A Tight Lower Bound for Comparison-Based Quantile Summaries

Graham Cormode
Pavel Veselý
{G.Cormode,Pavel.Vesely}@warwick.ac.uk
University of Warwick
United Kingdom

ABSTRACT

Quantiles, such as the median or percentiles, provide concise and useful information about the distribution of a collection of items, drawn from a totally ordered universe. We study data structures, called quantile summaries, which keep track of all quantiles of a stream of items, up to an error of at most ε. That is, an ε-approximate quantile summary first processes a stream and then, given any quantile query $0 \leq \phi \leq 1$, returns an item from the stream, which is a ϕ'-quantile for some $\phi' = \phi \pm \varepsilon$. We focus on comparison-based quantile summaries that can only compare two items and are otherwise completely oblivious of the universe.

The best such deterministic quantile summary to date, due to Greenwald and Khanna [6], stores at most $O(\frac{1}{\varepsilon} \cdot \log \varepsilon N)$ items, where N is the number of items in the stream. We prove that this space bound is optimal by showing a matching lower bound. Our result thus rules out the possibility of constructing a deterministic comparison-based quantile summary in space $f(\varepsilon) \cdot o(\log N)$, for any function f that does not depend on N. As a corollary, we improve the lower bound for biased quantiles, which provide a stronger, relative-error guarantee of $(1 \pm \varepsilon) \cdot \phi$, and for other related computational tasks.

CCS CONCEPTS

• **Theory of computation → Streaming models**; **Lower bounds and information complexity**.

KEYWORDS

lower bounds, quantiles

ACM Reference Format:
Graham Cormode and Pavel Veselý. 2020. A Tight Lower Bound for Comparison-Based Quantile Summaries. In *Proceedings of the 39th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems (PODS'20), June 14–19, 2020, Portland, OR, USA.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3375395.3387650

PODS'20, June 14–19, 2020, Portland, OR, USA

ACM ISBN 978-1-4503-7108-7/20/06…$15.00
https://doi.org/10.1145/3375395.3387650

Acknowledgments. The work is supported by European Research Council grant ERC-2014-CoG 647557.

1 INTRODUCTION

The streaming model of computation is a useful abstraction to understand the complexity of working with large volumes of data, too large to conveniently store. Efficient algorithms are known for many basic functions, such as finding frequent items, computing the number of distinct items, and measuring the empirical entropy of the data. Typically, in the streaming model we allow just one pass over the data and a small amount of memory, i.e., sublinear in the data size. While computing sums, averages, or counts is trivial with a constant memory, finding the median, quartiles, percentiles and their generalizations, quantiles, presents a challenging task. Indeed, four decades ago, Munro and Paterson [17] showed that finding the exact median in p passes over the data requires $\Omega(N^{1/p})$ memory, where N is the number of items in the stream. They also provide a p-pass algorithm for selecting the k-th smallest item in space $N^{1/p} \cdot \text{polylog}(N)$, and a $\text{polylog}(N)$-pass algorithm running in space $\text{polylog}(N)$.

Thus, either large space, or a large number of passes is necessary for finding the exact median. For this reason, subsequent research has mostly been concerned with the computation of approximate quantiles, which are often sufficient for applications. Namely, for a given precision guarantee $\varepsilon > 0$ and a query $\phi \in [0, 1]$, instead of finding the ϕ-quantile, i.e., the $\lfloor \phi N \rfloor$-th smallest item, we allow the algorithm to return a ϕ'-quantile for $\phi' \in [\phi - \varepsilon, \phi + \varepsilon]$. In other words, when queried for the k-th smallest item (where $k = \lfloor \phi N \rfloor$), the algorithm may return the k'-th smallest item for some $k' \in [k - \varepsilon N, k + \varepsilon N]$. Such an item is called an *ε-approximate ϕ-quantile*.

More precisely, we are interested in a data structure, called an *ε-approximate quantile summary*, that processes a stream of items from a totally ordered universe in a single pass. Then, it returns an ε-approximate ϕ-quantile for any query $\phi \in [0, 1]$. We optimize the space used by the quantile summary, measured in words, where a word can store any item or an integer with $O(\log N)$ bits (that is, counters, pointers, etc.).[1] We do not assume that items are drawn from a particular distribution, but rather focus on data independent solutions with worst-case guarantees. Quantile summaries are a valuable tool, since they immediately provide solutions for a range of related problems: estimating the cumulative distribution function; answering rank queries; constructing equi-depth histograms (where the number of items in each bucket must be approximately

[1] Hence, if instead b bits are needed to store an item, then the space complexity in bits is at most $\max(b, O(\log N))$ times the space complexity in words.

equal); performing Kolmogorov-Smirnov statistical tests [12]; and balancing parallel computations [19].

Note that offline, with random access to the whole data set, we can design an ε-approximate quantile summary with storage cost just $\lceil \frac{1}{2\varepsilon} \rceil$. We simply select the ε-quantile, the 3ε-quantile, the 5ε-quantile, and so on, and arrange them in a sorted array. Queries can be answered by returning the ϕ-quantile of this summary data set. Moreover, this is optimal, since there cannot be an interval $I \subset [0, 1]$ of size more than 2ε such that there is no ϕ-quantile for any $\phi \in I$ in the quantile summary.

Building on the work of Munro and Paterson [17], Manku, Rajagopalan, and Lindsay [14] designed a (streaming) quantile summary which uses space $O(\frac{1}{\varepsilon} \cdot \log^2 \varepsilon N)$, although it relies on the advance knowledge of the stream length N. Then, shaving off one log factor, Greenwald and Khanna [6] gave an ε-approximate quantile summary, which needs just $O(\frac{1}{\varepsilon} \cdot \log \varepsilon N)$ words and does not require any advance information about the stream. Both of these deterministic algorithms work for any universe with a total ordering as they just need to do comparisons of the items. We call such an algorithm *comparison-based*.

The question of whether one can design a 1-pass deterministic algorithm that runs in a constant space for a constant ε has been open for a long time, as highlighted by the first author in 2006 [1]. Following the above discussion, there is a trivial lower bound of $\Omega(\frac{1}{\varepsilon})$ that holds even offline. This was the best known lower bound until 2010 when Hung and Ting [10] proved that a deterministic comparison-based algorithm needs space $\Omega(\frac{1}{\varepsilon} \cdot \log \frac{1}{\varepsilon})$.

We significantly improve upon that result by showing that any deterministic comparison-based data structure providing ε-approximate quantiles needs to use $\Omega(\frac{1}{\varepsilon} \cdot \log \varepsilon N)$ memory on the worst-case input stream. Our lower bound thus matches the Greenwald and Khanna's result, up to a constant factor, and in particular, it rules out an algorithm running in space $f(\varepsilon) \cdot o(\log N)$, for any function f that does not depend on N. It also follows that a comparison-based data structure with $o(\frac{1}{\varepsilon} \cdot \log \varepsilon N)$ memory must fail to provide a ϕ-quantile for some $\phi \in [0, 1]$. Using a standard reduction (appending more items to the end of the stream), this implies that there is no deterministic comparison-based streaming algorithm that returns an ε-approximate median and uses $o(\frac{1}{\varepsilon} \cdot \log \varepsilon N)$ memory. Applying a different reduction, this yields a lower bound of $\Omega(\frac{1}{\varepsilon} \cdot \log \log \frac{1}{\delta})$ for any randomized comparison-based algorithm. We refer to Section 6 for a discussion of this and other corollaries of our result.

1.1 Overview and Comparison to Prior Bounds

Let $\mathcal{D}$ be a deterministic comparison-based quantile summary. From a high-level point of view, we prove the space lower bound for $\mathcal{D}$ by constructing two streams π and ϱ satisfying two opposing constraints: On one hand, the behavior of $\mathcal{D}$ on these streams is the same, implying that the memory states after processing π and ϱ are the same, up to an order-preserving renaming of the stored items. For this reason, π and ϱ are called *indistinguishable*. On the other hand, the adversary introduces as much uncertainty as possible. Namely, it makes the difference between the rank of a stored item with respect to (w.r.t.) π and the rank of the next stored item w.r.t. ϱ as large as possible, where the rank of an item w.r.t. stream σ is its position in the ordering of σ. If this difference, which we call the "gap", is too large, then $\mathcal{D}$ fails to provide an ε-approximate ϕ-quantile for some $\phi \in [0, 1]$. The crucial part of our lower bound proof is to construct the two streams in a way that yields a good trade-off between the number of items stored by the algorithm and the largest gap introduced.

While the previous lower bound of $\Omega(\frac{1}{\varepsilon} \cdot \log \frac{1}{\varepsilon})$ [10] is in the same computational model, and also works by creating indistinguishable streams with as much uncertainty as possible, our approach is substantially different. Mainly, the construction by Hung and Ting [10] is inherently sequential as it works in $m \approx \frac{1}{\varepsilon} \log \frac{1}{\varepsilon}$ iterations and appends $O(m)$ items in each iteration to the streams constructed (and moreover, up to $O(m)$ new streams are created from each former stream in each iteration). Thus, their construction produces (a large number of) indistinguishable streams of length $\Theta\left(\left(\frac{1}{\varepsilon} \log \frac{1}{\varepsilon}\right)^2\right)$. Furthermore, having the number of iterations equal to the number of items appended during each iteration (up to a constant factor) is crucial for the analysis in [10].

In contrast, our construction is naturally specified in a recursive way, and it produces just two indistinguishable streams of length N for any $N = \Omega(\frac{1}{\varepsilon})$. For $N \approx \left(\frac{1}{\varepsilon}\right)^2$, our lower bound of $\Omega(\frac{1}{\varepsilon} \cdot \log \varepsilon N)$ implies the previous one of $\Omega(\frac{1}{\varepsilon} \cdot \log \frac{1}{\varepsilon})$, and hence for higher N, our lower bound is strictly stronger than the previous one.

The value in using a recursive construction is as follows: The construction produces two indistinguishable streams of length $\frac{1}{\varepsilon} \cdot 2^k$ for an integer $k \geq 1$, and we need to prove that the quantile summary $\mathcal{D}$ must store at least $c \cdot \frac{1}{\varepsilon} \cdot k$ items while processing one of these streams, for a constant $c > 0$. The first half of the streams is constructed recursively, so $\mathcal{D}$ needs to store at least $c \cdot \frac{1}{\varepsilon} \cdot (k-1)$ items while processing the first half of either of these two streams (using an induction on k). If it already stores at least $c \cdot \frac{1}{\varepsilon} \cdot k$ items on the first half, then we are done. Otherwise, our inductive argument yields that there must a substantial uncertainty introduced while processing the first half, which we use in the recursive construction of the second half of the streams. Then our aim will be to show that, while processing the second half, $\mathcal{D}$ needs to store $c \cdot \frac{1}{\varepsilon} \cdot (k-1)$ items from the second half, by induction, and $c \cdot \frac{1}{\varepsilon}$ items from the first half, by a simple bound. Hence, it stores $c \cdot \frac{1}{\varepsilon} \cdot k$ items overall. However, using the inductive argument on the second half brings some technical difficulties, since the streams already contain items from the first half. Our analysis shows a space lower bound, called the "space-gap inequality", that depends on the uncertainty introduced on a particular part of the stream, and this inequality is amenable to a proof by induction.

Organization of the paper. In Section 2, we start by describing the formal computational model in which our lower bound holds and formally stating our result. In Section 3, we introduce indistinguishable streams, and in Section 4 we describe our construction. Then, in Section 5 we inductively prove the crucial inequality between the space and the largest gap (the uncertainty), which implies the lower bound. Finally, in Section 6 we give corollaries of the construction and discuss related open problems.

1.2 Related Work

The Greenwald-Khanna algorithm [6] is generally regarded as the best deterministic quantile summary. The space bound of $O(\frac{1}{\varepsilon} \cdot \log \varepsilon N)$ follows from a somewhat involved proof, and it has been questioned whether this approach could be simplified or improved. Our work answers this second question in the negative. For a known universe U of bounded size, Shrivastava *et al.* [18] designed a quantile summary q-digest using $O(\frac{1}{\varepsilon} \cdot \log |U|)$ words. Note that their algorithm is not comparison-based and so the result is incomparable to the upper bound of $O(\frac{1}{\varepsilon} \cdot \log \varepsilon N)$. We are not aware of any lower bound which holds for a known universe of bounded size, apart from the trivial bound $\Omega(\frac{1}{\varepsilon})$.

