THEORY OF COMPUTING, Volume 13 (14), 2017, pp. 1–18
www.theoryofcomputing.org

SPECIAL ISSUE: APPROX-RANDOM 2015

A Randomized Online Quantile Summary in $O((1/\varepsilon)\log(1/\varepsilon))$ Words

David Felber Rafail Ostrovsky*

Received November 18, 2015; Revised May 16, 2017; Published November 14, 2017

Abstract: A quantile summary is a data structure that approximates to ε error the order statistics of a much larger underlying dataset.

In this paper we develop a randomized online quantile summary for the cash register data input model and comparison data domain model that uses $O((1/\varepsilon)\log(1/\varepsilon))$ words of memory. This improves upon the previous best upper bound of $O((1/\varepsilon)\log^{3/2}(1/\varepsilon))$ by Agarwal et al. (PODS 2012). Further, by a lower bound of Hung and Ting (FAW 2010) no deterministic summary for the comparison model can outperform our randomized summary in terms of space complexity. Lastly, our summary has the nice property that $O((1/\varepsilon)\log(1/\varepsilon))$ words suffice to ensure that the success probability is at least $1-\exp(-\text{poly}(1/\varepsilon))$.

ACM Classification: F.2.2, E.1, F.1.2

AMS Classification: 68W20, 68W25, 68W27, 68P05

Key words and phrases: algorithms, data structures, data stream, approximation, approximation algorithms, online algorithms, randomized, summary, quantiles, RANDOM

A conference version of this paper appeared in the Proceedings of the 19th Internat. Workshop on Randomization and Computation (RANDOM 2015).

*Research supported in part by NSF grants CCF-0916574; IIS-1065276; CCF-1016540; CNS-1118126; CNS-1136174; US-Israel BSF grant 2008411, OKAWA Foundation Research Award, IBM Faculty Research Award, Xerox Faculty Research Award, B. John Garrick Foundation Award, Teradata Research Award, and Lockheed-Martin Corporation Research Award. This material is also based upon work supported by the Defense Advanced Research Projects Agency through the U.S. Office of Naval Research under Contract N00014-11-1-0392. The views expressed are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

© 2017 David Felber and Rafail Ostrovsky
Licensed under a Creative Commons Attribution License (CC-BY)
DOI: 10.4086/toc.2017.v013a014

1 Introduction

A quantile summary S is a fundamental data structure that summarizes an underlying dataset X of size n, in space much less than n. Given a query ϕ, S returns an element y of X such that the rank of y in X is (probably) approximately ϕn. Quantile summaries are used in sensor networks to aggregate data in an energy-efficient manner and in database query optimizers to generate query execution plans.

1.1 The model

Quantile summaries have been developed for a variety of different models and metrics. The data input model we consider is the standard online cash register streaming model. In this model there is a single machine at which computation takes place. Attached to the machine is a data stream that feeds items to the machine one at a time online. Time is measured in stream items received, so that the first item x_1 arrives at $t = 1$, the second at time $t = 2$, and so forth. After each x_t arrives, the machine may do any processing that it needs to do. Timestep t does not end, and time $t+1$ does not start, until this processing is complete. The set of all stream items received through time t is denoted X_t or $X(t)$. X_t can also be viewed as a (multi)set of the t-item prefix of the infinite stream X of all stream items that will be fed to the machine. In addition to receiving the data stream, the machine is also able to accept and answer queries. After stream item x_t arrives, the machine can be given queries ϕ, to which it responds within the same timestep t. The next item x_{t+1} does not arrive until any queries for timestep t have completed. Further, for randomized algorithms, the machine has a source of randomness from which it can sample random bits.

The data domain model we consider is the comparison model, in which stream items come from an arbitrary totally ordered domain $\mathcal{D}$ of stream items. $\mathcal{D}$ is specifically not required to be the integers. Stream items are indivisible tokens. The only operations permitted on stream items is to compare them and to copy, move, and delete them. In particular, stream items do not need to have any numeric value, and the comparison function can be an arbitrarily complex one.

The memory model we consider has two regions of memory. The first region can contain only stream items. Each copy of a stream item uses one unit of memory. When a new stream item x_t arrives from the data stream it arrives at a set location in this region of memory, and during processing can be copied to other locations in this region. When the next stream item x_{t+1} arrives, if x_t was not copied, then x_{t+1} overwrites the only copy of x_t, and therefore x_t is lost and cannot be retrieved. Stream items can also be deleted from this region, in which case they are also lost and cannot be retrieved.

The second region of memory is a scratch pad or bookkeeping area that can store numeric tokens and perform operations with and on those tokens. In particular, the operations permitted on the tokens are arithmetic, comparison, referencing locations of stream items in the first region of memory, and referencing locations of numeric tokens in the second region of memory. Also, to support randomness, the machine can in one operation sample a random bit into a fixed numeric token location in this second memory region. Each numeric token can take on a value in $\{0,1,2,\ldots,\max\}$, where max is $O(t)$ for a constant number of those tokens and $O(\text{poly}(1/\varepsilon))$ for the rest of the tokens.

