Approximate Sorting of Data Streams with Limited Storage

Farzad Farnoud (Hassanzadeh), Eitan Yaakobi, and Jehoshua Bruck

California Institute of Technology, Pasadena, CA, USA
{farnoud,yaakobi,bruck}@caltech.edu

Abstract. We consider the problem of approximate sorting of a data stream (in one pass) with limited internal storage where the goal is not to rearrange data but to output a permutation that reflects the ordering of the elements of the data stream as closely as possible. Our main objective is to study the relationship between the quality of the sorting and the amount of available storage. To measure quality, we use permutation distortion metrics, namely the Kendall tau and Chebyshev metrics, as well as mutual information, between the output permutation and the true ordering of data elements. We provide bounds on the performance of algorithms with limited storage and present a simple algorithm that asymptotically requires a constant factor as much storage as an optimal algorithm in terms of mutual information and average Kendall tau distortion.

1 Introduction

In many applications, such as sensor networks, finance, and web applications, data may be available as a transient stream that is not permanently accessible [1]. Often, in these applications, the large volume of data or time constraints prevent storage of the whole stream before processing. Even if data is locally stored, certain storage media only allow sequential access in a time-efficient manner.

In this paper, we study the fundamental problem of sorting a data stream when internal storage is limited. As the nature of the problem makes rearranging the data into a sorted stream impossible, by sorting we mean determining the ordering of the elements of the stream. In our model, the amount of available internal storage limits the number of elements of the data stream that can be stored internally. Furthermore, only elements in internal storage can be compared with each other. Lack of storage capable of holding the whole data stream implies that sorting must be approximate; the goal is to produce a permutation that represents the ordering of the elements of the data stream as faithfully as possible. As in [1], we consider algorithms that make only one pass over the data stream.

To evaluate performance, we measure the distortion between the output permutation and the permutation representing the true ordering of the data. There are many possible distortion measures on permutations [6], among which we consider the Kendall tau metric and the Chebyshev metric. The Kendall tau metric can be viewed as the number of mistakes made by the algorithm, while the

Z. Cai et al. (Eds.): COCOON 2014, LNCS 8591, pp. 465–476, 2014.

Chebyshev metric represents the maximum error in the rank of any element. Another quality measure considered in the paper is the mutual information between the true permutation and the output permutation, which reflects the amount of relevant information present in the output.

We first provide universal bounds on the performance of algorithms with limited storage, namely an upper bound on mutual information, and lower bounds on distortion, between the true permutation and the output. Further, we present a simple algorithm that is asymptotically optimal in terms of mutual information and asymptotically requires a constant factor as much storage as any algorithm with the same average Kendall tau distortion. For the Chebyshev distortion, the algorithm is also asymptotically constant-factor-optimal, provided that normalized distortion, to be defined later, is bounded away from 0.

The problem of sorting a data stream with limited storage goes back to the work of Munro and Paterson [12], where they considered sorting of, and selecting from, data stored on a read-only tape and showed that for exact sorting of a stream of length n in p passes, one requires storage of size $\Theta(n/p)$. While they allowed making multiple passes over the data and considered only exact sorting, in this work we study the quality of *approximate* sorting that can be obtained in *one* pass. Since the work of Munro and Paterson, many papers have studied problems related to *selection* in data streams, such as finding the kth highest value or quantiles, in one or many passes, e.g., [2, 8, 10, 11]. The problem of approximate sorting in one pass, however, to the best of our knowledge, has not been studied.

The rest of this paper is organized as follows. In Section 2, we present the formal problem statement and preliminaries. Section 3 includes universal bounds on the performance of algorithms with limited storage. In Section 4, an algorithm for sorting with limited storage is given and its performance is analyzed.

2 Problem Statement and Preliminaries

For a positive integer n, we let $[n] = \{1, \ldots, n\}$. The set of all permutations of $[n]$ is denoted by $\mathbb{S}_n$. For a permutation $\pi \in \mathbb{S}_n$ and distinct $i, j \in [n]$, we use $i \prec_\pi j$ (resp. $i \succ_\pi j$) to denote that i appears before j (resp. after j) in π. For example, if $\pi = (2, 3, 1)$, we have $2 \prec_\pi 3$ and $1 \succ_\pi 2$. The inverse of π is denoted by π^{-1}. The *rank* of element i in π is its position in π, that is, $\pi^{-1}(i)$.