If we tolerate randomization and relax the requirement for worst-case error guarantees, it is possible to design quantile summaries with space close to $\frac{1}{\varepsilon}$. After a sequence of improvements [2, 5, 13, 15], Karnin, Lang, and Liberty [11] designed a randomized comparison-based quantile summary with space bounded by $O(\frac{1}{\varepsilon} \cdot \log \log \frac{1}{\varepsilon\delta})$, where δ is the probability of not returning an ε-approximate ϕ-quantile for some ϕ. They also provide a reduction to transform the deterministic $\Omega(\frac{1}{\varepsilon} \cdot \log \frac{1}{\varepsilon})$ lower bound into a randomized lower bound of $\Omega(\frac{1}{\varepsilon} \cdot \log \log \frac{1}{\delta})$ for $\delta < 1/N!$, implying optimality of their approach in the comparison-based model for an exponentially small δ. We discuss further how the deterministic and randomized lower bounds relate in Section 6.

Luo *et al.* [13] compared quantile summaries experimentally and also provided a simple randomized algorithm with a good practical performance. This paper studies not only streaming algorithms for insertion-only streams (i.e., the cash register model), but also for turnstile streams, in which items may depart. Note that any algorithm for turnstile streams inherently relies on the bounded size of the universe. We refer the interested reader to the survey of Greenwald and Khanna [7] for a description of both deterministic and randomized algorithms, together with algorithms for turnstile streams, the sliding window model, and distributed algorithms.

Other results arise when relaxing the requirement for correctness under adversarial order to assuming that the input arrives in a random order. For random-order streams, Guha and McGregor [8] studied algorithms for exact and approximate selection of quantiles. Among other things, they gave an algorithm for finding the exact ϕ-quantile in space polylog(N) using $O(\log \log N)$ passes over a random-order stream, while with polylog(N) memory we need to do $\Omega(\log N / \log \log N)$ passes on the worst-case stream. The Shifting Sands algorithm [16] reduces the magnitude of the error from $O(n^{1/2})$ to $O(n^{1/3})$. Since our lower bound relies on carefully constructing an adversarial input sequence, it does not apply to this random order model.

2 COMPUTATIONAL MODEL

We present our lower bounds in a comparison-based model of computation, in line with prior work, most notably that of Hung and Ting [10]. We assume that the items forming the input stream are drawn from a totally ordered universe U, about which the algorithm has no further information. The only allowed operations on items are to perform an equality test or a comparison of two given items. This restriction specifically rules out manipulations which try to combine multiple items into a single storage location, or replace a group of items with an "average" representative. We assume that the universe is unbounded and continuous in the sense that any non-empty open interval contains an unbounded number of items. This fact is relied on in our proof to be able to draw new elements falling between any previously observed pair. An example of such a universe is a large enough set of long incompressible strings, ordered lexicographically (where the continuous assumption may be achieved by making the strings even longer).

Let $\mathcal{D}$ be a deterministic data structure for processing a stream of items, i.e., a sequence of items arriving one by one. We make the following assumptions about the memory contents of $\mathcal{D}$. The memory used by $\mathcal{D}$ will contain some items from the stream, each considered to occupy one memory cell, and some other information which could include lower and upper bounds on the ranks of stored items, counters, etc. However, we assume that the memory does not contain the result of any operation applied on any $k \geq 1$ items from the stream, apart from a comparison and the equality test (as other operations are prohibited by our model). Thus, we can partition the *memory state* into a pair $M = (I, G)$, where I is the *item array* for storing items from the input, indexed from 1, and there are no items stored in the *general memory* G.

We give our lower bound on the memory size only in terms of $|I|$, the number of items stored, and ignore the size of G. For simplicity, we assume without loss of generality that the contents of I are sorted non-decreasingly, i.e., $I[1] \leq I[2] \leq \cdots$. If this were not case, we could equivalently apply an in-place sorting algorithm after processing each item, while the information potentially encoded in the former ordering of I can be retained in G whose size we do not measure. Moreover, we assume that $|I|$ never decreases over time, i.e., once some memory is allocated to the item array, it is not released later (otherwise, we would need to take the maximum size of $|I|$ during the computation of $\mathcal{D}$). Finally, we can assume that the minimum and maximum elements of the input stream are always maintained, with at most a constant additional storage space.

Summarizing, we have the following definition.

Definition 2.1. We say that a quantile summary $\mathcal{D}$ is *comparison-based* if the following holds:

(i) $\mathcal{D}$ does not perform any operation on items from the stream, apart from a comparison and the equality test.

(ii) The memory of $\mathcal{D}$ is divided into the *item array* I, which stores only items that have already occurred in the stream (sorted non-decreasingly), and *general memory* G, which does not contain any item identifier. Furthermore, once an item is removed from I, it cannot be added back to I, unless it appears in the stream again.

(iii) Given the i-th item a_i from the input stream, the computation of $\mathcal{D}$ is determined solely by the results of comparisons between a_i and $I[j]$, for $j = 1, \ldots, |I|$, the number $|I|$ of items stored, and the contents of the general memory G.

(iv) Given a quantile query $0 \leq \phi \leq 1$, its computation is determined solely by the number of items stored ($|I|$), and the contents of the general memory G. Moreover, $\mathcal{D}$ can only return one of the items stored in I.

Note that quantile summaries satisfying Definition 2.1 include the Greenwald-Khanna algorithm [6] as well as many other deterministic [7, 14, 17] and randomized quantile summaries [2, 5,

11, 13, 15]. On the other hand, the q-digest structure [18] is not comparison-based, since it relies on building a binary tree over U and since it can actually return an item that did not occur in the stream, neither of which is allowed by Definition 2.1. Thus, our lower bound does not apply to this algorithm and indeed, for $N \gg |U|$, its space requirement of $O(\frac{1}{\varepsilon} \cdot \log |U|)$ words may be substantially smaller than $\Omega(\frac{1}{\varepsilon} \cdot \log \varepsilon N)$.

We are now ready to state our main result formally.

Theorem 2.2. *For any $0 < \varepsilon < \frac{1}{16}$, there is no deterministic comparison-based ε-approximate quantile summary which stores $o(\frac{1}{\varepsilon} \cdot \log \varepsilon N)$ items on any input stream of length N.*

Fix the approximation guarantee $0 < \varepsilon < \frac{1}{16}$ and assume for simplicity that $\frac{1}{\varepsilon}$ is an integer. Let $\mathcal{D}$ be a fixed deterministic comparison-based ε-approximate quantile summary. We show that for any integer $k \geq 1$, data structure $\mathcal{D}$ needs to store at least $\Omega(\frac{1}{\varepsilon} \cdot k)$ items from some input stream of length $N_k := \frac{1}{\varepsilon} \cdot 2^k$ (thus, we have $\log_2 \varepsilon N_k = k$).

Notation and conventions. We assume that $\mathcal{D}$ starts with an empty memory state $M_\emptyset = (I_\emptyset, G_\emptyset)$ with $|I_\emptyset| = 0$. For an item a, let $\mathcal{D}(M, a)$ be the resulting memory state after processing item a if the memory state was M before processing a. Moreover, for a stream $\sigma = a_1, \ldots, a_N$, let $\mathcal{D}(M, \sigma) = \mathcal{D}(\ldots \mathcal{D}(\mathcal{D}(M, a_1), a_2), \ldots, a_N)$ be the memory state after processing stream σ. For brevity, we use $(I_\sigma, G_\sigma) = \mathcal{D}(M_\emptyset, \sigma)$, or just I_σ for the item array after processing stream σ.

When referring to the order of a set of items, we always mean the non-decreasing order. For an item a in stream σ, let $\mathrm{rank}_\sigma(a)$ be the rank of a in the order of σ, i.e., the position of a in the ordering of σ. In our construction, all items in each of the streams will be distinct, thus $\mathrm{rank}_\sigma(a)$ is well-defined and equal to one more than the number of items that are strictly smaller than a.

3 INDISTINGUISHABLE STREAMS

We start by defining an equivalence of memory states of the fixed summary $\mathcal{D}$, which captures their equality up to renaming stored items. Then, we give the definition of indistinguishable streams.

Definition 3.1. Two memory states (I_1, G_1) and (I_2, G_2) are said to be *equivalent* if (i) $|I_1| = |I_2|$, i.e., the number of items stored is the same, and (ii) $G_1 = G_2$.

Definition 3.2. We say that two streams $\pi = a_1 a_2 \ldots a_N$ and $\varrho = b_1 b_2 \ldots b_N$ of length N are *indistinguishable* for $\mathcal{D}$ if (1) the final memory states (I_π, G_π) and (I_ϱ, G_ϱ) are equivalent, and (2) for any $1 \leq i \leq |I_\pi| = |I_\varrho|$, there exists $1 \leq j \leq N$ such that both $I_\pi[i] = a_j$ and $I_\varrho[i] = b_j$.

We remark that condition (2) is implied by (1) if the positions of stored items in the stream are retained in the general memory, but we make this property explicit as we shall use it later. In the following, let π and ϱ be two indistinguishable streams with N items. Note that, after $\mathcal{D}$ processes one of π and ϱ and receives a quantile query $0 \leq \phi \leq 1$, $\mathcal{D}$ must return the i-th item of array I for some i, regardless of whether the stream was π or ϱ. This follows, since $\mathcal{D}$ can make its decisions based on the values in G, which are identical in both cases, and operations on values in I, which are indistinguishable under the comparison-based model.

For any $k \geq 1$, our general approach is to recursively construct two streams π_k and ϱ_k of length N_k that satisfy two constraints set in opposition to each other: They are indistinguishable for $\mathcal{D}$, but at the same time, for some j, the rank of $I_\pi[j]$ in stream π and the rank of $I_\varrho[j+1]$ in stream ϱ are as different as possible — we call this difference the "gap". The latter constraint is captured by the following definition.

Definition 3.3. We define the *largest gap* between indistinguishable streams π and ϱ (for $\mathcal{D}$) as

$$\mathrm{gap}(\pi, \varrho) = \max_{1 \leq i < |I_\pi|} \max\big(\mathrm{rank}_\pi(I_\pi[i+1]) - \mathrm{rank}_\varrho(I_\varrho[i]), \\ \mathrm{rank}_\varrho(I_\varrho[i+1]) - \mathrm{rank}_\pi(I_\pi[i]) \big).$$

As we assume that I is sorted, $I_\pi[i+1]$ is the next stored item after $I_\pi[i]$ in the ordering of I_π. In the construction in Section 4, we will also ensure that $\mathrm{rank}_\pi(I_\pi[i]) \leq \mathrm{rank}_\varrho(I_\varrho[i])$ for any $1 \leq i \leq |I_\pi|$. Hence, we can simplify to

$$\mathrm{gap}(\pi, \varrho) = \max_i \mathrm{rank}_\varrho(I_\varrho[i+1]) - \mathrm{rank}_\pi(I_\pi[i]).$$

We also have that $\mathrm{gap}(\pi, \varrho) \geq \mathrm{gap}(\pi, \pi)$, which follows, since for any i it holds by construction that

$$\mathrm{rank}_\varrho(I_\varrho[i+1]) - \mathrm{rank}_\pi(I_\pi[i]) \geq \mathrm{rank}_\pi(I_\pi[i+1]) - \mathrm{rank}_\pi(I_\pi[i]).$$

Lemma 3.4. *If $\mathcal{D}$ is an ε-approximate quantile summary, then $\mathrm{gap}(\pi, \varrho) \leq 2\varepsilon N$.*

Proof. Suppose that $\mathrm{gap}(\pi, \varrho) > 2\varepsilon N$. We show that $\mathcal{D}$ fails to provide an ε-approximate ϕ-quantile for some $0 \leq \phi \leq 1$, which is a contradiction. Namely, because $\mathrm{gap}(\pi, \varrho) > 2\varepsilon N$, there is $1 \leq i < |I_\pi| = |I_\varrho|$ such that $\mathrm{rank}_\varrho(I_\varrho[i+1]) - \mathrm{rank}_\pi(I_\pi[i]) > 2\varepsilon N$. Let ϕ be such that

$$\phi \cdot N = \frac{1}{2}\big(\mathrm{rank}_\varrho(I_\varrho[i+1]) + \mathrm{rank}_\pi(I_\pi[i]) \big),$$

i.e., $\phi \cdot N$ is in the middle of the "gap". Since streams π and ϱ are indistinguishable and $\mathcal{D}$ is comparison-based, given query ϕ, $\mathcal{D}$ must return the j-th item of item array I for some j, regardless of whether the stream is π or ϱ. Observe that if $j \leq i$ and the input stream is π, item $I_\pi[j]$ does not meet the requirements to be an ε-approximate ϕ-quantile of items in π. Otherwise, when $j > i$, then item $I_\varrho[j]$ is not an ε-approximate ϕ-quantile of stream ϱ. In either case, we get a contradiction. □

As the minimum and maximum elements of stream π are in I_π, it holds that $\mathrm{gap}(\pi, \pi) \geq N/|I_\pi|$, thus the number of stored items is at least $N/\mathrm{gap}(\pi, \pi) \geq N/\mathrm{gap}(\pi, \varrho) \geq \frac{1}{2\varepsilon}$, where the last inequality is by Lemma 3.4. This gives an initial lower bound of $\Omega(\frac{1}{\varepsilon})$ space. Our construction of adversarial inputs for $\mathcal{D}$ in the next section increases this bound.