For simplicity, we assume that there are items $\inf\mathcal{D}$ and $\sup\mathcal{D}$ in $\mathcal{D}$. If not, we can define two new items $\inf\mathcal{D}$ and $\sup\mathcal{D}$ that are lesser and greater than any item in $\mathcal{D}$, and simulate them in the scratch pad.

1.2 The problem

Our problem is defined by an error parameter $\varepsilon \leq 1/2$. The goal is to maintain in the machine at all times t a quantile summary S_t of the dataset X_t. A quantile summary is a data structure with two operations. The first is an update operation, that takes a summary S_{t-1} and a stream item x_t and returns an updated summary S_t. This update operation is performed in the processing phase of each timestep after stream item x_t arrives. The second operation is a query operation, that takes a summary S_t and a value ϕ in $(0,1]$, and returns a stream item $y = y(\phi)$ from the first memory region (and therefore from X_t) so that $|R(y,X_t) - \phi t| \leq \varepsilon t$, where $R(a,Z)$ is the *rank of item a in set Z*, defined as $|\{z \in Z : z \leq a\}|$.

For randomized quantile summaries, there is an additional parameter δ, and we only require that for any time t and any query ϕ we have that $P(|R(y,X_t) - \phi t| \leq \varepsilon t) \geq 1 - \delta$; that is, y's rank is only probably close to ϕt, not definitely close. For our main result we use a fixed $\delta = \mathrm{e}^{-\mathrm{poly}(1/\varepsilon)}$. It will be easier to deal with the rank directly, so we define $\rho = \phi t$ and use that in what follows.

In either case, the measure of an algorithm for the problem is the space required to store the quantile summary, meaning that the number of stream items should be small and also the number of numeric tokens should be small. The algorithm we develop here uses $O((1/\varepsilon)\log(1/\varepsilon))$ stream items and $O((1/\varepsilon)\log(1/\varepsilon))$ numeric tokens through time t. Calling each of these stream items or numeric tokens a memory word, our algorithm uses a total of $O((1/\varepsilon)\log(1/\varepsilon))$ words.

The quantile summary is maintained in memory. Any stream items in the data structure are stored in the first memory region that contains only stream items. Any bookkeeping information, such as the number t, is stored in the second memory region. However, to make our algorithm easy to understand we ignore this distinction in the algorithm's description.

Lastly, note that the problem of maintaining a small summary is only interesting in the online case, in which items arrive one by one. Offline, a trivial summary of X_n is the set of $1/\varepsilon$ items at ranks $\varepsilon n, 2\varepsilon n, 3\varepsilon n, \ldots, n$.

1.3 Previous and related work

The two most directly relevant pieces of prior work ([1] and [8]) are randomized online quantile summaries for the cash register/comparison model. Aside from oblivious sampling algorithms (which require storing $\Omega(1/\varepsilon^2)$ samples) the only other such work of which we are aware is an approach by Wang, Luo, Yi, and Cormode [12] that combines the methods of [1] and [8] into a hybrid with the same space bound as [1].

The newer of the two is that of Agarwal, Cormode, Huang, Phillips, Wei, and Yi [1]. Among other results, Agarwal et al. develop a randomized online quantile summary for the cash register/comparison model that uses $O((1/\varepsilon)\log^{3/2}(1/\varepsilon))$ words of memory. This summary has the nice property that any two such summaries can be combined to form a summary of the combined underlying dataset without loss of accuracy or increase in size.

The earlier such summary is that of Manku, Rajagopalan, and Lindsay [8], which uses $(1/\varepsilon)\log^2(1/\varepsilon)$ space. At a high level, their algorithm downsamples the input stream in a non-uniform way and feeds the downsampled stream into a deterministic summary, while periodically adjusting the downsampling rate.

We note here for those familiar with the result of Manku et al. that, while our algorithm at a high level may appear similar, there are important differences. We defer a discussion of similarities and differences to Section 4 after the presentation of our algorithm in Section 3.

For the comparison model, the best deterministic online summary to date is the (GK) summary of Greenwald and Khanna [4], which uses $O((1/\varepsilon)\log(\varepsilon n))$ space. This improved upon a deterministic (MRL) summary of Manku, Rajagopalan, and Lindsay [7] and a summary implied by Munro and Paterson [9], which use $O((1/\varepsilon)\log^2(\varepsilon n))$ space.

A more restrictive domain model than the comparison model is the bounded universe model, in which elements are drawn from the integers $\{1,\ldots,u\}$. For this model there is a deterministic online summary by Shrivastava, Buragohain, Agrawal, and Suri [10] that uses $O((1/\varepsilon)\log(u))$ space.