The data stream is denoted by the sequence $s = s_1, s_2, \ldots, s_n$. We assume there is a permutation $X \in \mathbb{S}_n$ that represents the ordering of the elements of s; if $i \prec_X j$, then $s_i < s_j$ with respect to X. The goal is to approximate X as closely as possible. While X is not directly accessible in our setting, the relationship between every two elements s_i and s_j of s can be queried (or computed) if they are both present in internal storage, and the result of the query is either $i \prec_X j$ or $j \prec_X i$. Throughout the paper, our assumption is that X is chosen uniformly and at random among the permutations of $\mathbb{S}_n$ but we only consider deterministic algorithms.

The elements of s are revealed in a streaming fashion, i.e., one by one. If an element of the stream is not stored internally when revealed, it will not be possible to access it in the future. The storage limitation is that there are m cells each of which can store one element of s and thus any algorithm can only access m elements of the sequence s at any one time. The set of these m cells is termed *stream memory.* When a new element s_i of the stream s arrives, it can only be stored in the stream memory if there is an empty cell or if the contents of a cell is discarded; otherwise, s_i is ignored. To make a query regarding the relative order of s_i and s_j with respect to X, both s_i and s_j should be stored in the stream memory. We do not impose any other type of storage limitation. For example, there is no restriction on the number of integer values that an algorithm can store and access. This assumption is for simplifying the analysis and is also valid when each element of s is much larger than other types of data that an algorithm may require. To avoid trivial cases, we assume $n, m \geq 2$.

The output of the algorithms considered here is a permutation, denoted Y. To measure performance, we evaluate how "close" Y is to X. Closeness between two permutations can be quantified in a variety of ways. We use the Kendall tau and Chebyshev metrics, defined below, as well as the mutual information between X and Y.

The *Kendall tau distance* between two permutations $\pi, \sigma \in \mathbb{S}_n$ is the number of pairs of distinct elements i and j such that $i \prec_\pi j$ and $j \prec_\sigma i$, or equivalently, the number of adjacent transpositions needed to take π to σ. This distance is denoted as $d_\tau(\pi, \sigma)$. The Chebyshev distance between π and σ, denoted $d_C(\pi, \sigma)$, is defined as

$$\max_{i \in [n]} \left| \pi^{-1}(i) - \sigma^{-1}(i) \right| \ .$$

In other words, the Chebyshev distance is the maximum difference in the rank of any element in the two permutations.

For two functions f_n and g_n of n, the notation $f_n \sim g_n$ is used to denote $\lim_{n\to\infty} f_n/g_n = 1$. Furthermore, we use lg and ln as shorthands for $\log_2$ and $\log_e$, respectively.

3 Universal Bounds

In this section, we present bounds on the performance of any algorithm that can only store m elements of the sequence s. To derive these bounds, we use the fact that to make a query for comparing two elements s_i and s_j, both need to be present in the stream memory and so the amount of information that can be obtained via queries is limited because of the limitation on storage. As mentioned earlier, X is a random element of $\mathbb{S}_n$ but only deterministic algorithms are considered. We first present bounds on the mutual information between X and the output permutation Y and then consider distortion under the Kendall tau and Chebyshev metrics.

We use $H(X)$ and $I(X;Y)$ to refer to the entropy of X and the mutual information between X and Y, respectively. For these functions, logarithms are base 2. Note that as X is a random element of $\mathbb{S}_n$, we have $H(X) = \lg n!$.

Theorem 1. *For any algorithm with stream memory of size m, we have*

$$I\left(X;Y\right) \leq n\lg m - m\lg e + O(\lg m)\ .$$

Furthermore, $I\left(X;Y^*\right)/H\left(X\right) \sim \lg m/\lg n$ *for* $m,n \to \infty$*, where* Y^* *is the output of an algorithm that maximizes the mutual information between* X *and* Y*.*

Proof. Let Z be the set of responses provided to the comparison queries made by the algorithm. Since the algorithm can only have access to X through Z, by the data processing inequality [5], we have $I(Y;X) \leq I(Y;Z)$. Furthermore, $I(Y;Z) \leq H\left(Z\right)$.

We now show that $H\left(Z\right) \leq \lg m! + (n-m)\lg m$. The first m elements of s can be fully compared and so $m!$ cases arise from their ordering. Let Z_0' be an integer in $[m!]$ denoting the permutation representing the ordering of the first m elements. Each of the next $n-m$ elements can at most be compared with $m-1$ elements already present in the stream memory. These $m-1$ elements define m intervals, into one of which the new element falls. For $i \in \{m+1,\ldots,n\}$, let Z_i' be an integer in $[m]$ denoting the interval in which the ith element of the stream falls. Given the algorithm, Z is a deterministic function of $\left(Z_0', Z_{m+1}', Z_{m+2}',\ldots,Z_n'\right)$ and thus

$$\begin{aligned} H\left(Z\right) &\leq H\left(Z_0', Z_{m+1}', Z_{m+2}',\ldots,Z_n'\right) \\ &\leq H\left(Z_0'\right) + \sum_{i=m+1}^{n} H\left(Z_i'\right) \\ &\leq \lg m! + (n-m)\lg m\ . \end{aligned}$$