4 RECURSIVE CONSTRUCTION OF INDISTINGUISHABLE STREAMS

4.1 Intuition

We will define our construction of the two streams π and ϱ using a recursive adversarial procedure for generating items into the two streams. This procedure tries to make the gap as large as possible, but ensures that they are indistinguishable. It helps to consider the

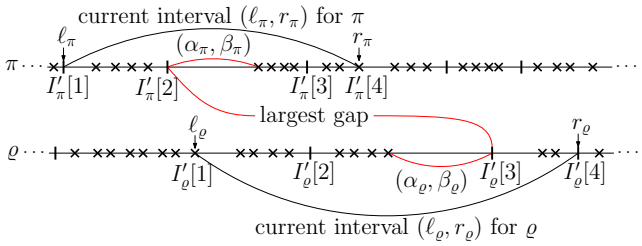


Figure 1: An illustration of the largest gap computation.

recursion tree. This tree is a full binary tree with k levels, with the root at level 1 and thus with 2^{k-1} leaves at level k. In each leaf, $2/\varepsilon$ items are appended to the stream, while the adversary generates no items in internal (i.e., non-leaf) nodes. The construction performs the in-order traversal of the recursion tree.

One of the key concepts needed is the maintenance of two open intervals during the construction, one for stream π, denoted (ℓ_π, r_π), and the other for stream ϱ, denoted $(\ell_\varrho, r_\varrho)$. Initially, these intervals cover the whole universe, but they are refined in each internal node of the recursion tree. More precisely, consider the execution in an internal node v at level i of the recursion tree. We first execute the left subtree, which generates $\frac{1}{\varepsilon} \cdot 2^{i-1}$ items into the streams inside the current intervals. We then identify the largest gap inside the current intervals w.r.t. item arrays of $\mathcal{D}$ after processing streams π and ϱ (more precisely, after $\mathcal{D}$ has completed processing the prefixes of π and ϱ constructed so far). Having the largest gap, we identify new open intervals for π and ϱ in "extreme regions" of this gap, so that they do not contain any item so far. We explain this subroutine in greater detail when describing procedure RefineIntervals. We choose these intervals so that indistinguishability of the streams is preserved, while the rank difference between the two streams (the uncertainty) is maximized. The execution of the procedure in node v ends by executing the right subtree of v, which generates a further $\frac{1}{\varepsilon} \cdot 2^{i-1}$ items into the new, refined intervals of the two streams. Recall that we consider the universe of items to be continuous, namely, that we can always generate sufficiently many items within both of the new intervals.

4.2 Notation

For an item a in stream σ, let $\text{next}(\sigma, a)$ be the next item in the ordering of σ, i.e., the smallest item in σ that is larger than a (we never invoke $\text{next}(\sigma, a)$ when a is the largest item in σ). Similarly, for an item b in stream σ, let $\text{prev}(\sigma, b)$ be the previous item in the ordering of σ (left undefined for the smallest item in σ). Note that $\text{next}(\sigma, a)$ or $\text{prev}(\sigma, b)$ may well *not* be stored by $\mathcal{D}$.

For an interval (ℓ, r) of items and an array I of items, we use $I^{(\ell,r)}$ to denote the restriction of I to (ℓ, r), enclosed by ℓ and r. That is, $I^{(\ell,r)}$ is the array of items $\ell, I[i], I[i+1], \ldots, I[j], r$, where i and j are the minimal and maximal indexes of an item in I that falls within the interval (ℓ, r), respectively. Items in $I^{(\ell,r)}$ are taken to be sorted and indexed from 1. Recall also that by our convention, I_σ is the item array after processing some stream σ.

Pseudocode 1 Adversarial procedure RefineIntervals

Input: Streams π and ϱ and intervals (ℓ_π, r_π) and $(\ell_\varrho, r_\varrho)$ of items such that:
(i) π and ϱ are indistinguishable, and
(ii) only the last $N' \geq 2$ items from π and ϱ are from intervals (ℓ_π, r_π) and $(\ell_\varrho, r_\varrho)$, respectively
Output: Intervals $(\alpha_\pi, \beta_\pi) \subset (\ell_\pi, r_\pi)$ and $(\alpha_\varrho, \beta_\varrho) \subset (\ell_\varrho, r_\varrho)$
1: $I'_\pi \leftarrow I_\pi^{(\ell_\pi, r_\pi)}$ and $I'_\varrho \leftarrow I_\varrho^{(\ell_\varrho, r_\varrho)}$
2: $i \leftarrow \arg\max_{1 \leq i < |I'_\varrho|} \text{rank}_\varrho(I'_\varrho[i+1]) - \text{rank}_\pi(I'_\pi[i])$
▷ Position of the largest gap in intervals (ℓ_π, r_π) and $(\ell_\varrho, r_\varrho)$
3: $(\alpha_\pi, \beta_\pi) \leftarrow \big(I'_\pi[i], \text{next}(\pi, I'_\pi[i])\big)$ ▷ New interval for π
4: $(\alpha_\varrho, \beta_\varrho) \leftarrow \big(\text{prev}(\varrho, I'_\varrho[i+1]), I'_\varrho[i+1]\big)$ ▷ New interval for ϱ
5: **return** (α_π, β_π) and $(\alpha_\varrho, \beta_\varrho)$

4.3 Procedure RefineIntervals

We next describe our procedure to find the largest gap and refine the intervals, defined in Pseudocode 1. It takes as input indistinguishable streams π and σ and two open intervals (ℓ_π, r_π) and $(\ell_\varrho, r_\varrho)$ of the universe, such that intervals (ℓ_π, r_π) and $(\ell_\varrho, r_\varrho)$ contain only the last N' items from π and ϱ, respectively, for some $N' \geq 2$. Note that I'_π and I'_ϱ are the item arrays of $\mathcal{D}$ for π and ϱ restricted to the intervals (ℓ_π, r_π) and $(\ell_\varrho, r_\varrho)$, respectively, as defined above. In these restricted arrays we find the largest gap (in line 2), which is determined by the largest rank difference of consecutive items in the two arrays. Finally, in lines 3 and 4, we define new, refined intervals in the extreme regions of the gap. To be precise, the new open interval for π is between item $I'_\pi[i]$ (whose rank is used to determine the largest gap) and the next item after $I'_\pi[i]$ in the ordering of π, i.e., $\text{next}(\pi, I'_\pi[i])$. The new open interval for ϱ is defined in a similar way: It is between item $I'_\varrho[i+1]$ (used to determine the largest gap) and the item that precedes it in the ordering of ϱ, i.e., $\text{prev}(\varrho, I'_\varrho[i+1])$.

In Figure 1 we give an illustration. In this figure, the items in the streams are real numbers and we depict them on the real line, the top one for π and the bottom one for ϱ. Each item is represented either by a short line segment if it is stored in the item array, or by a cross otherwise (indicating that it has been "forgotten" by $\mathcal{D}$). The procedure looks for the largest gap only within the current intervals (ℓ_π, r_π) and $(\ell_\varrho, r_\varrho)$. The ranks of items in the restricted item arrays (i.e., disregarding items outside the current intervals) can be verified to be 1, 6, 11, and 14 w.r.t. both streams. (Note that r_π is the last item in the restricted item array I'_π, even though it was discarded from the whole item array I_π by the algorithm, and similarly for ℓ_ϱ and I'_ϱ.) Thus the largest gap size is 5 items, and is found between the second item in the restricted item array I'_π and the third item in I'_ϱ, as highlighted in the figure. In this example, there is another, equal sized, gap between the first and second item in these arrays. Ties can be broken arbitrarily. The new intervals in the extreme regions of the largest gap are depicted as well. □

We claim that in the RefineIntervals procedure $|I'_\pi| = |I'_\varrho|$, which implies that the largest gap in line 2 is well-defined. Let $\pi = a_1 \ldots a_N$ and $\varrho = b_1 \ldots b_N$ be the items in streams π and ϱ, respectively. Since streams π and ϱ are indistinguishable, condition (2) in Definition 3.2 implies that for any $1 \leq i \leq |I_\pi| = |I_\varrho|$

Pseudocode 2 Adversarial procedure AdvStrategy

Input: Integer $k \geq 1$, streams π and ϱ, and intervals (ℓ_π, r_π) and $(\ell_\varrho, r_\varrho)$ of items such that:
(i) π and ϱ are indistinguishable,
(ii) π contains no item from (ℓ_π, r_π) and ϱ contains no item from $(\ell_\varrho, r_\varrho)$, and
(iii) for any $a \in (\ell_\pi, r_\pi)$ and $b \in (\ell_\varrho, r_\varrho)$, it holds that $\min\{i | a \leq I_\pi[i]\} = \min\{i | b \leq I_\varrho[i]\}$[2]

Output: Streams $\pi'' = \pi\pi_k$ and $\varrho'' = \varrho\varrho_k$, where π_k and ϱ_k are substreams with $\frac{1}{\varepsilon} \cdot 2^k$ items from (ℓ_π, r_π) and $(\ell_\varrho, r_\varrho)$, respectively

1: **if** k = 1 **then** ▷ Leaf node of the recursion tree
2: $\pi'' \leftarrow$ stream π followed by $2/\varepsilon$ items from interval (ℓ_π, r_π), in order
3: $\varrho'' \leftarrow$ stream ϱ followed by $2/\varepsilon$ items from interval $(\ell_\varrho, r_\varrho)$, in order
4: **return** streams π'' and ϱ''
5: **else** ▷ Internal node of the recursion tree
6: $(\pi', \varrho') \leftarrow \text{AdvStrategy}\big(k-1, \pi, \varrho, (\ell_\pi, r_\pi), (\ell_\varrho, r_\varrho)\big)$
7: $(\alpha_\pi, \beta_\pi), (\alpha_\varrho, \beta_\varrho) \leftarrow \text{RefineIntervals}(\pi', \varrho', (\ell_\pi, r_\pi), (\ell_\varrho, r_\varrho))$
8: **return** $(\pi'', \varrho'') \leftarrow \text{AdvStrategy}\big(k-1, \pi', \varrho', (\alpha_\pi, \beta_\pi), (\alpha_\varrho, \beta_\varrho)\big)\big)$

(where I_π and I_ϱ are the full item arrays), there exists $1 \leq j \leq N$ such that both $I_\pi[i] = a_j$ and $I_\varrho[i] = b_j$. As only the last N' items of π and of σ are from intervals (ℓ_π, r_π) and $(\ell_\varrho, r_\varrho)$, respectively, we obtain that the restricted item arrays $I'_\pi = I_\pi^{(\ell_\pi, r_\pi)}$ and $I'_\varrho = I_\varrho^{(\ell_\varrho, r_\varrho)}$ must have the same size, proving the claim.

Finally, we show two properties that will be useful later and follow directly from the definition of the new intervals.