Between submission of our paper to Theory of Computing and its acceptance, a new method was developed by Karnin, Lang, and Liberty [6] that obtains matching upper and lower bounds for the problem of maintaining a randomized online summary. Their upper bound, $O((1/\varepsilon)\log\log(1/\delta))$, can be viewed as extending our upper bound of $O((1/\varepsilon)\log(1/\varepsilon))$ to arbitrary δ, which is particularly valuable when δ is large compared to $e^{-\text{poly}(1/\varepsilon)}$. Prior to [6] the best lower bound was a simple lower bound of $\Omega(1/\varepsilon)$ which intuitively comes from the fact that no one element can satisfy more than $2\varepsilon n$ different rank queries. For the deterministic version of the problem there is a lower bound of $\Omega((1/\varepsilon)\log(1/\varepsilon))$ by Hung and Ting [5].

1.4 Our contributions

In the next section we describe a simple $O((1/\varepsilon)\log(1/\varepsilon))$ streaming summary that is online except that it requires n to be given up front and that it is unable to process queries until it has seen a constant fraction of the input stream. This simple summary is not new (it is mentioned in Wang et al. [12], for example) but the discussion provides exposition for Section 3, in which we develop this summary into a fully online summary with the same asymptotic space complexity that can answer queries at any point in time. At that point we will have proven the following theorem, which constitutes our main result.

Theorem 1.1. *There is a randomized online quantile summary algorithm that runs in deterministic $O((1/\varepsilon)\log(1/\varepsilon))$ space and guarantees that, for any sequence X of items, for any n, and for any rank $\rho \le n$, the summary's response to query ρ just after receiving x_n will be an item $y \in X_n$ such that*

$$P(|R(y,X_n)-\rho| \le \varepsilon n) \ge 1-\exp(-1/\varepsilon)\,.$$

Finally, we close in Section 4 by examining the similarities and differences between our summary and previous work and discuss a design approach for similar streaming problems.

2 A simple streaming summary

Before we describe the algorithm we must first describe its two main components in a bit more detail than was used in the introduction. The two components are Bernoulli sampling and the GK summary [4].

2.1 Bernoulli sampling

Bernoulli sampling downsamples a stream X of size n to a sample stream S by choosing to include each next item into S with independent probability m/n. (As stated this requires knowing the size of X in

advance.) At the end of processing X, the expected size of S is m, and the expected rank of any sample y in S is $E(R(y,S)) = (m/n)R(y,X)$. In fact, for any times $t \leq n$ and partial streams X_t and S_t, where S_t is the sample stream of X_t, we have $E(|S_t|) = mt/n$ and $E(R(y,S_t)) = (m/n)R(y,X_t)$. To generate an estimate for $R(y,X_t)$ from S_t we use $\hat{R}(y,X_t) = (n/m)R(y,S_t)$. The following theorem bounds the probability that S is very large or that $\hat{R}(y,X_t)$ is very far from $R(y,X_t)$. A generalization of this theorem is due to Vapnik and Chervonenkis [11]; the proof of this special case is a simple known application of Chernoff bounds.

Theorem 2.1. *Let $0 \leq m \leq n$ and $0 < \varepsilon < 1$. For any time $t \geq n/64$,*

$$P(|S_t| > 2tm/n) < \exp(-m/192).$$

Further, for any time $t \geq n/64$ and any item y,

$$P(|\hat{R}(y,X_t) - R(y,X_t)| > \varepsilon t/8) < 2\exp(-\varepsilon^2 m/12288).$$

Proof. For the first part we use a Chernoff bound of

$$P(|S_t| > (1+\delta)E(|S_t|)) < \exp(-\delta E(|S_t|)/3).$$

Here, $E(|S_t|) = tm/n$, so $\delta = 1$, and

$$P(|S_t| > 2tm/n) < \exp(-tm/3n) < \exp(-m/192)$$

since $t \geq n/64$. For the second part,

$$P(|\hat{R}(y,X_t) - R(y,X_t)| > \varepsilon t/8) = P(|R(y,S_t) - E(R(y,S_t))| > \varepsilon tm/8n).$$

The Chernoff bound is

$$P(|R(y,S_t) - E(R(y,S_t))| > \delta E(R(y,S_t))) < 2\exp(-\min\{\delta,\delta^2\}E(R(y,S_t))/3).$$

Here, $\delta = \varepsilon t/8E(R(y,S_t))$. If $\delta^2 > \delta > 1$ then

$$P < 2\exp(-\varepsilon t/24) \leq 2\exp(-\varepsilon n/1536) \leq 2\exp(-\varepsilon^2 m/12288).$$

Otherwise, $\delta^2 \leq \delta \leq 1$, in which case

$$P < 2\exp(-\varepsilon^2 t^2 m/192nE(R(y,S_t))) \leq 2\exp(-\varepsilon^2 m/12288),$$

finishing the proof. □

This means that, given any $1 \leq \rho \leq t$, if we choose to return the sample $y \in S_t$ with $R(y,S_t) = \rho m/n$, then $R(y,X_t)$ is likely to be close to ρ, as long as m is $\Omega((1/\varepsilon^2)\log(1/\varepsilon))$

2.2 GK summary

The GK summary is a deterministic summary that can answer queries to ε error over any portion of the received stream. Let G_t be the summary after inserting the first t items X_t from stream X into G. Greenwald and Khanna guarantee in [4] that with only $O((1/\varepsilon)\log(\varepsilon t))$ words, given any $1 \leq \rho \leq t$, G_t can return an element $y \in X_t$ so that $|R(y,X_t) - \rho| \leq \varepsilon t/8$. We call this the *GK guarantee*.