It follows that $I\left(X;Y\right) \leq H\left(Z\right) \leq \lg m! + (n-m)\lg m$. The first theorem statement then follows from the Stirling approximation: For a positive integer k, we have $\lg k! = k\lg k - k\lg e + O\left(\lg k\right).$

Since $I\left(X;Y\right) \leq \lg m! + (n-m)\lg m$ holds for $Y = Y^*$, we have

$$\frac{I\left(X;Y^*\right)}{H\left(X\right)} \leq \frac{n\lg m + O(m)}{n\lg n + O(n)} = \frac{\lg m}{\lg n}\left(1 + o\left(1\right)\right)\ , \tag{1}$$

where we have used the fact that $\frac{m}{n\lg m} = O(1/\lg n) = o(1)$. In Section 4, we present an algorithm that produces an output Y_1 such that

$$\frac{I\left(X;Y_1\right)}{H\left(X\right)} \geq \frac{\lg m}{\lg n}\left(1 + o\left(1\right)\right), \quad m,n \to \infty\ \cdot$$

Since $I(X;Y^*) \geq I\left(X;Y_1\right)$, we have

$$\frac{I\left(X;Y^*\right)}{H\left(X\right)} \geq \frac{\lg m}{\lg n}\left(1 + o\left(1\right)\right), \quad m,n \to \infty\ \cdot \tag{2}$$

The second statement of the theorem follows from (1) and (2). □

In particular, if $m = n^{\beta} + O(1)$ for a constant β, then a β fraction of the information of X can be recovered by an algorithm with stream memory m.

Next, we use the rate-distortion theory to find lower bounds on the average Kendall tau distortion between X and Y, defined as $E[d_{\tau}(X,Y)]$. We use δ to denote the normalized version of this distortion, that is, $\delta = E[d_{\tau}(X,Y)]/n$. This choice leads to simpler expressions. Note that since d_{τ} can be of the order of n^2, δ can take on values in the range $[0,\infty)$.

The following theorem applies to any algorithm with stream memory m. We use W_0 and W_{-1} to respectively denote the principal and the lower branches of the Lambert W function. The Lambert W function $W(x)$ is defined as the function satisfying $W(x)e^{W(x)} = x$ [4].

Theorem 2. *Let $\mu = \frac{m}{n}$ and $\delta = \frac{E[d_{\tau}(X,Y)]}{n}$. Suppose ϵ is a positive constant. For any algorithm with stream memory m and $\delta > \epsilon$, we have*

$$\mu \geq -W_0\left(-\frac{\delta^{\delta}}{e(1+\delta)^{1+\delta}}\right)\left(1-\frac{K_{\epsilon}\lg n}{n}\right) \quad , \tag{3}$$

where K_{ϵ} is a constant that depends on ϵ, and

$$\mu \geq \frac{1}{e^2\delta}\left(1+O\left(\frac{\lg n}{n}\right)+O\left(\frac{1}{\delta}\right)\right) \quad . \tag{4}$$

Proof. Since we only consider deterministic algorithms, the number M of outputs of a given algorithm is bounded from above by $m!m^{n-m}$. This statement can be proven in a similar manner to the upper bound on $H(Z)$ in Theorem 1.

Let $A = \frac{1}{n}\lg\frac{M}{n!}$. We have $\lg M \leq n\lg m - m\lg e + O(\lg m)$ and so

$$\begin{aligned} A &\leq \frac{1}{n}(n\lg m - m\lg e - n\lg n + n\lg e + O(\lg n)) \\ &= \lg\left(\mu e^{1-\mu}\right) + O\left(n^{-1}\lg n\right) \quad . \end{aligned}$$

Hence, there exists a positive constant K_1 such that $A \leq \lg\left(\mu e^{1-\mu}\right) + K_1\lg n/n$.