Observation 1. *For intervals (α_π, β_π) and $(\alpha_\varrho, \beta_\varrho)$ returned by RefineIntervals$(\pi, \varrho, (\ell_\pi, r_\pi), (\ell_\varrho, r_\varrho))$, it holds that*
(i) π contains no item in the interval (α_π, β_π) and ϱ contains no item in the interval $(\alpha_\varrho, \beta_\varrho)$; and
(ii) for any $a \in (\alpha_\pi, \beta_\pi)$ and $b \in (\alpha_\varrho, \beta_\varrho)$ we have that $\min\{i | a \leq I_\pi[i]\} = \min\{i | b \leq I_\varrho[i]\}$.

4.4 Recursive Adversarial Strategy

Pseudocode 2 gives the formal description of the recursive adversarial strategy. The procedure AdvStrategy takes as input the level of recursion k and the indistinguishable streams π and ϱ constructed so far. It also takes two open intervals (ℓ_π, r_π) and $(\ell_\varrho, r_\varrho)$ of the universe such that so far there is no item from interval (ℓ_π, r_π) in stream π and similarly, ϱ contains no item from $(\ell_\varrho, r_\varrho)$.

The initial call of the strategy for some integer k is AdvStrategy $(k, \emptyset, \emptyset, (-\infty, \infty), (-\infty, \infty))$, where $\emptyset$ stands for the empty stream and $-\infty$ and ∞ represent the minimum and maximum items in U, respectively. Note that the assumptions on the input for the initial call are satisfied. The strategy for $k = 1$ is trivial: We just append $2/\varepsilon$ arbitrary items from (ℓ_π, r_π) to π and any $2/\varepsilon$ items from $(\ell_\varrho, r_\varrho)$ to ϱ, in the same order for both streams. For $k > 1$, we first use AdvStrategy recursively for level $k-1$. Then, we apply procedure RefineIntervals on the streams constructed after the first recursive call, and we get two new intervals on the extreme regions of the largest gap inside the current intervals. Finally, we use AdvStrategy recursively for $k-1$ in these new intervals. Below, we prove that the assumptions on input for these two recursive calls and for RefineIntervals are satisfied.

4.5 Example of the Adversarial Strategy

We now give an example of the construction with $k = 3$ in Figure 2. The universe is $U = \mathfrak{R}$, which we depict by the real line. For simplicity, we set $\varepsilon = \frac{1}{6}$ (although recall that we require $\varepsilon < \frac{1}{16}$ for our analysis in Section 5 to hold).

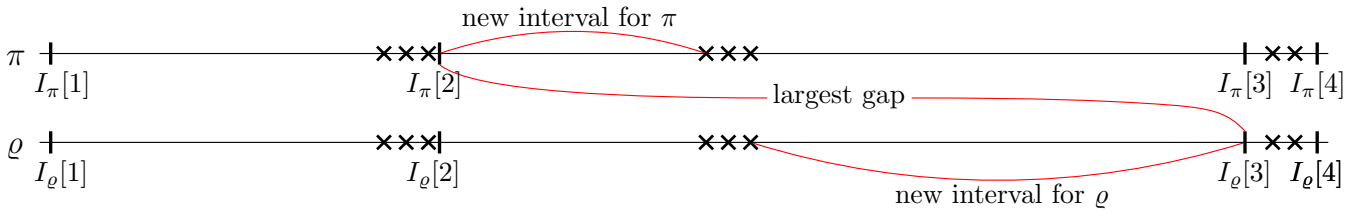
The adversarial construction starts by calling AdvStrategy $(3, \emptyset, \emptyset, (-\infty, \infty), (-\infty, \infty))$. The procedure then recursively calls itself twice and in the base case $k = 1$, the two streams π and ϱ are initialized by $\frac{2}{\varepsilon} = 12$ items (we can assume the same items are added to the two streams). The quantile summary under consideration ($\mathcal{D}$) chooses to store some of them, but as $2\varepsilon N_1 = 4$, it cannot forget four consecutive items.

At this point, we are in the execution of AdvStrategy $(2, \emptyset, \emptyset, (-\infty, \infty), (-\infty, \infty))$, having finished the recursive call in line 6. Figure 2a shows the first 12 items sent to streams π and ϱ, depicted on the real line for each stream. A short line segment represents an item that is stored in item array I, while a cross depicts an item not stored by $\mathcal{D}$. Note that the largest gap is between the second and the third stored item, i.e., $i = 2$ in line 2 of AdvStrategy. This is because $\text{rank}_\pi(I_\pi[2]) = 5$ and $\text{rank}_\varrho(I_\varrho[3]) = 9$ (the gap of the same size is also between the first and the second item). Next, the procedure RefineIntervals finds the largest gap and identifies new intervals (α_π, β_π) and $(\alpha_\varrho, \beta_\varrho)$ for the second recursive call.

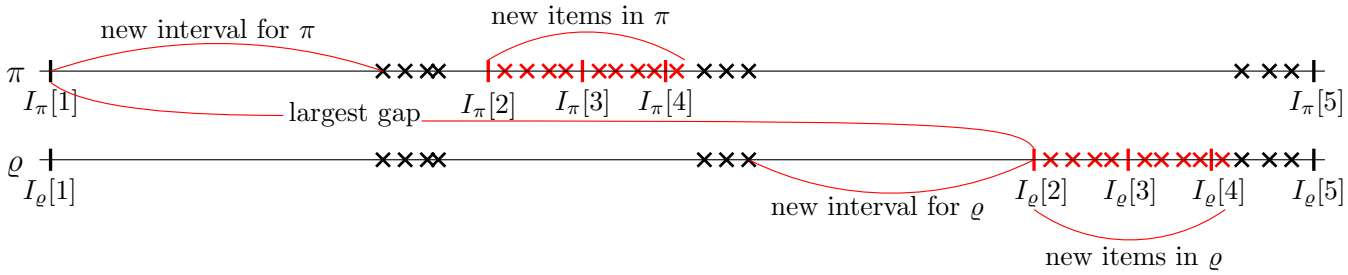
In the execution of AdvStrategy$(1, \pi', \varrho', (\alpha_\pi, \beta_\pi), (\alpha_\varrho, \beta_\varrho))$, there are $\frac{2}{\varepsilon} = 12$ items appended to the streams and the largest gap can be of size at most $2\varepsilon N_2 = 8$. In Figure 2b, we show the last 12 items, appended in the second leaf of the recursion tree, highlighted in red. Note that fewer of the first 12 items in the streams are now stored and that among the 12 newly appended items, the first, the sixth, and the eleventh are stored for both streams. The execution returns to the root node of the recursion tree and the adversary finds the largest gap together with new intervals. One of the largest gaps is now between the first and the second stored item (in this example, all gaps have the same size of 8).

The execution then goes to the third leaf, where 12 items are appended for the third time. Figure 2c illustrates this, with the most recent 12 items shown smaller and in blue. In the execution of AdvStrategy for $k = 2$, the largest gap is found — note that we look for it only in the current intervals, and that its size can be at most $2\varepsilon \cdot 3 \cdot \frac{2}{\varepsilon} = 12$ items. One of the two largest gaps is between the second and the third item in the restricted item arrays; these are also the second and the third item in the whole item arrays

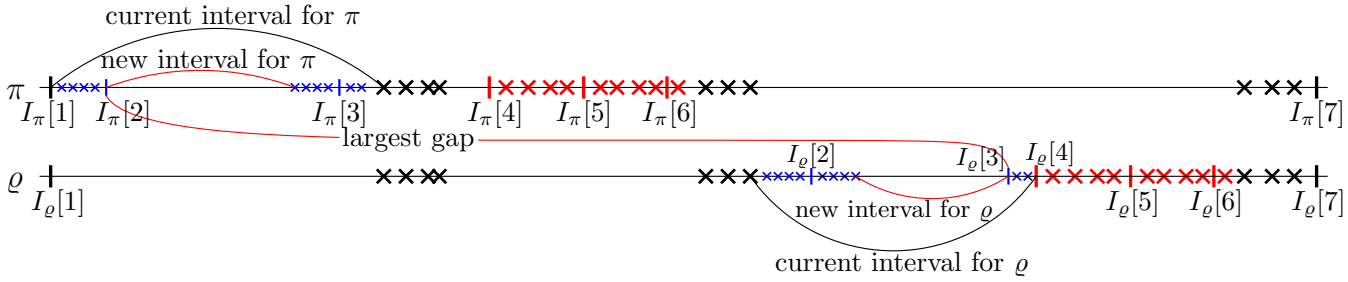
[2]The minimum over an empty set is defined arbitrarily to be ∞.



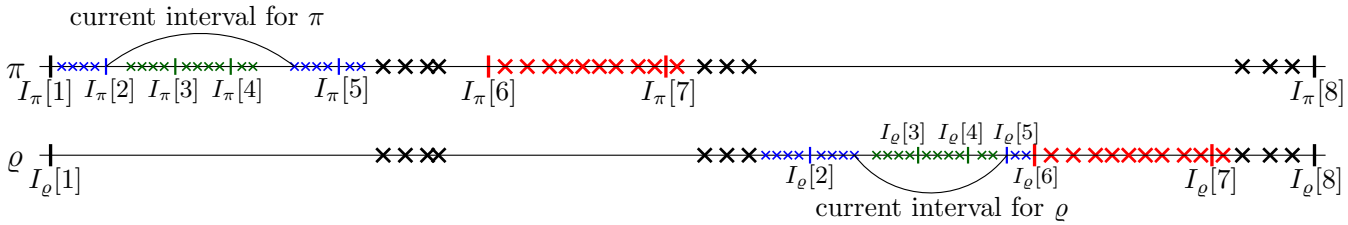
(a) After the first 12 items are sent to π and ϱ



(b) After 24 items are appended



(c) After 36 items are appended



(d) Streams π and σ with all $N_3 = 48$ items

Figure 2: An example of the construction of streams π and σ.

(the other gap of the same size is between the first and the second stored item). Again, new intervals are identified for the execution of the last leaf of the recursion tree.

Finally, the last 12 items are appended to the streams, which completes the construction. Figure 2d shows the final state, with these last 12 items added in green. The current intervals are with respect to the last leaf of the recursion tree. □

4.6 Properties of the Adversarial Strategy

We first give some observations. Note that the recursion tree of an execution of AdvStrategy(k) indeed has 2^{k-1} leaves which each correspond to calling the strategy for $k = 1$, and that the items are appended to streams only in the leaves, namely, $\frac{2}{\varepsilon}$ items to each stream in each leaf. It follows that the number of items appended is $N_k = \frac{1}{\varepsilon} \cdot 2^k$. Observe that for a general recursive call of AdvStrategy, the input streams π and ϱ may already contain some items. Also, the behavior of comparison-based quantile summary $\mathcal{D}$ may be different when processing items appended during the recursive call in line 6 and when processing items from the call in line 8. The reason is that the computation of $\mathcal{D}$ is also influenced by items outside the intervals, i.e., by items in streams π and ϱ that are from other branches of the recursion tree. We remark that items

in each of π'' and ϱ'' are distinct within the streams (but the two streams may share some items, which does not affect our analysis).

We now prove that the streams constructed are indistinguishable and that we do not violate any assumption on input for any recursive call. We use the following lemma derived from [10] (which is a simple consequence of the facts that $\mathcal{D}$ is comparison-based and the memory states (I_π, G_π) and (I_ϱ, G_ϱ) are equivalent).

Lemma 4.1 (Implied by Lemma 2 in [10]). *Suppose that streams π and ϱ are indistinguishable for $\mathcal{D}$ and let I_π and I_ϱ be the corresponding item arrays after processing π and ϱ, respectively. Let a, b be any two items such that $\min\{i|a \le I_\pi[i]\} = \min\{i|b \le I_\varrho[i]\}$. Then the streams πa and ϱb are indistinguishable.*

Lemma 4.2. *Consider an execution of AdvStrategy$(k, \pi, \varrho, (\ell_\pi, r_\pi), (\ell_\varrho, r_\varrho))$ for $k \ge 1$ and let π'' and ϱ'' be the returned streams. Suppose that streams π and ϱ and intervals (ℓ_π, r_π) and $(\ell_\varrho, r_\varrho)$ satisfy the assumptions on the input of AdvStrategy. Then, for $k > 1$, the assumptions on input for the recursive calls in lines 6 and 8 and for the call of RefineIntervals in line 7 are satisfied, and, for any $k \ge 1$, the streams π'' and ϱ'' are indistinguishable.*

Proof. The proof is by induction on k. In the base case $k = 1$, we use the fact that the $\frac{2}{\varepsilon}$ items from the corresponding intervals are appended in their order and that $\min\{i|a \le I_\pi[i]\} = \min\{i|b \le I_\varrho[i]\}$ for any $a \in (\ell_\pi, r_\pi)$ and $b \in (\ell_\varrho, r_\varrho)$ by assumption (iii) on the input of AdvStrategy. Thus, applying Lemma 4.1 for each pair of appended items, we get that π'' and ϱ'' are indistinguishable.

