follows since $\Pr_{\mu,\mathrm{coin}}[\mathcal{A} \text{ succeeds}] \geq 1 - \delta$ by assumption.   ☐

The next theorem follows immediately from Theorem 3.4 and Lemma 4.2.

THEOREM 4.3. *For $k \in [n]$, there exists a single-pass, $O(\log n)$-space algorithm in the ϵ-GR-order model that returns u such that $\mathrm{RANK}_S(u) = k \pm 11 \ln^2(n) \ln(\delta^{-1}) \sqrt{k}$ with probability at least $1 - \delta - \epsilon$.*

The second definition is computational in nature. We consider an adversary upstream of our algorithm that can reorder the elements subject to having limited memory to do this reordering.

DEFINITION 4.4 (t-bounded-adversary-random order). *A t-bounded adversary is an adversary that can only delay at most t elements at a time; i.e., when presented with a stream $\langle x_1, \ldots, x_n \rangle$, it can ensure that the received stream is $\langle x_{\pi(1)}, \ldots, x_{\pi(n)} \rangle$ if $\pi \in \mathrm{Sym}_n$ satisfies*

$$(4.1) \qquad \forall i \in [n], |\{i < j \leq n : \pi(j) < \pi(i)\}| \leq t \ .$$

The order of a stream is t-bounded-adversary-random (t-BAR) if it is generated by a t-bounded adversary acting on a stream whose order is random.

For example, a 2-bounded adversary acting on the stream $\langle 1, 2, 3, 4, 5, 6, 7, 8, 9 \rangle$ can transform it into $\langle 3, 2, 1, 6, 5, 4, 9, 8, 7 \rangle$ or $\langle 3, 4, 5, 6, 7, 8, 9, 1, 2 \rangle$ but can not generate $\langle 9, 8, 7, 6, 5, 4, 3, 2, 1 \rangle$. In particular, in the adversarial-order model the stream order is $(n-1)$-BAR, while in the random-order model the order is 0-BAR.

LEMMA 4.5. *Consider streams $\langle x_1, \ldots, x_n \rangle$ and $\langle x_{\pi(1)}, \ldots, x_{\pi(n)} \rangle$, where π satisfies (4.1). Then for any $j, w \in [n]$, $|\{x_j, \ldots, x_{j+w-1}\} \cap \{x_{\pi(j)}, \ldots, x_{\pi(j+w-1)}\}| \geq w - 2t$.*

We assume that $t \leq \sqrt{k}$. Given the above lemma, it is straightforward to transform the algorithm of the previous section into one that is correct (with prescribed probability) when processing a stream in t-BAR order. In particular, it is sufficient

to set $l_1 = O(n\Upsilon^{-1}\ln(3n^2p/\delta) + t\delta^{-1})$ and to choose a random u among $S_i \cap (a, b)$ in each sample phase. Note that $l_1 < l_2$ for $t \leq \sqrt{k}$. In each estimate phase a t-bounded adversary can introduce an extra $2nt/l_2 \leq t\Upsilon/\sqrt{k} \leq \Upsilon$ error. Hence, the total error is at most 2Υ.

THEOREM 4.6. *For $k \in [n]$, there exists a single-pass, $O(\log n)$-space algorithm in the t-BAR-order model that returns u such that* $\text{RANK}_S(u) = k \pm 20 \ln^2(n) \ln(\delta^{-1})\sqrt{k}$ *with probability at least $1 - \delta$.*

5. Random-order lower bound. In this section we will prove a lower bound on the space required to n^δ approximate the median in the single-pass, random-order model. Our lower bound will be based on a reduction from the communication complexity of indexing [18]. However, the reduction is significantly more involved than typical reductions because different segments of a stream cannot be determined independently by different players if the stream is in random order.