The parameter M can be viewed as the size of a rate-distortion code. Hence, from [7, Theorem 5], we have the following relationship between the average distortion $E\left[d_{\tau}(X,Y)\right]$ and M, expressed in terms of δ and A,

$$A \geq \lg\frac{\delta^{\delta}}{(1+\delta)^{1+\delta}} - \frac{\lg n}{n} \quad .$$

From this and the fact that $A \leq \lg\left(\mu e^{1-\mu}\right) + K_1\lg n/n$, we obtain

$$\mu e^{1-\mu} \geq \frac{\delta^{\delta}}{(1+\delta)^{1+\delta}}\left(1-K_2\frac{\lg n}{n}\right) \quad ,$$

where $K_2 = (1+K_1)\ln 2$, or equivalently,

$$-\mu e^{-\mu} \leq \frac{-\delta^{\delta}}{e(1+\delta)^{1+\delta}}\left(1-K_2\frac{\lg n}{n}\right) \quad .$$

Hence

$$\mu \geq -W_0\left(\frac{-\delta^\delta}{e(1+\delta)^{1+\delta}}\left(1-K_2\frac{\lg n}{n}\right)\right) \ . \tag{5}$$

For convenience, let $g\left(\delta\right)=\frac{\delta^\delta}{e(1+\delta)^{1+\delta}}$. By taking derivatives, one can show that the function W_0 is concave. Hence,

$$\begin{aligned}
W_0\left(-g(\delta)+g(\delta)K_2\frac{\lg n}{n}\right) &\leq W_0(-g(\delta))+W_0'(-g(\delta))g(\delta)K_2\frac{\lg n}{n}\\
&= W_0(-g(\delta))+\frac{W_0(-g(\delta))}{-g(\delta)\left(1+W_0(-g(\delta))\right)}g(\delta)K_2\frac{\lg n}{n}\\
&= W_0(-g(\delta))\left(1-\frac{K_2}{1+W_0(-g(\delta))}\frac{\lg n}{n}\right)\\
&\leq W_0(-g(\delta))\left(1-K_\epsilon\frac{\lg n}{n}\right) \ ,
\end{aligned}$$

where $K_\epsilon=\frac{K_2}{1+W_0(-g(\epsilon))}$. Note that for $\delta\geq 0$, the expression $-g(\delta)$ is strictly increasing and we have $-g(\delta)\in[-1/e,0)$. Since $\epsilon>0$, we have $-g\left(\epsilon\right)>-1/e$ and so $W_0\left(-g\left(\epsilon\right)\right)>-1$. Thus K_ϵ is well defined. Furthermore, $W_0\left(-g\left(\delta\right)\right)>W_0\left(-g\left(\epsilon\right)\right)$ for $\delta>\epsilon$ and from this and the fact that $W_0(-g(\delta))<0$, the last step of the above derivation follows. We finally have,

$$\mu \geq -W_0\left(-\frac{\delta^\delta}{e(1+\delta)^{1+\delta}}\right)\left(1-\frac{K_\epsilon\lg n}{n}\right) \ . \tag{6}$$

To prove the second statement, note that by concavity of W_0 and the facts that $W_0(0)=0$ and $W_0'(0)=1$, we have

$$\begin{aligned}
W_0\left(-\frac{\delta^\delta}{e(1+\delta)^{1+\delta}}\right) &\leq -\frac{\delta^\delta}{e(1+\delta)^{1+\delta}}\\
&\leq -\frac{1}{e^2(1+\delta)}\\
&= -\frac{1+O(1/\delta)}{e^2\delta} \ .
\end{aligned} \tag{7}$$

The second statement of the theorem then follows from (6) and (7). □

Finally, we consider the Chebyshev distortion between X and Y. The normalized Chebyshev distortion is $\chi=E\left[d_C(X,Y)\right]/n$. We only consider the case of $\chi\leq 1/2$ which is more important as it represents small distortions.

Theorem 3. *Let $\mu=\frac{m}{n}$ and $\chi=\frac{E[d_C(X,Y)]}{n}$. Suppose $2/n\leq\chi\leq 1/2$. For any algorithm with stream memory m, we have*

$$\mu \geq -W_0\left(-\frac{(e/2)^{2\chi}}{2\chi n}\right)\left(1+O\left(n^{-1}\lg n\right)\right) \ .$$

Proof. Let M be defined as in the proof of Theorem 2 and let $R = \frac{1}{n}\lg M$. Since $\lg M \le n\lg m - m\lg e + O(\lg m)$, we have $R \le \lg m - \frac{m}{n}\lg e + O(n^{-1}\lg m)$. From [7, Theorem 16], we find $R \ge \lg\frac{1}{2\chi} + 2\chi\lg\frac{e}{2} + O(n^{-1}\lg n)$ for $\chi \le 1/2$. Hence,

$$\lg\frac{1}{2\chi} + 2\chi\lg\frac{e}{2} \le \lg m - \frac{m}{n}\lg e + O(n^{-1}\lg n)$$

implying that $\mu e^{-\mu} \ge \frac{(e/2)^{2\chi}}{2\chi n}\left(1 + O\left(n^{-1}\lg n\right)\right)$, or equivalently,

$$\mu \ge -W_0\left(-\frac{(e/2)^{2\chi}}{2\chi n}\left(1 + O\left(n^{-1}\lg n\right)\right)\right) .$$