2.3 A simple streaming summary

We combine Bernoulli sampling with the GK summary by downsampling the input data stream X to a sample stream S and then feeding S into a GK summary G. It looks like this.

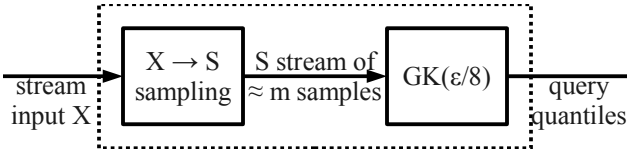

Figure 1: The big picture.

The key reason this gives us a small summary is that we never need to store S; each time we sample an item into S we immediately feed it into G. Therefore, we only use as much space as $G(S(X_t))$ uses. In particular, for $m = O(\text{poly}(1/\varepsilon))$, we use only $O((1/\varepsilon)\log(1/\varepsilon))$ words. To answer a query ρ for X_t, we scale ρ by m/n, ask $G(S(X_t))$ for that, and return the resulting sample y.

We formalize this intuition in the following lemma, which combines the ideas in the proof of Theorem 2.1 with the GK guarantee to yield approximation and correctness guarantees.

Lemma 2.2. *Fix some time $t \geq n/64$ and some rank $\rho \leq t$, and consider querying $G(S(X_t))$ with $q = \min\{\rho m/n, |S|\}$, obtaining y as the result. Say that $S = S(X_t)$ is* good *if*

$$\left||S| - mt/n\right| \leq \varepsilon mt/8n$$

and if none of the first $\leq mt/n$ samples z in S has

$$\left|R(z,S) - \frac{m}{n}R(z,X_t)\right| > \varepsilon mt/8n\,.$$

If S is good then

$$|R(y,X_t) - \rho| \leq \varepsilon t/2\,.$$

Further, if

$$m \geq \frac{400000\ln 1/\varepsilon}{\varepsilon^3}$$

then

$$P(S \text{ is not good}) \leq \varepsilon^3 \mathrm{e}^{-1/\varepsilon}/8\,.$$

Proof. First, by the GK guarantee, $G(S)$ returns some item y with $|R(y,S) - q| \leq \varepsilon t/8$. If S is good, then

$$\left|q - \frac{\rho m}{n}\right| \leq \frac{\varepsilon mt}{8n}\,, \quad \text{and also} \quad \left|R(y,S) - \frac{m}{n}R(y,X_t)\right| \leq \frac{\varepsilon mt}{8n}\,.$$

By the triangle inequality,

$$\left|\frac{m}{n}R(y,X_t) - \frac{\rho m}{n}\right| \leq \frac{3\varepsilon mt}{8n}\,.$$

Equivalently, $|R(y,X_t) - \rho| \leq 3\varepsilon t/8$.

Now, following the proof of Theorem 2.1, we have that

$$P(||S_t| - mt/n| > \varepsilon mt/8n) < 2\exp(-\varepsilon^2 m/12288)$$

and also for each of the first $\leq m$ samples z that

$$P(|R(z,S) - \tfrac{m}{n}R(z,X_t)| > \varepsilon mt/8n) < 2\exp(-\varepsilon^2 m/12288)\,.$$

By the union bound,

$$P(S \text{ is not good}) \leq 4m\exp(-\varepsilon^2 m/12288)\,.$$

Choosing

$$m \geq \frac{400000\ln 1/\varepsilon}{\varepsilon^3}$$

suffices to bound this quantity by $\varepsilon^3 \mathrm{e}^{-1/\varepsilon}/8$. □

2.4 Caveats

There are two serious issues with this summary. The first is that it requires us to know the value of n in advance to perform the sampling. Also, as a byproduct of the sampling, we can only obtain approximation guarantees after we have seen at least $1/64$ (or at least some constant fraction) of the items. This means that while the algorithm is sufficient for approximating order statistics over streams stored on disk, more is needed to get it to work for online streaming applications, in which (1) the stream size n is not known in advance, and (2) queries can be answered approximately at any time $t \leq n$ and not just when $t \geq n/64$.