Now consider $k > 1$. Note that assumptions (i)-(iii) of the first recursive call (in line 6) are satisfied by the assumptions of the considered execution. So, by applying the inductive hypothesis for the first recursive call, streams π' and ϱ' are indistinguishable.

Next, the assumptions of procedure RefineIntervals, called in line 7, are satisfied, since streams π' and ϱ' are indistinguishable, π contains no item from (ℓ_π, r_π), ϱ contains no item from $(\ell_\varrho, r_\varrho)$, and the first recursive call in line 6 generates $N' = \frac{1}{\varepsilon} \cdot 2^{k-1}$ items from (ℓ_π, r_π) into π' and N' items from $(\ell_\varrho, r_\varrho)$ into ϱ'.

Then, assumption (i) of the second recursive call in line 8 holds, since π' and ϱ' are indistinguishable, and assumptions (ii) and (iii) are satisfied by applying Observation 1. Finally, we use the inductive hypothesis for the recursive call in line 8 and get that streams π'' and ϱ'' are indistinguishable. □

Our final observation is that for any $1 \le i \le |I_{\pi''}|$, we have that $\mathrm{rank}_{\pi''}(I_{\pi''}[i]) \le \mathrm{rank}_{\varrho''}(I_{\varrho''}[i])$. The proof follows by induction on k (similarly to Lemma 4.2) and by the definition of the new intervals in lines 2-4 of procedure RefineIntervals, namely, since the new interval for π is in the leftmost region of the largest gap, while the new interval for ϱ is in the rightmost region.

5 SPACE-GAP INEQUALITY

5.1 Intuition for the Inequality

In this section, we analyze the space required by data structure $\mathcal{D}$ when invoked on the two adversarial inputs from the previous section. Recall that our general goal is to prove that $\mathcal{D}$ needs to store $c \cdot \frac{1}{\varepsilon}$ items from the first half of the whole stream π (or, equivalently, from ϱ) and $c \cdot \frac{1}{\varepsilon} \cdot (k-1)$ items from the second half (by using induction on the second half), where $c > 0$ is a constant. Note also that if $\mathcal{D}$ stores $c \cdot \frac{1}{\varepsilon} \cdot k$ items from the first half of the stream, the second half of the argument is not even needed.

However, we actually need to prove a similar result for *any* internal node of the recursion tree, where the bounds as stated above may not hold. For instance, $\mathcal{D}$ may use nearly no space for some part of the stream, which implies a lot of uncertainty there, but still may be able to provide any ε-approximate ϕ-quantile, since the largest gap introduced earlier is very low. We thus give a space lower bound for an execution of AdvStrategy that depends on the largest gap size, denoted g, which is introduced in this execution. Roughly, the space lower bound is $c \cdot (\log g) \cdot N_k / g$ for a constant $c > 0$, so by setting $g = 2\varepsilon N_k$ we get the desired result. For technical reasons, the actual bound stated below as (2) is a bit more complicated. We refer to this bound as the "space-gap inequality", and the bulk of the work in this section is devoted to proving this inequality.

The crucial claim needed in the proof is that, for $k > 1$, the largest gap size g is, in essence, the sum of the largest gap sizes g' and g'' introduced in the first and the second recursive call, respectively. This claim allows us to distinguish two cases: Either the gap g' from the first recursive call is small (less than approximately half of g) and thus $\mathcal{D}$ uses a lot of space for items from the first recursive call, or $g' \gtrsim \frac{1}{2}g$, so $g'' \lesssim \frac{1}{2}g$ and we use induction on the second recursive call, together with a straightforward space lower bound for items from the first half of the stream.

5.2 Stating the Space-Gap Inequality

We perform the formal analysis by induction. We define

$$S(k, \pi, \varrho, (\ell_\pi, r_\pi), (\ell_\varrho, r_\varrho)) := \left| I_{\pi''}^{(\ell_\pi, r_\pi)} \right|,$$

where

$$(\pi'', \varrho'') = \textsc{AdvStrategy}(k, \pi, \varrho, (\ell_\pi, r_\pi), (\ell_\varrho, r_\varrho)).$$

In words, it is the size of the item array restricted to (ℓ_π, r_π) after the execution of $\mathcal{D}$ on stream π'' (or, equivalently, with ϱ instead of π). For simplicity, we write $S_k = S(k, \pi, \varrho, (\ell_\pi, r_\pi), (\ell_\varrho, r_\varrho))$.

We prove a lower bound for S_k that depends on the largest gap between the restricted item arrays for π and for ϱ. We enhance the definition of the gap to take the intervals restriction into account.

Definition 5.1. For indistinguishable streams σ and τ and intervals (ℓ_σ, r_σ) and (ℓ_τ, r_τ), let $\overline{\sigma}$ and $\overline{\tau}$ be the substreams of σ and τ consisting only of items from intervals (ℓ_σ, r_σ) and (ℓ_τ, r_τ), respectively. Moreover, let $I'_\sigma = I_\sigma^{(\ell_\sigma, r_\sigma)}$ and $I'_\tau = I_\tau^{(\ell_\tau, r_\tau)}$ be the restricted item arrays after processing σ and τ, respectively. We define the *largest gap* between I'_σ and I'_τ in intervals (ℓ_σ, r_σ) and (ℓ_τ, r_τ) as

$$\mathrm{gap}\left(\sigma, \tau, (\ell_\sigma, r_\sigma), (\ell_\tau, r_\tau)\right) = \max_{1 \le i < |I'_\tau|} \mathrm{rank}_{\overline{\tau}}(I'_\tau[i+1]) - \mathrm{rank}_{\overline{\sigma}}(I'_\sigma[i]).$$

Note that the ranks are with respect to substreams $\overline{\sigma}$ and $\overline{\tau}$, and that the largest gap is always at least one, supposing that the ranks of stored items are not smaller for τ than for σ. We again have $\mathrm{gap}\left(\sigma, \tau, (\ell_\sigma, r_\sigma), (\ell_\tau, r_\tau)\right) \ge \mathrm{gap}\left(\sigma, \sigma, (\ell_\sigma, r_\sigma), (\ell_\sigma, r_\sigma)\right)$. Also, as the restricted item arrays are enclosed by interval boundaries, the following simple bound holds:

$$S_k = S(k,\pi,\varrho,(\ell_\pi,r_\pi),(\ell_\varrho,r_\varrho)) \geq \frac{N_k}{\operatorname{gap}\left(\pi'',\pi'',(\ell_\pi,r_\pi),(\ell_\pi,r_\pi)\right)} \geq \frac{N_k}{\operatorname{gap}\left(\pi'',\varrho'',(\ell_\pi,r_\pi),(\ell_\varrho,r_\varrho)\right)}, \tag{1}$$

where $(\pi'',\varrho'') = \textsc{AdvStrategy}(k,\pi,\varrho,(\ell_\pi,r_\pi),(\ell_\varrho,r_\varrho))$ and $N_k = \frac{1}{\varepsilon}\cdot 2^k$. The following lemma (proved below) shows a stronger inequality between the space and the largest gap.

Lemma 5.2 (Space-gap inequality). *Consider an execution of* $\textsc{AdvStrategy}(k,\pi,\varrho,(\ell_\pi,r_\pi),(\ell_\varrho,r_\varrho))$. *Let* π'' *and* ϱ'' *be the returned streams, and let* $g := \operatorname{gap}\left(\pi'',\varrho'',(\ell_\pi,r_\pi),(\ell_\varrho,r_\varrho)\right)$. *Then, for* $S_k = S(k,\pi,\varrho,(\ell_\pi,r_\pi),(\ell_\varrho,r_\varrho))$, *the following* space-gap inequality *holds with* $c = \frac{1}{8} - 2\varepsilon$:

$$S_k \geq c\cdot(\log_2 g + 1)\cdot\left(\frac{N_k}{g} - \frac{1}{4\varepsilon}\right). \tag{2}$$

We remark that we do not optimize the constant c. Note that the right-hand side (RHS) of (2) is non-increasing for integer $g \geq 1$, as $(\log_2 g+1)/g$ is decreasing for $g \geq 2$ and equals 1 for $g \in \{1,2\}$.

First, observe that Theorem 2.2 directly follows from Lemma 5.2, and so our subsequent work will be in proving this space-gap inequality. Indeed, consider any integer $k \geq 1$ and let $(\pi,\varrho) = \textsc{AdvStrategy}(k,\emptyset,\emptyset,(-\infty,\infty),(-\infty,\infty))$ be the constructed streams of length N_k. Let $g = \operatorname{gap}\left(\pi,\varrho,(-\infty,\infty),(-\infty,\infty)\right) = \operatorname{gap}(\pi,\varrho)$. Since π and ϱ are indistinguishable by Lemma 4.2, we have $g \leq 2\varepsilon N_k$ by Lemma 3.4. Since the RHS of (2) is decreasing for $g \geq 2$ and $2\varepsilon N_k \geq 2$, it becomes the smallest for $g = 2\varepsilon N_k$. Thus, by Lemma 5.2, the memory used is at least

$$\begin{aligned} S_k &\geq c\cdot(\log_2 g+1)\cdot\left(\frac{N_k}{g}-\frac{1}{4\varepsilon}\right) \\ &\geq c\cdot(\log_2 2\varepsilon N_k+1)\cdot\left(\frac{1}{2\varepsilon}-\frac{1}{4\varepsilon}\right) \\ &= \Omega\left(\frac{1}{\varepsilon}\cdot\log\varepsilon N_k\right). \end{aligned}$$

5.3 Preliminaries for the Proof of Lemma 5.2

The proof is by induction on k. First, observe that (2) holds almost immediately if $g \leq 2^7$. Here, we have $\log_2 g + 1 \leq 8 \leq \frac{1}{c}$, and so by the bound in (1), $S_k > N_k/g \geq c\cdot(\log_2 g+1)\cdot\left(\frac{N_k}{g}-\frac{1}{4\varepsilon}\right)$. Similarly, if $g \geq 4\varepsilon N_k$, then (2) holds, since the RHS of (2) is at most 0 and $S_k \geq 0$.[3] We thus assume that $g \in (2^7, 4\varepsilon N_k)$, which immediately implies the base case $k=1$ of the induction, since $4\varepsilon N_1 = 8 < 2^7$ because $N_1 = \frac{2}{\varepsilon}$.

We now consider $k > 1$. We refer to streams $\pi,\varrho,\pi',\varrho',\pi'',\varrho''$, intervals (α_π,β_π) and $(\alpha_\varrho,\beta_\varrho)$ with the same meaning as in Pseudocode 2. Let $I'_{\pi'} = I_{\pi'}^{(\ell_\pi,r_\pi)}$ and $I'_{\varrho'} = I_{\varrho'}^{(\ell_\varrho,r_\varrho)}$ be the restricted item arrays, as in Pseudocode 1. We make use of the following notation:

- Let $\pi'_{k-1},\varrho'_{k-1}$ be the substreams constructed during the recursive call in line 6. Let S'_{k-1} be the size of $I'_{\pi'}$ (or, equivalently, of $I'_{\varrho'}$), and let $g' = \operatorname{gap}\left(\pi',\varrho',(\ell_\pi,r_\pi),(\ell_\varrho,r_\varrho)\right)$ be the largest gap in the input intervals after $\mathcal{D}$ processes one of streams π' and ϱ'.
- Let $I''_{\pi''} = I_{\pi''}^{(\alpha_\pi,\beta_\pi)}$ and $I''_{\varrho''} = I_{\varrho''}^{(\alpha_\varrho,\beta_\varrho)}$ be the item arrays restricted to the new intervals after $\mathcal{D}$ processes streams π'' and ϱ'', respectively. Let S''_{k-1} be the size of $I''_{\pi''}$, and let $g'' = \operatorname{gap}\left(\pi'',\varrho'',(\alpha_\pi,\beta_\pi),(\alpha_\varrho,\beta_\varrho)\right)$ be the largest gap in the new intervals. Let π''_{k-1} and ϱ''_{k-1} be the substreams constructed during the recursive call in line 8.
- Let $I'_{\pi''} = I_{\pi''}^{(\ell_\pi,r_\pi)}$ and $I'_{\varrho''} = I_{\varrho''}^{(\ell_\varrho,r_\varrho)}$ be the item arrays restricted to the input intervals after $\mathcal{D}$ processes streams π'' and ϱ'', respectively.
- Finally, let π_k and ϱ_k be the substreams of π'' and ϱ'', restricted to (ℓ_π,r_π) and (ℓ_ϱ,r_ϱ), respectively (i.e., π_k and ϱ_k consist of the items appended by the considered execution).