Since $\chi \le 1/2$, we have $(e/2)^{2\chi} \le e/2$ and since $\chi \ge 2/n$, we have $2\chi n \ge 4$. So $-\frac{(e/2)^{2\chi}}{2\chi n} \ge -\frac{e}{8} > -\frac{1}{e}$. Hence, $W_0\left(-\frac{(e/2)^{2\chi}}{2\chi n}\right)$ is bounded away from -1. We have

$$\begin{aligned}
&W_0\left(-\frac{(e/2)^{2\chi}}{2\chi n}\left(1 + O\left(n^{-1}\lg n\right)\right)\right) \\
&\quad\overset{(\mathsf{a})}{\le} W_0\left(-\frac{(e/2)^{2\chi}}{2\chi n}\right) + W_0'\left(-\frac{(e/2)^{2\chi}}{2\chi n}\right)\frac{(e/2)^{2\chi}O\left(n^{-1}\lg n\right)}{2\chi n} \\
&\quad= W_0\left(-\frac{(e/2)^{2\chi}}{2\chi n}\right) + W_0\left(-\frac{(e/2)^{2\chi}}{2\chi n}\right)\frac{O\left(n^{-1}\lg n\right)}{1 + W_0\left(-\frac{(e/2)^{2\chi}}{2\chi n}\right)} \\
&\quad\overset{(\mathsf{b})}{=} W_0\left(-\frac{(e/2)^{2\chi}}{2\chi n}\right)\left(1 + O\left(n^{-1}\lg n\right)\right) .
\end{aligned}$$

where (a) and (b) follow from the concavity of W_0 and the fact that $1 + W_0\left(-\frac{(e/2)^{2\chi}}{2\chi n}\right)$ is bounded away from 0, respectively. □

4 Algorithm for Limited-Storage Approximate Sorting

We present the following simple algorithm for approximately sorting a stream using storage of size m and then present results regarding its performance. Let $c_1, \ldots, c_m$ denote the m memory cells capable of storing elements of the stream. Recall that $s_i < s_j$ if i appears before j in X, i.e., $i \prec_X j$.

Algorithm 1

1. Store the first $m-1$ elements of s in memory cells $c_1, \ldots, c_{m-1}$.
2. Find permutation y of $\{1, \ldots, m-1\}$ such that $s_{y_1} < s_{y_2} < \ldots < s_{y_{m-1}}$.
3. Let $Y_1 \leftarrow y$.
4. For each new element $s_i, i = m, m+1, \ldots, n$, of the stream:
   (a) Store s_i in c_m.
   (b) If there exists j such that $s_{y_{j-1}} < s_i < s_{y_j}$, insert i immediately before y_j in Y_1.
   (c) If $s_i < s_j$ for all $j \in [m-1]$, insert i immediately before y_1 in Y_1.
   (d) If $s_i > s_j$ for all $j \in [m-1]$, append i to the end of Y_1.

In this algorithm, the first $m-1$ elements, namely, $s_1, \ldots, s_{m-1}$, are stored in the memory for the duration of the algorithm and every new element is compared with these. An element that is stored in memory, for the purpose that new elements can be compared with it, is called a *pivot*.

Example 1. Suppose $X = (5,4,2,3,7,6,1,9,8)$ and $m = 3$. After step 3 of Algorithm 1, we have $y = Y_1 = (2,1)$. For $i = 3$, Y_1 is updated to $(\mathbf{2},3,\mathbf{1})$, where the indices of the pivots are shown in bold. For $i = 4$ and $i = 5$, Y_1 is respectively updated to $(4,\mathbf{2},3,\mathbf{1})$ and $(4,5,\mathbf{2},3,\mathbf{1})$. The final output is $Y_1 = (4,5,\mathbf{2},3,6,7,\mathbf{1},8,9)$. For the Kendall tau and Chebyshev distortions, we have $d_\tau(X,Y_1) = 3$ and $d_C(X,Y_1) = 1$.