Adapting this basic streaming summary idea to work online constitutes the next section and the bulk of our contribution. We start with a high-level overview of our online summary algorithm. In Section 3.1 we formally define an initial version of our algorithm whose expected size at any given time is $O((1/\varepsilon)\log(1/\varepsilon))$ words. In Section 3.2 we show that our algorithm guarantees that for any n and any ρ we have that $P(|R(y,X_n) - \rho| \leq \varepsilon n) \geq 1 - \exp(-1/\varepsilon)$. In Section 3.3 we discuss the slight modifications necessary to get a deterministic $O((1/\varepsilon)\log(1/\varepsilon))$ space complexity, and also perform a time complexity analysis.

3 An online summary

Our algorithm works in *rows*, which are illustrated in Figure 2 and Figure 3. Row r is a summary of the first $2^r 32m$ stream items. Since we do not know how many items will actually be in the stream, we cannot start all of these rows running at the outset. Therefore, we start each row $r \geq 1$ once we have seen $1/64$ of its total items. However, since we cannot save these items for every row we start, we need to construct an approximation of this fraction of the stream, which we do by using the summary of the previous row, and join this approximating stream with the new items that arrive while the row is live. We then wait until the row has seen a full half of its items before we permit it to start answering queries; this dilutes the influence of approximating the $1/64$ of its input that we could not store.

Operation within a row is very much like the operation of our fixed-n streaming summary. We feed the joint approximate prefix + new item stream through a Bernoulli sampler to get a sample stream, which

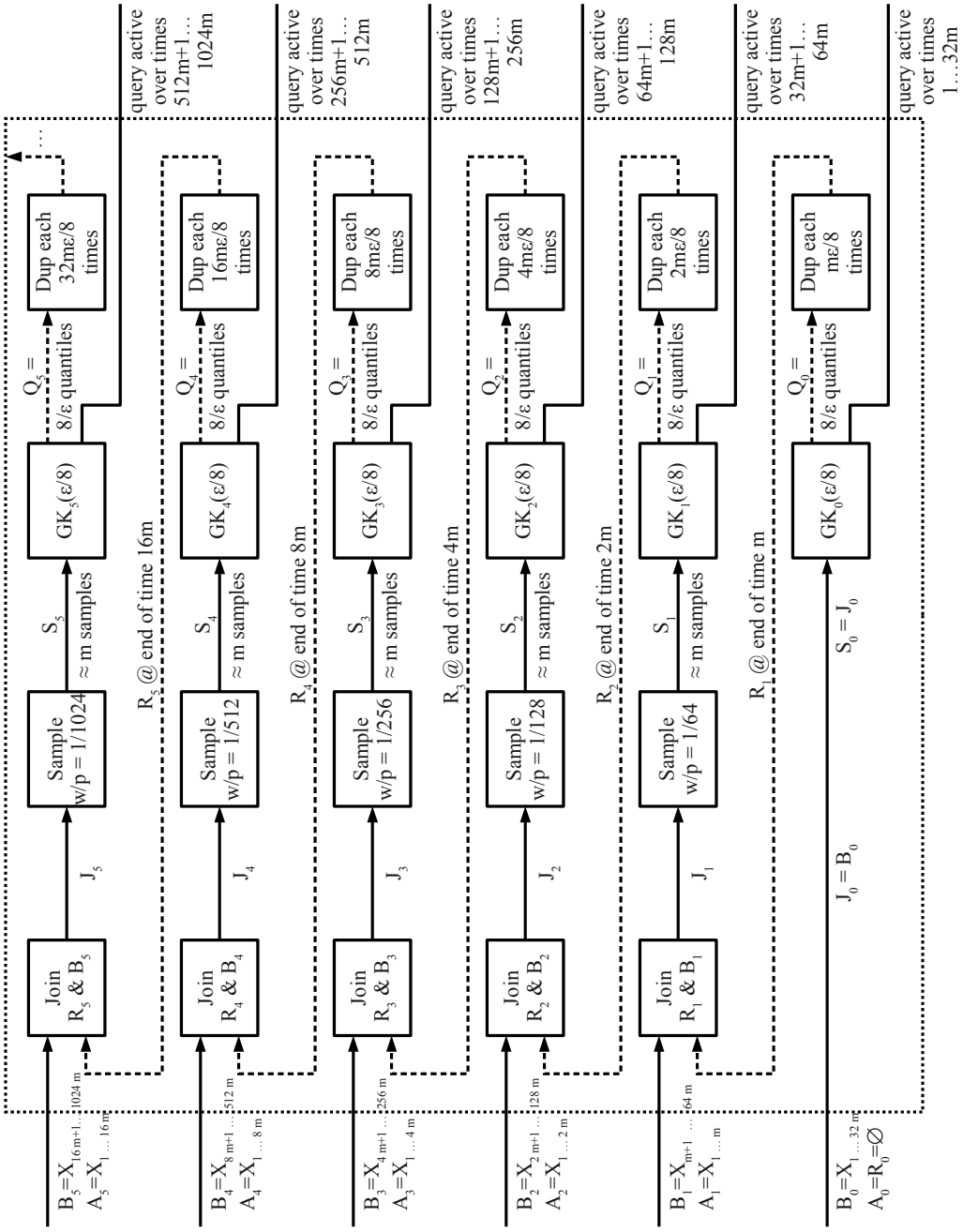

Figure 2: Each row r has its own copy G_r of the GK algorithm that approximates its input to $\varepsilon/8$ error. A_r is the prefix stream of row r, B_r is its suffix stream, R_r is its prefix stream replacement (generated by the previous row), J_r is the joint stream R_r followed by B_r, S_r is its sample stream, and Q_r is a one-time stream generated from G_r at time $2^r m$ to get the replacement prefix R_{r+1}.