Consider two players Alice and Bob, where Alice has a binary string σ of length s and Bob has an index $r \in [s]$, where s will be determined later. It is known that for Bob to determine $\text{INDEX}(\sigma, r) = \sigma_r$ after a single message from Alice with probability at least $4/5$, this message must consist of $\Omega(s)$ bits.

THEOREM 5.1 (see, e.g., [18]). $R_{1/5}^{1-\text{way}}(\text{INDEX}) \geq c^*s$ *for some constant $c^* > 0$.*

We start by assuming that there exists an algorithm $\mathcal{A}$ that computes an n^δ-approximate median in the single-pass, random-order model with probability at least $9/10$. We then use this to construct a one-way communication protocol that will allow Alice and Bob to solve their INDEX problem. They do this by simulating $\mathcal{A}$ on a stream of length n, where Alice determines a long prefix of the stream and Bob determines the remaining elements. For convenience we assume n is even and consider the median to be the element of rank $n/2$. The stream they construct consists of the union of the following sets of elements:

  X: A size x set consisting of $n/2 + n^\delta - (2n^\delta + 1)r$ copies of 0.
  Y: A size y set consisting of $n/2 - n^\delta - (2n^\delta + 1)(s - r)$ copies of $2s + 2$.
  Z: A size $z = (2n^\delta + 1)s$ set consisting of $2n^\delta + 1$ copies of $\{2i + \sigma_i : i \in [s]\}$.

Note that any n^δ-approximate median of $U = S \cup X \cup Y$ is $2r + \sigma_r$. The difficulty we face is that we may only assume $\mathcal{A}$ returns an n^δ-approximate median of U if U is ordered randomly. Ensuring this seems to require a significant amount of communication between Alice and Bob. How else can Alice determine the balance of elements from X and Y in the prefix of the stream or can Bob know the elements of Z that should appear in the suffix of the stream?

In what follows we will argue that by carefully choosing the length of the prefix, suffix, and s, it is possible for Alice and Bob to ensure that the ordering of the stream is $1/20$-GR, while only communicating a sufficiently small number of bits with probability at least $19/20$. Then, by appealing to Lemma 4.2, we may assume that the protocol is correct with probability at least $4/5$.

5.1. Generating a stream in semirandom order. Let A be the set of elements in the prefix of the stream which is determined by Alice. Let $B = U \setminus A$ be the set of elements in the remaining part of the stream which is determined by Bob. Roughly speaking, A will consist of $n - \tilde{\Theta}(n^{1-\delta})$ of the stream elements, including most of the elements from Z. The number of elements from X and Y will be determined on the assumption that $x = y$. B will consist of the remaining $\tilde{\Theta}(n^{1-\delta})$ elements from $X \cup Y \cup Z$. The intuition is that if B is too large, then B will contain too many elements from Z whereas, if B is too small, the assumption that $x = y$ in

the determination of A will be problematic when it comes to arguing that the order of the stream is nearly random.

Let $p = c^*/(8n^\delta \log n)$ and consider the following protocol:

1. Alice determines $A \cap Z$ and $B \cap Z$ by placing an element from Z into B with probability p, and placing it in A otherwise. Alice picks t_0 according to $T_0 \sim \text{Bin}(n/2-z, 1-p)$ and t_1 according to $T_1 \sim \text{Bin}(n/2-z, 1-p)$. She places t_0 copies of 0 and t_1 copies of $2s+2$ into A. She sends a message encoding $B \cap Z$, t_0, t_1, and the memory state of $\mathcal{A}$ run on a random permutation of A.

2. Bob instantiates $\mathcal{A}$ with memory state sent by Alice and continues running it on a random permutation of $B = (B \cap Z) \cup \{x - t_0 \text{ copies of } 0\} \cup \{y - t_1 \text{ copies of } 2s+2\}$. Finally, Bob returns 1 if the output of the algorithm is odd, and 0 otherwise.

Let ν be the distribution over stream orders generated by the above protocol. The next lemma establishes that ν is almost uniform. This will be required to prove the correctness of the algorithm.