In Algorithm 1, the index set of pivots is $\{1,2,\ldots,m-1\}$ and they are in correct order in Y_1. However, indices of elements between the pivots, and between the pivots and the boundaries, are sorted in the natural increasing order which may differ from their order in X, e.g., the subsequence $3,6,7$ of Y_1 in the preceding example. Let $r_1, \ldots, r_{m-1}$ be an increasing sequence that denotes the positions of the indices of the pivots in X (or equivalently in Y_1). Furthermore, let $r_0 = 0$ and $r_m = n+1$. In Example 1, we have $r_0 = 0$, $r_1 = 3$, $r_2 = 7$, and $r_3 = 10$. For $j \in [m]$, the elements of Y_1 between positions r_{j-1} and r_j can have any order in X. Additionally, all possibilities are equally probable. Given Y_1, the number of possible cases for X is given by $\prod_{j=1}^{m} (r_j - r_{j-1} - 1)!$. We will use this fact to compute the conditional entropy of X given Y_1 in the next theorem.

Theorem 4. *Algorithm* 1 *is asymptotically optimal for $m, n \to \infty$, with respect to mutual information.*

Proof. Since $I(X;Y_1) = H(X) - H(X|Y_1)$ and $H(X) = \lg n!$, to find $I(X;Y_1)$, it suffices to find $H(X|Y_1)$. We have

$$\begin{aligned} H(X|Y_1) &= \sum_{z \in \mathbb{S}_n} P(Y_1 = z) H(X|Y_1 = z) \\ &= \binom{n}{m-1}^{-1} \sum_{r_0 < \cdots < r_m} \sum_{j=1}^{m} \lg\,(r_j - r_{j-1} - 1)! \ . \end{aligned}$$

For given values of $r_0, \ldots, r_m$, the set $[n]$ is divided into m blocks with lengths $r_j - r_{j-1} - 1$. To compute the above sum, we count how many times a block of size k occurs for all possible values of $r_0, \ldots, r_m$. The number of times a block of length k appears starting at position 1 equals $\binom{n-k-1}{m-2}$ since we have $r_1 = k+1$ but must choose the values of $r_2, \ldots, r_{m-1}$ among the $n-(k+1)$ possibilities. The number of times a block of size k ends at position n is the same. A similar argument shows that the number of times a block of length k starts at position i and ends at position $i+k-1$, for each $i \in \{2, \ldots, n-k\}$, is $\binom{n-k-2}{m-3}$. Thus, the total number of blocks of size k is $2\binom{n-k-1}{m-2} + (n-k-1)\binom{n-k-2}{m-3} = m\binom{n-k-1}{m-2}$. Hence,

$$H(X|Y_1) = \binom{n}{m-1}^{-1} m \sum_{k=2}^{n-m+1} \binom{n-k-1}{m-2} \lg k! \ . \tag{8}$$

It can be shown that

$$\sum_{k=1}^{n-m+1} \binom{n-k-1}{m-2} k \lg k \leq \binom{n}{m} \left(\lg \frac{n}{m} + O(1) \right) \ . \tag{9}$$

From (8), (9), and the fact that $\lg k! < k \lg k$, we obtain

$$H\left(X|Y_1\right) \leq (n-m+1) \left(\lg \frac{n}{m} + O(1) \right) = n \lg \frac{n}{m} + O(n)$$

and so $I(X;Y_1) \geq \lg n! - n \lg \frac{n}{m} + O(n) = n \lg m + O(n)$. Thus $\frac{I(X;Y_1)}{H(X)} \geq \frac{\lg m}{\lg n}\left(1 + o\left(1\right)\right)$ for $m, n \to \infty$. Recall from the proof of Theorem 1 that $\frac{I(X;Y)}{H(X)} \leq \frac{\lg m}{\lg n}\left(1 + o\left(1\right)\right)$ for the output Y of any algorithm. Therefore,

$$\frac{I\left(X;Y_1\right)}{H\left(X\right)} = \frac{\lg m}{\lg n}\left(1 + o\left(1\right)\right), \quad m, n \to \infty \ ,$$

which is optimal. □

Next, we discuss the average Kendall tau distortion of Algorithm 1.

Theorem 5. *Suppose Algorithm 1 has stream memory $m = \mu_1 n$ and produces an output with average Kendall tau distortion δn. We have*

$$\mu_1 \leq \left(1 + \delta - \sqrt{\delta(\delta+2)}\right)\left(1 + O(1/n)\right) \ .$$

Furthermore, Algorithm 1 asymptotically requires at most a constant factor as much storage as an optimal algorithm with the same average Kendall tau distortion.