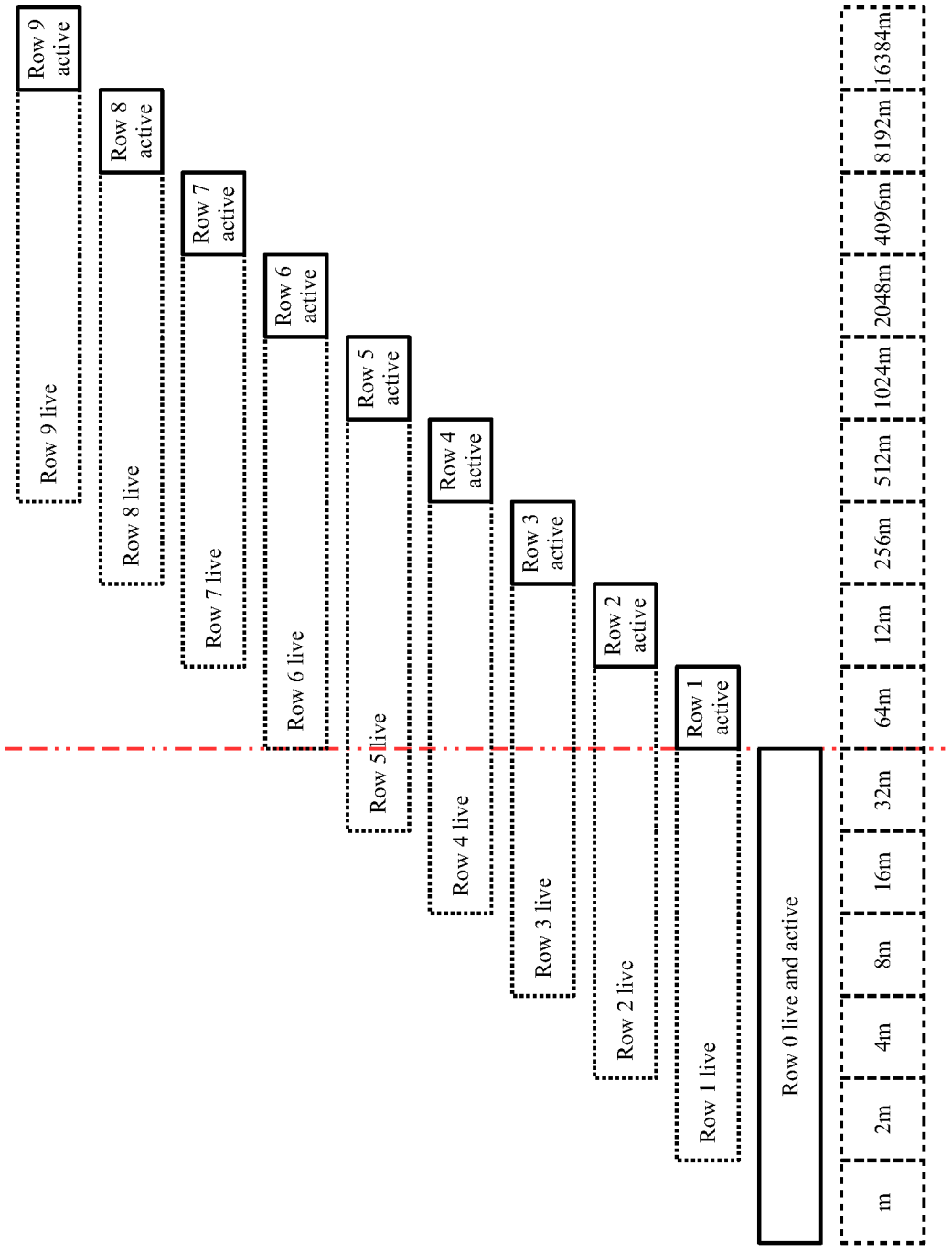

Figure 3: There are at most six live rows and one active row at any time. Time is shown here in dashed boxes at the bottom of the diagram on a logarithmic scale; for example, box $2m$ represents items $m+1,\ldots,2m$. The dashed red line at time $32m$ marks the first deletion of a row.

is then fed into a GK summary (which is stored). After row r has seen half of its items, its GK summary becomes the one used to answer quantile queries. When row $r+1$ has seen $1/64$ of *its* total items, row r generates an approximation of those items from its GK summary and feeds them as a stream into row $r+1$.