LEMMA 5.2. *If $z = 10^{-6}\sqrt{pn}$, then $\|\mu - \nu\|_1 \le 1/20$ where μ is the uniform distribution on Sym_n.*

Proof. Define the random variables $T_0' \sim \text{Bin}(x, 1-p)$ and $T_1' \sim \text{Bin}(y, 1-p)$ and let $a_0 = x - n/2 + z$ and $a_1 = y - n/2 + z$. Note that $a_0, a_1 \ge 0$ and $a_0 + a_1 = z$. We upper bound $\|\mu - \nu\|_1$ as follows:

$$\|\mu - \nu\|_1 = \sum_{t_0, t_1} |\Pr[T_0 = t_0, T_1 = t_1] - \Pr[T_0' = t_0, T_1' = t_1]|$$

$$\le \max_{\substack{t_0 \in (1-p)x \pm b^* \\ t_1 \in (1-p)y \pm b^*}} \left| \frac{\Pr[T_0 = t_0, T_1 = t_1]}{\Pr[T_0' = t_0, T_1' = t_1]} - 1 \right|$$

$$+ \Pr[\max\{|T_0 - E[T_0]|, |T_1 - E[T_1]|\} \ge b^* - pz]$$
$$+ \Pr[\max\{|T_0' - E[T_0']|, |T_1' - E[T_1']|\} \ge b^*],$$

where $b^* = 10\sqrt{pn/2} + pz$. By the Chernoff bound,

$$\Pr[\max\{|T_0 - E[T_0]|, |T_1 - E[T_1]|\} \ge b^* - pz]$$
$$+ \Pr[\max\{|T_0' - E[T_0']|, |T_1' - E[T_1']|\} \ge b^*] \le 8\exp\left(-2(b^* - pz)^2/(3pn)\right),$$

and hence the (sum of the) last two terms are upper bounded by $1/40$ for sufficiently large n.

Let $t_0 = (1-p)x + b_0$ and $t_1 = (1-p)x + b_1$ and assume that $|b_0|, |b_1| \le b^*$. Then,

$$\frac{\Pr[T_0 = t_0, T_1 = t_1]}{\Pr[T_0' = t_0, T_1' = t_1]} = \frac{\binom{n/2-z}{t_0}\binom{n/2-z}{t_1}}{\binom{x}{t_0}\binom{y}{t_1}p^z}$$

$$= \left(\prod_{i \in [a_0]} \frac{xp - i + 1 - b_0}{(x - i + 1)p}\right)\left(\prod_{i \in [a_1]} \frac{yp - i + 1 - b_1}{(y - i + 1)p}\right),$$

and therefore

$$\exp\left(\frac{-zb^*}{p(x-z)} + \frac{-zb^*}{p(y-z)}\right) \le \frac{\Pr[T_0 = t_0, T_1 = t_1]}{\Pr[T_0' = t_0, T_1' = t_1]} \le \exp\left(\frac{2z^2 + zb^*}{p(x-z)} + \frac{2z^2 + zb^*}{p(y-z)}\right).$$

Substituting z establishes that $|\Pr[T_0 = t_0, T_1 = t_1]/\Pr[T_0' = t_0, T_1' = t_1] - 1| \le 1/40$ for sufficiently large n. The lemma follows. $\square$

The next lemma will be necessary to bound the communication of the protocol.

LEMMA 5.3. $\Pr[|Z \cap B| \geq c^*s/(2\log n)] \leq 1/20$ *for* $s = \omega(\log n)$.