We remark that notation I' abbreviates the restriction to intervals (ℓ_π,r_π) and (ℓ_ϱ,r_ϱ) (depending on the stream), while notation I'' implicitly denotes the restriction to the new intervals (α_π,β_π) and $(\alpha_\varrho,\beta_\varrho)$. Note that $\pi' = \pi\pi'_{k-1}$, and $\pi'' = \pi'\pi''_{k-1} = \pi\pi_k$, and $\pi_k = \pi'_{k-1}\pi''_{k-1}$, and similarly for streams ϱ', ϱ'', and ϱ_k.

We now show a crucial relation between the gaps.

Claim 1. $g \geq g' + g'' - 1$

Proof. Define i to be

$$i := \arg\max_{1\leq i'<|I''_{\pi''}|} \operatorname{rank}_{\varrho''_{k-1}}(I''_{\varrho''}[i'+1]) - \operatorname{rank}_{\pi''_{k-1}}(I''_{\pi''}[i']),$$

i.e., the position of the largest gap in the arrays $I''_{\pi''}$ and $I''_{\varrho''}$. Let $a := I''_{\pi''}[i]$ and $b := I''_{\varrho''}[i+1]$ be the two items whose rank difference determines the largest gap size. Note that, while $\mathcal{D}$ stores a and b in $I''_{\pi''}$ and $I''_{\varrho''}$, these two items does not necessarily need to be stored in $I'_{\pi''}$ and $I'_{\varrho''}$, respectively. This may happen for a only in case $a = \alpha_\pi$ and thus $i = 1$, and similarly, for b only in case $b = \beta_\varrho$ and $i = |I''_{\pi''}| - 1$. Indeed, for $i > 1$, item $a = I''_{\pi''}[i]$ must be in the whole item array $I_{\pi''}$ and thus also in $I'_{\pi''}$, and similarly, if $i < |I''_{\pi''}| - 1$, item $b = I''_{\varrho''}[i+1]$ must be in $I_{\varrho''}$ and thus in $I'_{\varrho''}$. (In the special case $|I''_{\pi''}| = 2$, both a and b may not be in $I'_{\pi''}$ and in $I'_{\varrho''}$, respectively, while if $|I''_{\pi''}| > 2$, at least one of a or b is actually stored.)

Let j be the largest integer such that $I'_{\pi''}[j] \leq a$, and let $a' := I'_{\pi''}[j]$; by the above observations, $a' = a$ unless $i = 1$ and $a \notin I'_{\pi''}$. Let $b' := I'_{\varrho''}[j+1]$. We now show that $b' \geq b$. Indeed, this clearly holds if $b' = b$, so suppose $b' \neq b$. This may only happen if $b = \beta_\varrho$ is not in $I'_{\varrho''}$ and $i = |I''_{\pi''}| - 1$. We consider two cases:

Case 1: If $a' = I'_{\pi''}[j] \in (\alpha_\pi,\beta_\pi)$, then $I'_{\varrho''}[j] \in (\alpha_\varrho,\beta_\varrho)$ as π'' and ϱ'' are indistinguishable and only the last N_{k-1} items are from these intervals. Moreover, as $i = |I''_{\pi''}| - 1$, index j is the largest such that $I'_{\varrho''}[j] \in (\alpha_\varrho,\beta_\varrho)$, thus $b' = I'_{\varrho''}[j+1] \geq \beta_\varrho = b$.

Case 2: Otherwise, $a' \leq a = \alpha_\pi$, which may only happen when $i = 1$. As also $i = |I''_{\pi''}| - 1$, we have $|I''_{\pi''}| = 2$, i.e., no items from (α_π,β_π) and from $(\alpha_\varrho,\beta_\varrho)$ are stored in $I'_{\pi''}$ and in $I'_{\varrho''}$, respectively. Then we have $I'_{\pi''}[j+1] \geq \beta_\pi$, by the definition of j. Before the

[3] Note, however, that we cannot use Lemma 3.4 to show $g \leq 2\varepsilon N_k$, since the largest gap has size bounded by 2ε times the length of π'' or ϱ'', which can be much larger than N_k (due to items from other branches of the recursion tree).

second recursive call, it holds that $\alpha_\pi = I'_{\pi'}[\ell]$ and $\beta_\varrho = I'_{\varrho'}[\ell+1]$ for some index ℓ, i.e., there are ℓ items stored in $I'_{\pi'}$ and in $I'_{\varrho'}$ which are not larger than α_π and α_ϱ, respectively. By $a' = I'_{\pi''}[j] \le \alpha_\pi$ and by the definition of j, there are $j \le \ell$ items in $I'_{\pi''}$ no larger than α_π, and hence, by indistinguishability of π'' and ϱ'', there are j items in $I'_{\varrho''}$ no larger than α_ϱ. Since no item in $(\alpha_\varrho, \beta_\varrho)$ is stored in $I'_{\varrho''}$, we conclude that $b' = I'_{\varrho''}[j+1] \ge \beta_\varrho = b$ holds.

To prove the claim, it is sufficient to show

$$\mathrm{rank}_{\varrho_k}(b') - \mathrm{rank}_{\pi_k}(a') \ge g' + g'' - 1\,, \tag{3}$$

as the difference on the LHS is taken into account in the definition of g. We have

$$g'' = \mathrm{rank}_{\varrho''_{k-1}}(b) - \mathrm{rank}_{\pi''_{k-1}}(a) \le \mathrm{rank}_{\varrho''_{k-1}}(b') - \mathrm{rank}_{\pi''_{k-1}}(a')\,,$$

since $b' \ge b$ and $a' \le a$. This rank difference is w.r.t. substreams π''_{k-1} and ϱ''_{k-1}, and we now show that when considering π_k and ϱ_k, the difference increases by $g'-1$. Indeed, as $a' < \beta_\pi$ and $b' > \alpha_\varrho$, it holds that

$$\mathrm{rank}_{\varrho''_{k-1}}(b') - \mathrm{rank}_{\pi''_{k-1}}(a') = \mathrm{rank}_{\varrho_k}(b') - \mathrm{rank}_{\pi_k}(a') - (g'-1),$$

using the definitions of g' and of the new intervals in lines 2—4 of procedure RefineIntervals (Pseudocode 1). Summarizing, we have $g'' \le \mathrm{rank}_{\varrho''_{k-1}}(b') - \mathrm{rank}_{\pi''_{k-1}}(a') = \mathrm{rank}_{\varrho_k}(b') - \mathrm{rank}_{\pi_k}(a') - (g'-1)$, which shows (3) by rearrangement. □

5.4 Completing the Proof of Lemma 5.2

In the inductive proof of (2) for $k > 1$, we consider two main cases, according to whether or not g' is relatively small (compared to g). Recall that we still assume that $g \in (2^7, 4\varepsilon N_k)$.

Case 1: Suppose that the following inequality holds

$$c \cdot (\log_2 g' + 1) \cdot \left(\frac{N_{k-1}}{g'} - \frac{1}{4\varepsilon}\right) \ge c \cdot (\log_2 g + 1) \cdot \left(\frac{N_k}{g} - \frac{1}{4\varepsilon}\right). \tag{4}$$

We claim that this inequality is sufficient for (2). Indeed, first observe that $S_k \ge S'_{k-1}$. This follows from the assumption that the size of the (whole) item array does not decrease and that all items that are appended to the streams in the considered execution of AdvStrategy are within the current intervals (ℓ_π, r_π) and $(\ell_\varrho, r_\varrho)$, so the number of stored items from π'' that are outside (ℓ_π, r_π) cannot increase while $\mathcal{D}$ processes items from the considered execution. Then we use the induction hypothesis from (2) to get $S'_{k-1} \ge c \cdot (\log_2 g' + 1) \cdot \left(\frac{N_{k-1}}{g'} - \frac{1}{4\varepsilon}\right)$, and finally, (2) follows from (4), since we have $S_k \ge S'_{k-1} \ge c \cdot (\log_2 g' + 1) \cdot \left(\frac{N_{k-1}}{g'} - \frac{1}{4\varepsilon}\right) \ge c \cdot (\log_2 g + 1) \cdot \left(\frac{N_k}{g} - \frac{1}{4\varepsilon}\right)$.

Case 2: In the remainder of the analysis, assume that (4) does not hold. We first show that g'' is substantially smaller than g, by a factor a bit larger than $\frac{1}{2}$. Namely, we prove the following:

Lemma 5.3. *Assuming $g > 2^7$ and that (4) does not hold we have*

$$g'' < \frac{1}{2} \cdot g \cdot \frac{\log_2 g + 4}{\log_2 g + 1}\,. \tag{5}$$

Proof. To show (5), since (4) does not hold, we have

$$c \cdot (\log_2 g' + 1) \cdot \left(\frac{N_{k-1}}{g'} - \frac{1}{4\varepsilon}\right) < c \cdot (\log_2 g + 1) \cdot \left(\frac{N_k}{g} - \frac{1}{4\varepsilon}\right). \tag{6}$$

By Claim 1, it holds that $g \ge g' + g'' - 1 \ge g'$ as $g'' \ge 1$, which allows us to simplify (6) to

$$c \cdot (\log_2 g' + 1) \cdot \frac{N_{k-1}}{g'} < c \cdot (\log_2 g + 1) \cdot \frac{N_k}{g}\,.$$

After dividing this inequality by $c \cdot N_k = c \cdot 2N_{k-1}$, we obtain

$$\frac{\log_2 g' + 1}{2g'} < \frac{\log_2 g + 1}{g}\,. \tag{7}$$

Rearranging, we get

$$g' > \frac{g}{2} \cdot \frac{\log_2 g' + 1}{\log_2 g + 1}\,. \tag{8}$$

Next, we claim that $\log_2 g' \ge \log_2 g - 2$. Suppose for a contradiction that $\log_2 g' < \log_2 g - 2$, i.e., $g' < \frac{1}{4}g$. Using that $\frac{\log_2 g' + 1}{2g'}$ is decreasing for $g' \ge 2$ and equal to $\frac{1}{2}$ for $g' \in \{0, 1\}$, we substitute $g' = \frac{1}{4}g$ into (7) and get $2 \cdot \frac{\log_2 g - 1}{g} < \frac{\log_2 g + 1}{g}$. After rearranging we have $\log_2 g < 3$, which is a contradiction with our assumption that $g > 2^7$.