Proof. For a random permutation of length k, the average Kendall tau distance from the identity is $\frac{1}{2}\binom{k}{2}$. Hence, from the discussion preceding Theorem 4, we obtain

$$\begin{aligned} E\left[d_\tau(X,Y_1)\right] &= \binom{n}{m-1}^{-1} \sum_{r_0 < \cdots < r_m} \sum_{j=1}^{m} \frac{1}{2} \binom{r_j - r_{j-1} - 1}{2} \\ &= \frac{1}{2} \binom{n}{m-1}^{-1} m \sum_{k=2}^{n-m+1} \binom{k}{2} \binom{n-k-1}{m-2} \\ &= \frac{1}{m+1} \binom{n-m+1}{2} \ , \end{aligned} \tag{10}$$

where for the second equality, we have used an argument similar to that of the proof of Theorem 4. We have $\delta n = E\left[d_\tau(X,Y_1)\right] = \frac{1}{\mu_1 n + 1}\binom{n-\mu_1 n+1}{2}$ and thus $\delta n \leq \frac{(n-\mu_1 n+1)^2}{2\mu_1 n}$. It follows that

$$\begin{aligned} \mu_1 &\leq 1 + \delta + \frac{1}{n} - \sqrt{\delta(\delta + 2 + 2/n)} \\ &= \left(1 + \delta - \sqrt{\delta(\delta+2)}\right)\left(1 + O(1/n)\right) \ . \end{aligned}$$

In particular, for large δ, we have $\mu_1 \le 1/(2\delta)\,(1+O(1/\delta))\,(1+O(1/n))$.

Let $\mu^* n$ be the smallest amount of stream memory of any algorithm with average Kendall tau distortion δn. From (4), we have

$$\frac{\mu_1}{\mu^*} \le \frac{1/(2\delta)}{1/\left(e^2\delta\right)}\,(1+O(1/\delta))\,(1+O(\lg n/n)) \quad .$$

Thus there is a constant c such that for $\delta, n \ge c$, μ_1/μ^* is bounded.

On the other hand, if $\delta < c$, from (3) and using the fact that μ^* is a decreasing function of δ, we have $\mu^* \ge -W_0\left(-c^c e^{-1}(1+c)^{-1-c}\right)(1+O(\lg n/n))$. Furthermore $\mu_1 \le 1$. Hence, if $\delta < c$, then μ_1/μ^* is bounded. □

Remark 1. There is an alternative way to show that $E\left[d_\tau(X,Y_1)\right] = \frac{1}{m+1}\binom{n-m+1}{2}$. Without loss of generality, assume $1 \prec_X \cdots \prec_X m-1$. Consider distinct $i,j \in \{m,\dots,n\}$, with $i<j$. The pair i,j will have incorrect order in Y_1, if and only if $j \prec_X i$ and there is no $p \in \{1,\dots,m-1\}$ such that $j \prec_X p \prec_X i$ (in other words, there is no pivot s_p such that $s_j < s_p < s_i$). Since X is random, it is straightforward to see that the probability of this event is $1/(m+1)$. There are $\binom{n-m+1}{2}$ possible choices for the pair i,j. The desired result then follows by the linearity of expectation.

The next theorem concerns the average Chebyshev distortion of Algorithm 1.

Theorem 6. *Suppose Algorithm 1 has memory m and produces an output with average Chebyshev distortion χn. Furthermore, suppose that $\chi \le 1/2$ and $m \ge 2$. We have*

$$m \le -\frac{1}{\chi}W_{-1}\left(-\frac{\chi}{e}\right) \quad .$$

Additionally, if χ is bounded away from zero, Algorithm 1 asymptotically requires at most a constant factor as much memory as an optimal algorithm with the same average Chebyshev distortion.

Proof. Consider an element i in Y_1 that is between positions r_{j-1} and r_j. We know that the position of this element in X is also between r_{j-1} and r_j. Thus, $\left|X^{-1}(i) - Y_1^{-1}(i)\right| \le r_j - r_{j-1} - 1$ and so

$$d_C\left(X,Y_1\right) \le \max_j\left(r_j - r_{j-1} - 1\right) \quad .$$

Suppose a stick of length n is randomly broken at $m-1$ points. Let the length of the longest piece among the m pieces be denoted by S. From [9], we have $E[S] = nE[S']/m$, where S' is the largest random variable among m iid exponential random variables with mean 1. We have $E[S'] = \sum_{i=1}^m 1/i \le \ln m + \gamma_e + \frac{1}{2m}$, where the inequality follows from [3] and γ_e is Euler's constant. Since the positions of the pivots in Algorithm 1 are random, with a coupling argument one can show that the expected length of the longest segment is not more than $E[S]$. That is, $E\left[\max_j\left(r_j - r_{j-1} - 1\right)\right] \le E\left[S\right]$. Hence,

$$E[d_C(X,Y_1)] = \chi n \le \frac{n}{m}\left(\ln m + \gamma_e + \frac{1}{2m}\right) \quad .$$