Proof. Note that $E[|Z \cap B|] = pz \leq 3c^*s/(8\log n)$. Then, by an application of the Chernoff bound,

$$\Pr[|Z \cap B| \geq c^*s/(2\log n)] = \Pr[|Z \cap B| \geq (4/3)E[|Z \cap B|]] \leq \exp(-c^*s/(72\log n)) \ . \quad \square$$

THEOREM 5.4. *Computing an n^δ-approximate median in the random-order model with probability at least $9/10$ requires $\Omega(\sqrt{n^{1-3\delta}/\log n})$ space.*

Proof. Let Alice and Bob follow the above protocol to solve their instance of INDEX using $\mathcal{A}$. Assume $\mathcal{A}$ uses M bits of space. By Lemmas 4.2 and 5.2, the protocol is correct with probability at least $9/10 - 1/20 = 17/20$. Furthermore, by Lemma 5.3, with probability at least $19/20$ the protocol requires at most $3c^*s/4 + M$ bits of communication (for sufficiently large n): $c^*s/2$ bits to transmit $Z \cap B$, $2\log n$ bits to transmit t_0 and t_1, and M bits for the memory state of $\mathcal{A}$. Therefore, there exists a protocol transmitting $3c^*s/4 + M$ bits that is correct with probability at least $17/20 - 1/20 = 4/5$. Hence, by Theorem 5.1, $M = \Omega(s) = \Omega(\sqrt{n^{1-3\delta}/\log n})$. $\quad \square$

6. Adversarial-order lower bound. In this section we prove that any p-pass algorithm that returns an n^δ-approximate median in the adversarial-order model requires $\Omega(n^{(1-\delta)/p}p^{-6})$ space. This, coupled with the upper bound of Munro and Paterson [21], will resolve the space complexity of multipass algorithms for median finding up to polylogarithmic factors. The proof will use a reduction from the communication complexity of a generalized form of pointer chasing that we now describe.

DEFINITION 6.1 (generalized pointer chasing). *For $i \in [p]$, let $f_i : [m] \to [m]$ be an arbitrary function. Then g_p is defined by*

$$g_p(f_1, f_2, \ldots, f_p) = f_p(f_{p-1}(\ldots(f_1(1))\ldots)) \ .$$

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

Note that $R_0^p(g_p) = O(p\log m)$. We will be looking at $(p-1)$-round protocols. The proof of the next result will be deferred to the next section.

THEOREM 6.2. $R_{1/10}^{p-1}(g_p) = \Omega(m/p^4 - p^2\log m)$.

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

THEOREM 6.3. *Finding an n^δ-approximate median in p passes with probability at least $9/10$ in the adversarial-order model requires $\Omega(n^{(1-\delta)/p}p^{-6})$ space.*

Proof. We will show how a p-pass algorithm $\mathcal{A}$ that computes a t-approximate median of a length n stream gives rise to a p-round protocol for computing g_{p+1} when $m = ((n/(2t+1))^{1/p} + 1)/2$. If $\mathcal{A}$ uses M bits of space, then the protocol uses at most $(p^2 + p - 1)M$ bits. Hence by Theorem 6.2, this implies that $M = \Omega(m/p^6) = \Omega((n/t)^{1/p}p^{-6})$.

The reduction proceeds as follows. Consider a $(p+1)$-level, m-ary tree T, where we say v has level j if the distance between v and the closest leaf is $j - 1$. We start by defining some notation:

1. For $j \in [p+1], i_p, \ldots, i_j \in [m]$, let $v[i_p, \ldots, i_j]$ denote the i_jth child of $v[i_p, \ldots, i_{j+1}]$, where $v[]$ is the root of the tree.

S_1	S_2	S_3
$(0,0,0)\times 5(3 - f_1(1))$		
	$(1,0,0)\times (3 - f_2(1))$	$(1,1,f_3(1)),(1,2,f_3(2)),(1,3,f_3(3))$
	$(1,4,0)\times (f_2(1) - 1)$	
	$(2,0,0)\times (3 - f_2(2))$	$(2,1,f_3(1)),(2,2,f_3(2)),(2,3,f_3(3))$
	$(2,4,0)\times (f_2(2) - 1)$	
	$(3,0,0)\times (3 - f_2(3))$	$(3,1,f_3(1)),(3,2,f_3(2)),(3,3,f_3(3))$
	$(4,4,0)\times (f_2(3) - 1)$	
$(4,0,0)\times 5(f_1(1) - 1)$		

FIG. 6.1. *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* $\text{median}(S_1 \cup S_2 \cup S_3) = f_1(1) \cdot 5^2 + f_2(f_1(1)) \cdot 5^1 + f_3(f_2(f_1(1))) \cdot 5^0$.