Thus, (8) and the above claim imply

$$g' > \frac{g}{2} \cdot \frac{\log_2 g - 1}{\log_2 g + 1}\,. \tag{9}$$

Using Claim 1 together with (9), we obtain $g > \frac{g}{2} \cdot \frac{\log_2 g - 1}{\log_2 g + 1} + g'' - 1$, and by rearranging, we get

$$\begin{aligned} g'' &< g \cdot \left(1 - \frac{1}{2} \cdot \frac{\log_2 g - 1}{\log_2 g + 1}\right) + 1 = \frac{1}{2} \cdot g \cdot \left(2 - \frac{\log_2 g - 1}{\log_2 g + 1} + \frac{2}{g}\right) \\ &< \frac{1}{2} \cdot g \cdot \left(2 - \frac{\log_2 g - 1}{\log_2 g + 1} + \frac{1}{\log_2 g + 1}\right) \\ &= \frac{1}{2} \cdot g \cdot \frac{2 \cdot (\log_2 g + 1) - (\log_2 g - 1) + 1}{\log_2 g + 1} = \frac{1}{2} \cdot g \cdot \frac{\log_2 g + 4}{\log_2 g + 1}\,, \end{aligned}$$

where in the third line we use $\log_2 g + 1 < \frac{1}{2}g$ for $g > 2^7$. This concludes the proof of the lemma. □

We continue in the proof of (2) in Case 2. We now take the second recursive call (in line 8) into account. By induction, the space used for items from the second recursive call, which equals to $|I''_{\pi''}| = |I''_{\varrho''}|$, is at least $S''_{k-1} \ge c \cdot (\log_2 g'' + 1) \cdot \left(\frac{N_{k-1}}{g''} - \frac{1}{4\varepsilon}\right)$. Using (5) and the monotonicity of the RHS of (2), we get

$$S''_{k-1} \ge c \cdot \left(\log_2 \left(\frac{1}{2} \cdot g \cdot \frac{\log_2 g + 4}{\log_2 g + 1}\right) + 1\right) \cdot \left(\frac{N_{k-1}}{\frac{1}{2} \cdot g \cdot \frac{\log_2 g + 4}{\log_2 g + 1}} - \frac{1}{4\varepsilon}\right). \tag{10}$$

The second factor on the RHS of (10) is at least $\log_2 g$, since

$$\log_2 \left(\frac{1}{2} \cdot g \cdot \frac{\log_2 g + 4}{\log_2 g + 1}\right) + 1 \ge \log_2 \left(\frac{1}{2} \cdot g\right) + 1 = \log_2 g.$$

Using also $N_{k-1} = \frac{1}{2}N_k$, we get

$$S''_{k-1} \ge c \cdot \log_2 g \cdot \left(\frac{\frac{1}{2}N_k}{\frac{1}{2} \cdot g \cdot \frac{\log_2 g + 4}{\log_2 g + 1}} - \frac{1}{4\varepsilon}\right) = \frac{c \cdot \log_2 g \cdot N_k}{g \cdot \frac{\log_2 g + 4}{\log_2 g + 1}} - \frac{c \cdot \log_2 g}{4\varepsilon}\,. \tag{11}$$

Consider the N_{k-1} items from π'_{k-1}, which are the items from the first recursive call (in line 6). For them, we just use a simple bound (1): Since the largest gap in $I'_{\pi''}$ is at most g and since there

can be two gaps around stored items from π''_{k-1} (i.e., those in $I''_{\pi''}$), the number of items from π'_{k-1} stored in $I'_{\pi''}$ is at least

$$\frac{N_{k-1} - 2g}{g} = \frac{N_k - 4g}{2g} \geq \frac{N_k - 16\varepsilon N_k}{2g}\,, \tag{12}$$

using the assumption that $g \leq 4\varepsilon N_k$.

Summarizing, (11) gives a lower bound on $|I''_{\pi''}|$, i.e., the number of stored items from π''_{k-1}, and (12) a lower bound on the number of items in $I'_{\pi''}$ that are not in $I''_{\pi''}$. Thus, our aim is to show that

$$\frac{c \cdot \log_2 g \cdot N_k}{g \cdot \frac{\log_2 g+4}{\log_2 g+1}} - \frac{c \cdot \log_2 g}{4\varepsilon} + \frac{N_k - 16\varepsilon N_k}{2g}$$
$$\geq c \cdot (\log_2 g + 1) \cdot \frac{N_k}{g} - \frac{c \cdot (\log_2 g + 1)}{4\varepsilon}\,, \tag{13}$$

which implies (2) as $S_k \geq |I'_{\pi''}|$ and $|I'_{\pi''}|$ is lower bounded by the LHS of (13). To show (13), first note that $-\frac{c \cdot \log_2 g}{4\varepsilon} \geq -\frac{c \cdot (\log_2 g+1)}{4\varepsilon}$, we thus ignore these expressions. Next, we multiply both sides of (13) by $g/(c \cdot N_k)$ and get that it suffices to show

$$\frac{\log_2 g}{\frac{\log_2 g+4}{\log_2 g+1}} + \frac{1 - 16\varepsilon}{2c} \geq \log_2 g + 1\,. \tag{14}$$

After multiplying both sides of (14) by $\frac{\log_2 g+4}{\log_2 g+1} \geq 1$ (the second fraction on the LHS is not multiplied, for simplicity), we obtain $\log_2 g + \frac{1-16\varepsilon}{2c} \geq \log_2 g + 4$, which holds for $c \leq \frac{1}{8} - 2\varepsilon$. This completes the proof of Lemma 5.2, and so the space bound follows.

6 COROLLARIES AND CONCLUSIONS

Our construction closes the asymptotic gap in the space bounds for deterministic comparison-based quantile summaries and yields the optimality of the Greenwald and Khanna's quantile summary [6]. A drawback of their quantile summary is that it carries out an intricate merging of stored tuples, where each tuple consists of a stored item together with lower and upper bounds on its rank. A simplified (greedy) version, which merges stored tuples whenever it is possible, was suggested already in [6], and according to experiments reported in Luo *et al.* [13], it performs better in practice than the intricate algorithm analyzed in [6]. It is an interesting open problem whether or not the upper bound of $O(\frac{1}{\varepsilon} \cdot \log \varepsilon N)$ holds for some simpler variant of the Greenwald and Khanna's algorithm.

6.1 Finding an Approximate Median

One of the direct consequences of our result is that finding an ε-approximate median requires roughly the same space as constructing a quantile summary. (This can be done similarly for any other ϕ-quantile as long as $\varepsilon \ll \phi \ll 1 - \varepsilon$.)

Theorem 6.1. *For any $\varepsilon > 0$ small enough, there is no deterministic comparison-based streaming algorithm that finds an ε-approximate median in the stream and runs in space $o(\frac{1}{\varepsilon} \cdot \log \varepsilon N)$ on any stream of length N.*

Proof sketch. Consider the streams π and ϱ constructed by the adversarial procedure from Section 4, i.e., $(\pi, \varrho) = \text{AdvStrategy}(k, \emptyset, \emptyset, (-\infty, \infty), (-\infty, \infty))$. Let $g = \text{gap}(\pi, \varrho)$. If $g \leq 4\varepsilon N_k$, then the analysis in Section 5, with an appropriately adjusted space-gap inequality, shows that the algorithm uses space $\Omega(\frac{1}{\varepsilon} \cdot \log \varepsilon N_k)$. Thus, consider the case $g > 4\varepsilon N_k$, which implies that there exists $\phi' \in (0, 1)$ such that the item array does not store a 2ε-approximate ϕ'-quantile. If $\phi' < 0.5$, we append $(1 - 2\phi') \cdot N_k \leq N_k$ items to streams π and ϱ that are smaller than any item appended so far, and after that the algorithm cannot return an ε-approximate median. Otherwise, $\phi' \geq 0.5$ and we append $(2\phi' - 1) \cdot N_k \leq N_k$ items to streams π and ϱ that are larger than any item appended so far. Thus, in this case also an ε-approximate median is not stored. □

6.2 Estimating Rank

We now consider data structures for the following Estimating Rank problem, which is closely related to computing ε-approximate quantiles: The input arrives as a stream of N items from a totally ordered universe U, and the goal is to design a data structure with small space cost which is able to provide an ε-approximate rank for any query $q \in U$, i.e., the number of items in the stream which are not larger than q, up to an additive error of $\pm\varepsilon N$. Our construction directly implies a space lower bound for comparison-based data structures, which are defined similarly as in Definition 2.1.[4]

Theorem 6.2. *For any $0 < \varepsilon < \frac{1}{16}$, there is no deterministic comparison-based data structure for Estimating Rank which stores $o(\frac{1}{\varepsilon} \cdot \log \varepsilon N)$ items on any input stream of length N.*

Proof sketch. Let $\mathcal{D}$ be a deterministic comparison-based data structure for Estimating Rank. Consider again the pair of streams $(\pi, \varrho) = \text{AdvStrategy}(k, \emptyset, \emptyset, (-\infty, \infty), (-\infty, \infty))$. Let $g = \text{gap}(\pi, \varrho)$. The space-gap inequality (Lemma 5.2) holds, using the same proof. As shown at the beginning of Section 5, if $g \leq 2\varepsilon N_k + 2$, then $\mathcal{D}$ needs to store $\Omega(\frac{1}{\varepsilon} \cdot \log \varepsilon N_k)$ items (the +2 makes no effective difference in the calculation). It remains to observe that if $\mathcal{D}$ provides an ε-approximate rank of any query $q \in U$, then $g \leq 2\varepsilon N_k + 2$.

Indeed, suppose for a contradiction that $g > 2\varepsilon N_k + 2$, which implies that there is $1 \leq i < |I_\pi| = |I_\varrho|$ such that $\text{rank}_\varrho(I_\varrho[i+1]) - \text{rank}_\pi(I_\pi[i]) > 2\varepsilon N_k + 2$. Let q_π be an item which lies in $(I_\pi[i], \text{next}(\pi, I_\pi[i]))$, that is, just after $I_\pi[i]$ in U (q_π exists by our continuity assumption). Similarly, let q_ϱ be an item in $(\text{prev}(\varrho, I_\varrho[i+1]), I_\varrho[i+1])$. Let r be the rank returned by $\mathcal{D}$ when run on query q_π after processing stream π. Observe that $\mathcal{D}$ returns r also on query q_ϱ after processing stream ϱ, since π and ϱ are indistinguishable, $\mathcal{D}$ is comparison-based, and the results of comparisons with stored items are the same in both cases. However, the true ranks satisfy $\text{rank}_\pi(q_\pi) = \text{rank}_\pi(I_\pi[i]) + 1$ and $\text{rank}_\varrho(q_\varrho) = \text{rank}_\varrho(I_\varrho[i+1]) - 1$, thus $\text{rank}_\varrho(q_\varrho) - \text{rank}_\pi(q_\pi) > 2\varepsilon N_k$. It follows that r differs from $\text{rank}_\pi(q_\pi)$ or from $\text{rank}_\varrho(q_\varrho)$ by more than εN_k, which is a contradiction. □

6.3 Randomized Algorithms

We now turn our attention to randomized quantile summaries, which may fail to provide an ε-approximate ϕ-quantile, for some ϕ, with probability bounded by a parameter δ. Karnin *et al.* [11] designed a randomized comparison-based quantile summary with storage cost $O(\frac{1}{\varepsilon} \cdot \log\log \frac{1}{\varepsilon\delta})$. They also proved the matching lower

[4] We only need to replace item *(iv)* of Definition 2.1 by *(iv) Given a query $q \in U$, the computation of $\mathcal{D}$ is determined solely by the results of comparisons between q and $I[j]$, for $j = 1, \ldots, |I|$, the number of items stored, and the contents of G.*

bound, which however holds only for a certain stream length (depending on ε) and for δ exponentially close to 0. We state it more precisely as follows.

Theorem 6.3 (Theorem 6 in [11]). *There is no randomized comparison-based ε-approximate quantile summary with failure probability less than $\delta = 1/N!$, which stores $o(\frac{1}{\varepsilon} \cdot \log\log \frac{1}{\delta})$ items on any input stream of length $N = \Theta\left(\frac{1}{\varepsilon^2} \cdot \log^2 \frac{1}{\varepsilon}\right)$.*

The proof follows from reducing the randomized case to the deterministic case and using the lower bound of $\Omega(\frac{1}{\varepsilon} \cdot \log \frac{1}{\varepsilon})$ [10], which holds for streams of length $N = \Theta\left(\frac{1}{\varepsilon^2} \cdot \log^2 \frac{1}{\varepsilon}\right)$. Suppose for a contradiction that there exists a comparison-based ε-approximate quantile summary which stores $o(\frac{1}{\varepsilon} \cdot \log\log \frac{1}{\delta})$ items for $\delta = 1/N!$. Note that if failure probability is below $1/N!$, a randomized comparison-based quantile summary succeeds simultaneously for all streams of length N with probability > 0 (by the union bound). More precisely, it succeeds for all permutations of any given set of N distinct items, which is sufficient in the comparison-based model. Thus, there exists a choice of random bits which provides a correct result for all streams of length N. Hard-coding these bits, we obtain a deterministic algorithm running in space $o(\frac{1}{\varepsilon} \cdot \log\log \frac{1}{\delta}) = o(\frac{1}{\varepsilon} \cdot \log\log e^{N \log N}) = o(\frac{1}{\varepsilon} \cdot \log N) = o(\frac{1}{\varepsilon} \cdot \log \frac{1}{\varepsilon})$, which contradicts the lower bound in [10]. We remark that the lower bound holds even for finding the median.