Since $m \geq 2$ we have $\gamma_e + \frac{1}{2m} \leq 1$, and thus $\chi \leq \frac{\ln(me)}{m}$. This in turn implies that $-m\chi e^{-m\chi} \leq -\frac{\chi}{e}$, from which it follows that $-(1/\chi)W_0(-\chi/e) \leq m \leq -(1/\chi)W_{-1}(-\chi/e)$ and so we have the first statement in the theorem. Note that for $\chi \leq 1$, we have $-(1/\chi)W_0(-\chi/e) \leq 1$ and hence the inequality $-(1/\chi)W_0(-\chi/e) \leq m$ does not give us any useful information.

Let μ_1 denote m/n for Algorithm 1 and μ^* denote the smallest amount of storage of any algorithm with Chebyshev distortion χn. From Theorem 3,

$$\begin{aligned}\frac{\mu_1}{\mu^*} &\leq \frac{-\frac{1}{\chi n}W_{-1}\left(-\frac{\chi}{e}\right)}{-W_0\left(-\frac{(e/2)^{2\chi}}{2\chi n}\right)}\left(1 + O\left(\frac{\lg n}{n}\right)\right)\\ &\sim \frac{-\frac{1}{\chi n}W_{-1}\left(-\frac{\chi}{e}\right)}{\frac{(e/2)^{2\chi}}{2\chi n}}\\ &\sim \frac{-2W_{-1}\left(-\frac{\chi}{e}\right)}{(e/2)^{2\chi}}\ . \end{aligned} \tag{11}$$

Suppose χ is bounded away from 0. It follows that $-\chi/e$ is also bounded away from 0. This in turn implies that $-W_{-1}(-\chi/e)$ is bounded and so is the right side of (11). This completes the proof of the theorem. □

Acknowledgements. The authors would like to thank Ryan Gabrys and Yue Li for useful discussions and comments.

References

1. Babcock, B., Babu, S., Datar, M., Motwani, R., Widom, J.: Models and issues in data stream systems. In: Proc. 21st ACM Symp. Principles of Database Systems (PODS), New York, NY, USA (2002)
2. Chakrabarti, A., Jayram, T.S., Pătraşcu, M.: Tight lower bounds for selection in randomly ordered streams. In: ACM-SIAM Symp. Discrete Algorithms (SODA), pp. 720–729. Society for Industrial and Applied Mathematics, Philadelphia (2008), http://dl.acm.org/citation.cfm?id=1347082.1347161
3. Chen, C.P., Qi, F.: The best lower and upper bounds of harmonic sequence. Global Journal of Applied Mathematics and Mathematical Sciences 1(1), 41–49 (2008)
4. Corless, R.M., Gonnet, G.H., Hare, D.E.G., Jeffrey, D.J., Knuth, D.E.: On the Lambert W function. Advances in Computational Mathematics 5(1), 329–359 (1996), http://dx.doi.org/10.1007/BF02124750
5. Cover, T.M., Thomas, J.A.: Elements of information theory. John Wiley & Sons (2006)
6. Diaconis, P.: Group Representations in Probability and Statistics, vol. 11. Institute of Mathematical Statistics (1988)
7. Farnoud, F., Schwartz, M., Bruck, J.: Rate-distortion for ranking with incomplete information. arXiv preprint (2014), http://arxiv.org/abs/1401.3093
8. Greenwald, M., Khanna, S.: Space-efficient online computation of quantile summaries. In: Proc. ACM SIGMOD Int. Conf. Management of Data, pp. 58–66. ACM, New York (2001), http://doi.acm.org/10.1145/375663.375670

9. Holst, L.: On the lengths of the pieces of a stick broken at random. Journal of Applied Probability 17(3), 623–634 (1980)
10. Manku, G.S., Rajagopalan, S., Lindsay, B.G.: Approximate medians and other quantiles in one pass and with limited memory. In: Proc. ACM SIGMOD Int. Conf. Management of Data, pp. 426–435. ACM, New York (1998), http://doi.acm.org/10.1145/276304.276342
11. McGregor, A., Valiant, P.: The shifting sands algorithm. In: ACM-SIAM Symp. Discrete Algorithms (SODA), pp. 453–458. SIAM (2012), http://dl.acm.org/citation.cfm?id=2095116.2095155
12. Munro, J., Paterson, M.: Selection and sorting with limited storage. Theoretical Computer Science 12(3), 315–323 (1980), http://www.sciencedirect.com/science/article/pii/0304397580900614