2. Let the $(p + 1)$ tuple $(h_p, \ldots, h_0)$ denote $\sum_{i=0}^{p} h_i (m + 2)^i$.
3. For each internal node of level j, e.g., $v = v[i_p, \ldots, i_j]$, we associated a multi-set of elements $S(v)$ of size a_j. Let $a_1 = 2t + 1$ and $a_j = (m - 1)b_{j-1}$, where

$$b_{j-1} = a_{j-1} + m a_{j-2} + m^2 a_{j-3} + \cdots + m^{j-2} a_1 \ .$$

Note that $|\cup_{v \in V(T)} S(v)| = b_{p+1} = (2m - 1)^p (2t + 1)$. $S(v)$ contains

$$b_{j-1}(m - f_{p+2-j}(i_j)) \text{ copies of } (i_p, \ldots, i_j, 0, 0, \ldots 0)$$
$$\text{and } b_{j-1}(f_{p+2-j}(i_j) - 1) \text{ copies of } (i_p, \ldots, i_j, m + 1, 0, \ldots, 0),$$

where we define $i_{p+1} = 1$.
4. For a leaf node, e.g., $v = v[i_p, \ldots, i_1]$, we generate $2t + 1$ copies of

$$(i_p, \ldots, i_1, f_{p+1}(i_1)) \ .$$

It can be shown by induction that any t-approximate median of $\cup_{v \in V(T)} S(v)$ equals $(g_1, g_2, \ldots, g_{p+1})$. See Figure 6.1 for the case when $p = 2, m = 3$, and $t = 0$.

Let S_j be the union of $S(v)$ over all v in the jth layer. Note that S_j can be determined by the $(p + 2 - j)$th player, P_{p+2-j}, who knows the function f_{p+2-j}. The players emulate $\mathcal{A}$ on the stream $\langle S_1, S_2, \ldots, S_{p+1} \rangle$ in the standard way: P_{p+1} runs $\mathcal{A}$ on S_1, transmits the memory state to P_p, who instantiates the algorithm with the transmitted memory state and continues running $\mathcal{A}$ on S_2, etc., until p passes of the algorithm have been emulated. Note that this is a p-round protocol in which $M(p(p + 1) - 1)$ bits are communicated. The result follows. □

6.1. Proof of Theorem 6.2. The proof is a generalization of a proof by Nisan and Widgerson [22]. We present the entire argument for completeness. In the proof we lower bound the $(p - 1)$-round distributional complexity, $D_{1/20}^{p-1}(g_p)$; 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

$$D_{1/20}^{p-1}(g_p) \le 2R_{1/10}^{p-1}(g_p) \ .$$

Let T be the protocol tree of a deterministic p-round protocol. We consider the input distribution, where each f_i is chosen uniformly from F, the set of all m^m functions from $[m]$ to $[m]$. Note that this distribution over inputs gives rise to a 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 $i \leq j$; e.g., for $p = 4$ the appended information is shown in the following table, where $g_0 = 1$.

	Round 1				Round 2				Round 3			
Player	4	3	2	1	4	3	2	1	4	3	2	1
Appended	g_0	g_0	g_0	g_1	g_1	g_1	g_2	g_2	g_2	g_3	g_3	-

This is possible with only $O(p^2 \log m)$ 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 $\epsilon = 10^{-4}(p+1)^{-4}$, and derive a contradiction.