Row 0 is slightly different in order to bootstrap the algorithm. There is no join step since there is no previous row to join. Also, row 0 is active from the start. Lastly, we get rid of the sampling step so that we can answer queries over timesteps $1,\ldots,m/2$.

After the first $32m$ items, row 0 is no longer needed, so we can clean up the space used by its GK summary. Similarly, after the first $2^r 32m$ items, row r is no longer needed. The upshot of this is that we never need storage for more than six rows at a time. Since each GK summary uses $O((1/\varepsilon)\log(1/\varepsilon))$ words, the six live GK summaries also only use $O((1/\varepsilon)\log(1/\varepsilon))$ words.

Our error analysis, on the other hand, will require us to look back as many as $\Theta(\log 1/\varepsilon)$ rows to ensure our approximation guarantee. We stress that we will not need to actually *store* these $\Theta(\log 1/\varepsilon)$ rows for our guarantee to hold; we will only need that they did not have any bad events (as will be defined) when they *were* alive.

3.1 Algorithm description

Our algorithm works in rows. Each row r has its own copy G_r of the GK algorithm that approximates its input to $\varepsilon/8$ error. For each row r we define several streams: A_r is the prefix stream of row r, B_r is its suffix stream, R_r is its prefix stream replacement (generated by the previous row), J_r is the joint stream R_r followed by B_r, S_r is its sample stream, and Q_r is a one-time stream generated from G_r by querying it with ranks $\rho_1,\ldots,\rho_{8/\varepsilon}$, where $\rho_q = q(\varepsilon/8)(m/32)$ for $r \geq 1$ and $\rho_q = q\varepsilon m/8$ for $r = 0$.

The prefix stream $A_r = X(2^{r-1}m)$ for row $r \geq 1$, importantly, is not directly received by row r. Instead, at the end of timestep $2^{r-1}m$, row $r-1$ generates Q_{r-1} and duplicates each of those $8/\varepsilon$ items $2^{r-1}\varepsilon m/8$ times to get the replacement prefix R_r, which is then immediately fed into row r before timestep $2^{r-1}m+1$ begins.

Each row can be *live* or not and *active* or not. Row 0 is live in timesteps $1,\ldots,32m$ and row $r \geq 1$ is live in timesteps $2^{r-1}m+1,\ldots,2^r 32m$. Live rows require space; once a row is no longer live we can free up the space it used. Row 0 is active in timesteps $1,\ldots,32m$ and row $r \geq 1$ is active in timesteps $2^r 16m+1,\ldots,2^r 32m$. This definition means that exactly one row $r(t)$ is active in any given timestep t. Any queries that are asked in timestep t are answered by $G_{r(t)}$. Given query ρ, we ask $G_{r(t)}$ for $\rho/2^{r(t)}32$ (if $r \geq 1$) or for ρ (if $r = 0$) and return the result.

At each timestep t, when item x_t arrives, it is fed as the next item in the suffix stream B_r for each live row r. B_r joined with R_r defines the joined input stream J_r. For $r \geq 1$, J_r is downsampled to the sample stream S_r by sampling each item independently with probability $1/2^r 32$. For row 0, no downsampling is performed, so $S_0 = J_0$. Lastly, S_r is fed into G_r.

Figure 2 exhibits the operation of and the communication between the first six rows. Solid arrows indicate continuous streams and dashed arrows indicate one-time messages. Figure 3 shows when rows are created, change from live to active, and are deleted. Table 1 summarizes the variables used in this section. Algorithm 1 is a pseudocode listing of the algorithm.

Initially, allocate space for G_0. Mark row 0 as live and active.
for $t = 1, 2, \ldots$ **do**
  foreach *live row* $r \geq 0$ **do**
    with probability $1/2^r 32$ **do**
      Insert x_t into G_r.
  if $t = 2^{r-1}m$ *for some* $r \geq 1$ **then**
    Allocate space for G_r. Mark row r as live.
    Query G_{r-1} with $\rho_1, \ldots, \rho_{8/\varepsilon}$ to get $y_1, \ldots, y_{8/\varepsilon}$.
    for $q = 1, \ldots, 8/\varepsilon$ **do**
      for $1, \ldots, 2^{r-1}\varepsilon m/8$ **do**
        with probability $1/2^r 32$ **do**
          Insert y_q into G_r.
  if $t = 2^r 16m$ *for some* $r \geq 1$ **then**
    Unmark row $r-1$ as active and mark row r as active.
    Unmark row $r-1$ as live and free space for G_{r-1}.
on *query* ρ **do**
  Let $r = r(t)$ be the active row.
  Query G_r for rank $\rho/2^r 32$ (if $r \geq 1$) or for rank ρ (if $r = 0$).
  Return the result.

Algorithm 1: Procedural listing of the algorithm in Section 3.1.