Using our lower bound of $\Omega(\frac{1}{\varepsilon} \cdot \log \varepsilon N)$ for deterministic quantile summaries, we strengthen the randomized lower bound so that it holds for any stream length N, which in turn gives a higher space bound. Hence, using the same proof, we obtain:

Theorem 6.4. *There is no randomized comparison-based ε-approximate quantile summary with failure probability less than $\delta = 1/N!$, which stores $o(\frac{1}{\varepsilon} \cdot \log\log \frac{1}{\delta})$ items on any input stream of length N.*

Note that the lower bound of $\Omega(\frac{1}{\varepsilon} \cdot \log\log \frac{1}{\delta})$ for randomized quantile summaries trivially holds if $\delta > 0$ is a fixed constant (say, $\delta = 0.01$), since any quantile summary needs to store $\Omega(\frac{1}{\varepsilon})$ items. It remains an open problem whether or not the lower bound of $\Omega(\frac{1}{\varepsilon} \cdot \log\log \frac{1}{\delta})$ holds for $\delta = 1/\text{poly}(N)$ or for $\delta = 1/\text{polylog}(N)$.

6.4 Biased Quantiles

Note that the quantiles problem studied in this paper gives a *uniform* error guarantee of εN for any quantile $\phi \in [0, 1]$. A stronger, relative-error guarantee of $\varepsilon\phi N$ was proposed by Cormode *et al.* [3], under the name of *biased quantiles*. Namely, given a query $\phi \in [0, 1]$, an ε-approximate biased quantile summary returns a ϕ'-quantile for some $\phi' = [(1 - \varepsilon) \cdot \phi, (1 + \varepsilon) \cdot \phi]$.[5] In other words, when queried for the k-th smallest item (where $k = \lfloor \phi N \rfloor$), the algorithm may return the k'-th smallest item for some $k' \in [(1 - \varepsilon) \cdot k, (1 + \varepsilon) \cdot k]$. Note that the relative-error guarantee and the uniform guarantee of εN are essentially the same for $\phi = \Omega(1)$, up to a constant factor. That is, biased quantiles provide a substantially stronger guarantee for extreme values of ϕ only, e.g., for $\phi = 1/\sqrt{N}$.

[5] Strictly speaking, the definition in [3] is weaker, requiring only to approximate items at ranks $\phi^j \cdot N$ with error at most $\varepsilon \cdot \phi^j \cdot N$ for $j = 0, \ldots, \lfloor \log_{1/\phi} N \rfloor$ and some parameter $\phi \in (0, 1)$ known in advance.

Any summary for biased quantiles, even constructed offline, requires space of $\Omega(\frac{1}{\varepsilon} \cdot \log \varepsilon N)$, which is the best lower bound proved so far. This follows by observing that any summary needs to store the $\frac{1}{\varepsilon}$ smallest items; among the next $\frac{1}{\varepsilon}$ items, it should store every other one; and more generally, it needs to store $\Omega(\frac{1}{\varepsilon})$ items among those with ranks between $\frac{2^i}{\varepsilon}$ and $\frac{2^{i+1}}{\varepsilon}$ for any $i = 0, \ldots, \log \varepsilon N$. The state-of-the-art upper bounds for the space requirement in the streaming setting are $O(\frac{1}{\varepsilon} \cdot \log^3 \varepsilon N)$, using a deterministic comparison-based "merge & prune" strategy [21], and $O(\frac{1}{\varepsilon} \cdot \log \varepsilon N \cdot \log |U|)$ for a fixed universe U [4], using a modification of q-digest from [18]. The only randomized algorithms are sampling-based and require space of $O(\frac{1}{\varepsilon^2} \cdot \log \frac{1}{\delta} \cdot \log \varepsilon N)$ in the worst case [9, 20].

We show that our construction from Section 4 can be used to improve the lower bound for ε-approximate biased quantile summaries by a further $\log \varepsilon N$ factor. Note that the definition of comparison-based summaries (Definition 2.1) translates to this setting, as well as Definitions 3.1 and 3.2 which define equivalent memory states and indistinguishable streams.

Theorem 6.5. *For any $0 < \varepsilon < \frac{1}{16}$, there is no deterministic comparison-based ε-approximate biased quantile summary which stores $o(\frac{1}{\varepsilon} \cdot \log^2 \varepsilon N)$ items on any input stream of length N.*

Proof sketch. For an integer k, we show that any deterministic comparison-based ε-approximate biased quantile summary needs to use $\Omega(\frac{1}{\varepsilon} \cdot k^2)$ space on some stream of length $O(\frac{1}{\varepsilon} \cdot 2^k)$, so that $k = \Omega(\log \varepsilon N)$. We have k phases, executed from phase 1 to phase k. In phase i, we use the construction from Section 4 to generate $N_i = \frac{1}{\varepsilon} \cdot 2^i$ new items that are larger than all items from previous phases $j < i$. That is, in phase i we execute $(\pi_i, \varrho_i) = \textsc{AdvStrategy}(i, \pi_{i-1}, \varrho_{i-1}, (\max(\pi_{i-1}), \infty), (\max(\varrho_{i-1}), \infty))$, where π_{i-1} and ϱ_{i-1} are the streams from the previous phase (and $\pi_0 = \varrho_0 = \emptyset$) and $\max(\sigma)$ is the largest item in stream σ (so σ contains no item from $(\max(\sigma), \infty)$). The streams π_i and ϱ_i are indistinguishable for any i, by an iterative application of Lemma 4.2.

A similar proof as in Lemma 3.4 shows that the largest gap among items sent in phase i is $O(\varepsilon N_i)$ with respect to the relative-error guarantee. This uses the fact that $\Theta(N_i)$ items were sent in previous phases and thus the relative-error guarantee for items from phase i is $\Theta(\varepsilon N_i)$. We can thus apply the analysis in Section 5, in particular the space-gap inequality (2). We remark that even though the streams already contain some items before phase i, this does not affect the analysis. Indeed, the space-gap inequality works for any execution in the recursion tree of AdvStrategy, and the streams may already contain many items before this execution.

Thus, the summary needs to store $\Omega(\frac{1}{\varepsilon} \cdot i)$ items from phase i. Note that this includes also the minimum and maximum items from phase i. With constant additional storage per phase, we may suppose that the minimum and maximum items from each phase are stored all the time after they arrive, and that we know their exact ranks (as the number of items in each phase is fixed). Consequently, different phases can be treated independently.

The final observation is that the largest gap among items from phase i remains $O(\varepsilon N_i)$ even after items from subsequent phases arrive. This follows from the relative-error guarantee, since all subsequent items in the streams are larger than items from phase i. Hence, the algorithm cannot remove items from phase i from the

memory when processing items from subsequent phases, except for items that can be removed when last item from phase i arrives. To conclude, the algorithm stores $\Omega(\frac{1}{\varepsilon} \cdot i)$ items from phase i at the end, and summing over all k phases gives the result. □

For randomized algorithms that provide all biased quantiles with probability more than $1 - \delta$ for $\delta = 1/N!$, the same reduction as for the quantiles problem in Section 6.3 (with uniform error) shows that there is no comparison-based randomized biased quantile summary running in space $o(\frac{1}{\varepsilon} \cdot \log \varepsilon N \cdot \log\log \frac{1}{\delta})$. Closing the gaps for (deterministic or randomized) biased quantiles remains open.

REFERENCES

[1] Problem 2: Quantiles. https://sublinear.info/2, 2006. Accessed: 2019-12-10.

[2] Pankaj K. Agarwal, Graham Cormode, Zengfeng Huang, Jeff M. Phillips, Zhewei Wei, and Ke Yi. Mergeable summaries. *ACM Trans. Database Syst.*, 38(4):26:1–26:28, December 2013.

[3] G. Cormode, F. Korn, S. Muthukrishnan, and D. Srivastava. Effective computation of biased quantiles over data streams. In *21st International Conference on Data Engineering (ICDE'05)*, pages 20–31, April 2005.

[4] Graham Cormode, Flip Korn, S. Muthukrishnan, and Divesh 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*, PODS '06, pages 263–272, New York, NY, USA, 2006. ACM.

[5] David Felber and Rafail Ostrovsky. A randomized online quantile summary in $O(1/\epsilon * \log(1/\epsilon))$ words. In *Approximation, Randomization, and Combinatorial Optimization Algorithms and Techniques (APPROX/RANDOM)*, volume 40, pages 775–785. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2015.

[6] Michael Greenwald and Sanjeev Khanna. Space-efficient online computation of quantile summaries. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, SIGMOD '01, pages 58–66, November 2001.

[7] Michael B. Greenwald and Sanjeev Khanna. *Quantiles and Equi-depth Histograms over Streams*, pages 45–86. Springer Berlin Heidelberg, 2016.

[8] S. Guha and A. McGregor. Stream order and order statistics: Quantile estimation in random-order streams. *SIAM Journal on Computing*, 38(5):2044–2059, 2009.

[9] Anupam Gupta and Francis X. Zane. Counting inversions in lists. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '03, pages 253–254, Philadelphia, PA, USA, 2003. Society for Industrial and Applied Mathematics.

[10] Regant Y. S. Hung and Hingfung F. Ting. An $\Omega(\frac{1}{\varepsilon} \log \frac{1}{\varepsilon})$ space lower bound for finding ε-approximate quantiles in a data stream. In *Frontiers in Algorithmics*, pages 89–100. Springer Berlin Heidelberg, 2010.

[11] Z. Karnin, K. Lang, and E. Liberty. Optimal quantile approximation in streams. In *2016 IEEE 57th Annual Symposium on Foundations of Computer Science*, FOCS '16, pages 71–78, Oct 2016.

[12] Ashwin Lall. Data streaming algorithms for the Kolmogorov-Smirnov test. In *2015 IEEE International Conference on Big Data*, pages 95–104, 2015.

[13] Ge Luo, Lu Wang, Ke Yi, and Graham Cormode. Quantiles over data streams: Experimental comparisons, new analyses, and further improvements. *The VLDB Journal*, 25(4):449–472, August 2016.

[14] Gurmeet Singh Manku, Sridhar Rajagopalan, and Bruce G. Lindsay. Approximate medians and other quantiles in one pass and with limited memory. In *Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data*, SIGMOD '98, pages 426–435. ACM, 1998.

[15] Gurmeet Singh Manku, Sridhar Rajagopalan, and Bruce G. Lindsay. Random sampling techniques for space efficient online computation of order statistics of large datasets. In *Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data*, SIGMOD '99, pages 251–262. ACM, 1999.

[16] Andrew McGregor and Paul Valiant. The shifting sands algorithm. In *Proceedings of the 23rd Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '12, pages 453–458, 2012.

[17] J.I. Munro and M.S. Paterson. Selection and sorting with limited storage. *Theoretical Computer Science*, 12(3):315 – 323, 1980.

[18] Nisheeth Shrivastava, Chiranjeeb Buragohain, Divyakant Agrawal, and Subhash Suri. Medians and beyond: New aggregation techniques for sensor networks. In *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems*, SenSys '04, pages 239–249. ACM, 2004.

[19] Yufei Tao, Wenqing Lin, and Xiaokui Xiao. Minimal MapReduce algorithms. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, SIGMOD '13, pages 529–540. ACM, 2013.

[20] Ying Zhang, Xuemin Lin, Jian Xu, F. Korn, and Wei Wang. Space-efficient relative error order sketch over data streams. In *22nd International Conference on Data Engineering (ICDE'06)*, pages 51–51, April 2006.

[21] Qi Zhang and Wei Wang. An efficient algorithm for approximate biased quantile computation in data streams. In *Proceedings of the 16th ACM Conference on Conference on Information and Knowledge Management*, CIKM '07, pages 1023–1026, New York, NY, USA, 2007. ACM.