Consider a node z in the protocol tree of T corresponding to the jth 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 $g_0, g_1, \ldots, g_{t-1}$ are specified by the messages so far.

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

DEFINITION 6.4. *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:*

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

where $H(\cdot)$ is the binary entropy.

CLAIM 1. *Given the protocol reaches node z and z is nice,*

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

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 that we have reached z is $\prod_{1 \leq j \leq p} |F_j^w|/|F_j^z| = |F_i^w|/|F_i^z|$. Furthermore, since z is nice,

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

where the second-to-last inequality follows from $a_w \geq \lg m$ and the last inequality follows by Kraft's inequality. Hence, with probability at least $1 - 1/m$, the next node in the protocol tree satisfies the first condition of 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. In this 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, $(f_{t+1}^w(1), \ldots, f_{t+1}^w(m))$, 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 subadditivity 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 $[m]$, then

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

by Markov's inequality. However, we are not interested in a v chosen uniformly at random but rather $v = g_t = f_t^z(g_{t-1})$. But, since the entropy of $f_t^z(g_{t-1})$ is large, it is "almost" distributed uniformly. Specifically, since $H(f_t^z(g_{t-1})) \geq \lg m - \delta$,

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

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. $\square$

Note that the height of the protocol tree is $p(p-1)$ 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 $p(p-1)(1/m + 4\sqrt{\epsilon}) \leq 1/25$. If the final leaf node is nice, 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_p correctly is at most $1 - 1/20$.

7. Conclusions. In this paper we motivated the study of random-order data streams and presented the first extensive study of the theoretical issues that arise in this model. We studied these issues in the context of quantile estimation, one of the most well studied problems in the data-stream model. Our results demonstrated some of the trade-offs that arise between space, passes, and accuracy in both the random-order and adversarial-order models. We resolved a long-standing open question of Munro and Paterson [21] by devising an $O(\text{polylog } n)$-space, $O(\log \log n)$-pass algorithm for exact selection in a randomly ordered stream of n elements. We also resolved an open question of Kannan [17] by demonstrating an exponential separation between the random-order and adversarial-order models: using $O(\text{polylog } n)$ space, exact selection requires $\Omega(\log n/\log \log n)$ passes in the adversarial-order model.

REFERENCES

[1] A. CHAKRABARTI, G. CORMODE, AND A. MCGREGOR, *Robust lower bounds for communication and stream computation*, in Proceedings of the 40th Annual ACM Symposium on Theory of Computing (British Columbia, Canada), 2008, pp. 641–650.
[2] A. CHAKRABARTI, T. S. JAYRAM, AND M. PĂTRAȘCU, *Tight lower bounds for selection in randomly ordered streams*, in Proceedings of the 19th Annual ACM-SIAM Symposium on Discrete Algorithms (San Francisco, CA), 2008, pp. 720–729.

[3] G. Cormode, F. Korn, S. Muthukrishnan, and D. Srivastava, *Space- and time-efficient deterministic algorithms for biased quantiles over data streams*, in Proceedings of the 25th ACM SIGMOD–SIGACT–SIGART Symposium on Principles of Database Systems (Chicago, IL), 2006, pp. 263–272.

[4] G. Cormode and S. Muthukrishnan, *An improved data stream summary: The count-min sketch and its applications*, J. Algorithms, 55 (2005), pp. 58–75.

[5] E. D. Demaine, A. López-Ortiz, and J. I. Munro, *Frequency estimation of internet packet streams with limited space*, in Proceedings of the 10th Annual European Symposium on Algorithms, Springer-Verlag, London, 2002, pp. 348–360.

[6] J. Feigenbaum, S. Kannan, M. Strauss, and M. Viswanathan, *Testing and spot-checking of data streams*, Algorithmica, 34 (2002), pp. 67–80.

[7] P. B. Gibbons and Y. Matia, *Synopsis data structures for massive data sets*, in External Memory Algorithms. Papers from the DIMACS Workshop on External Memory Algorithms (Rutgers University, Piscataway, NJ), J. M. Abello and J. S. Vitter, eds., DIMACS Ser. Discrete Math. Theoret. Comput. Sci. 50, AMS, Providence, RI, 1999, pp. 39–70.

[8] P. B. Gibbons, Y. Matias, and V. Poosala, *Fast incremental maintenance of approximate histograms*, ACM Trans. Database Syst., 27 (2002), pp. 261–298.

[9] A. C. Gilbert, Y. Kotidis, S. Muthukrishnan, and M. Strauss, *How to summarize the universe: Dynamic maintenance of quantiles*, in Proceedings of the 28th International Conference on Very Large Data Bases (Hong Kong), 2002, pp. 454–465.

[10] M. Greenwald and S. Khanna, *Efficient online computation of quantile summaries*, in Proceedings of the ACM SIGMOD International Conference on Management of Data (Santa Barbara, CA), 2001, pp. 58–66.

[11] S. Guha and A. McGregor, *Approximate quantiles and the order of the stream*, in Proceedings of the 25th ACM SIGMOD–SIGACT–SIGART Symposium on Principles of Database Systems (Chicago, IL), 2006, pp. 273–279.

[12] S. Guha and A. McGregor, *Lower bounds for quantile estimation in random-order and multipass streaming*, in Proceedings of the International Colloquium on Automata, Languages and Programming, Lecture Notes in Comput. Sci. 4596, Springer, New York, 2007, pp. 704–715.

[13] S. Guha and A. McGregor, *Space-efficient sampling*, in Proceedings of the AISTATS, 2007, pp. 169–176. Available online at http://www.stat.umn.edu/~aistat/proceedings/start.htm

[14] S. Guha, A. McGregor, and S. Venkatasubramanian, *Streaming and sublinear approximation of entropy and information distances*, in Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithms (Miami, FL), 2006, pp. 733–742.

[15] A. Gupta and F. Zane, *Counting inversions in lists*, Proceedings of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms (Baltimore, MD), 2003, pp. 253–254.

[16] W. Hoeffding, *Probability inequalities for sums of bounded random variables*, J. Amer. Statist. Assoc., 58 (1963), pp. 13–30.

[17] S. Kannan, *Open problems in data stream algorithms*, in Proceedings of the DIMACS Workshop on Streaming Data Analysis and Mining (Rutgers University, Piscataway, NJ), available online at http://dimacs.rutgers.edu/Workshops/Streaming/abstracts.html, 2001.

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

[19] G. S. Manku, S. Rajagopalan, and B. G. Lindsay, *Approximate medians and other quantiles in one pass and with limited memory*, in Proceedings of the ACM SIGMOD International Conference on Management of Data (Seattle, WA), 1998, pp. 426–435.

[20] G. S. Manku, S. Rajagopalan, and B. G. Lindsay, *Random sampling techniques for space efficient online computation of order statistics of large datasets*, in Proceedings of the ACM SIGMOD International Conference on Management of Data (Philadelphia, PA), 1999, pp. 251–262.

[21] J. I. Munro and M. Paterson, *Selection and sorting with limited storage*, Theoret. Comput. Sci., 12 (1980), pp. 315–323.

[22] N. Nisan and A. Wigderson, *Rounds in communication complexity revisited*, SIAM J. Comput., 22 (1993), pp. 211–219.

[23] N. Shrivastava, C. Buragohain, D. Agrawal, and S. Suri, *Medians and beyond: New aggregation techniques for sensor networks*, in Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems (Baltimore, MD), 2004, pp. 239–249.

[24] A. C. Yao, *Lower bounds by probabilistic arguments*, in Proceedings of the 24th Annual IEEE Symposium on Foundations of Computer Science (Tuscon, AZ), 1980, pp. 420–428.