3.2 Error analysis

Define $C_r = x(2^r 32m+1), x(2^r 32m+2), \ldots$ and Y_r to be R_r followed by B_r and then C_r. That is, Y_r is just the continuation of J_r for the entire length of the input stream.

Fix some time t. All of our claims will be relative to time t; that is, if we write S_r we mean $S_r(t)$. Our error analysis proceeds as follows. We start by proving that $R(y, Y_r)$ is a good approximation of $R(y, Y_{r-1})$ when certain conditions hold for S_{r-1}. By induction, this means that $R(y, Y_r)$ is a good approximation of $R(y, X = Y_0)$ when the conditions hold for all of $S_0, \ldots, S_{r-1}$, and actually it is enough for the conditions to hold for just $S_{r-\log 1/\varepsilon}, \ldots, S_{r-1}$ to get a good approximation. Having proven this claim, we then prove that the result $y = y(\rho)$ of a query to our summary has $R(y, X)$ close to ρ. Lastly, we show that $m = O(\text{poly}(1/\varepsilon))$ suffices to ensure that the conditions hold for $S_{r-\log 1/\varepsilon}, \ldots, S_{r-1}$ with very high probability $(1 - \mathrm{e}^{-1/\varepsilon})$. Table 1 summarizes the quantities used in this and the preceding section.

Lemma 3.1. *Let α_r be the event that $|S_r| > 2m$ and let β_r be the event that any of the first $\leq 2m$ samples z in S_r has*

$$|2^r 32 R(z, S_r) - R(z, Y_r)| > \frac{\varepsilon t}{8}.$$

Say that S_r is good *if neither α_r nor β_r occur (or if $r = 0$). For all rows $r \geq 1$ such that $t \geq t_r = 2^{r-1}m$, and all for all items y, if S_{r-1} is good then we have that*

$$|R(y, Y_r) - R(y, Y_{r-1})| \leq 2^r \varepsilon m.$$

Proof. At the end of time t_r we have $Y_r(t_r) = R_r(t_r)$, which is each item $y(\rho_q)$ in Q_{r-1} duplicated $\varepsilon t_r/8$ times. If $S_{r-1}(t_r)$ is good then $|R(y(\rho_q), Y_{r-1}(t_r)) - 2^{r-1}32\rho_q| \le \varepsilon t_r/2$ following Lemma 2.2.

Fix q so that $y(\rho_q) \le y < y(\rho_{q+1})$, where $y(\rho_0)$ and $y(\rho_{1+8/\varepsilon})$ are defined to be $\inf \mathcal{D}$ and $\sup \mathcal{D}$ for completeness. Fixing q this way implies that $R(y, Y_r(t_r)) = 2^{r-1}32\rho_q$. By the above bound on $R(y(\rho_q), Y_{r-1}(t_r))$ we also have that

$$2^{r-1}32\rho_q - \varepsilon t_r/2 \le R(y, Y_{r-1}(t_r)) < 2^{r-1}32\rho_{q+1} + \varepsilon t_r/2\,.$$

Recalling that $\rho_q = q\varepsilon m/256$, these bounds imply that

$$|R(y, Y_r(t_r)) - R(y, Y_{r-1}(t_r))| \le 2^r \varepsilon m\,.$$

For each time t after t_r, the new item x_t changes the rank of y in both streams Y_r and Y_{r-1} by the same additive offset, so

$$|R(y, Y_r) - R(y, Y_{r-1})| = |R(y, Y_r(t_r)) - R(y, Y_{r-1}(t_r))| \le 2^r \varepsilon m\,,$$

yielding the lemma. □

By applying this lemma inductively we can bound the difference between Y_r and $X = Y_0$ as follows.

Corollary 3.2. *For any $r \ge 1$ such that $t \ge t_r = 2^{r-1}m$, if all of $S_0(t_1), S_1(t_2), \ldots, S_{r-1}(t_r)$ are good, then $|R(y, Y_r) - R(y, X)| \le 2 \cdot 2^r \varepsilon m$.*

To ensure that all of these S_i are good would require m to grow with n, which would be bad, because the space complexity would also need to grow with n. Happily, it is enough to require only the last $\log_2 1/\varepsilon$ sample summaries to be good, since the other items we disregard constitute only a small fraction of the total stream.

Corollary 3.3. *Let $d = \log_2 1/\varepsilon$. For any $r \ge 1$ such that $t \ge t_r = 2^{r-1}m$, if all of $S_{r-1}(t_r), \ldots, S_{r-d}(t_{r-d+1})$ are good, then $|R(y, Y_r) - R(y, X)| \le 2^{r+2}\varepsilon m$.*

Proof. By Lemma 3.1 we have $|R(y, Y_r) - R(y, Y_{r-d})| \le 2^{r+1}\varepsilon m$. At time $t \ge t_{r-d}$, Y_{r-d} and X share all except possibly the first $2^{(r-d)-1}m = 2^{r-1}m/2^d = 2^{r-1}\varepsilon m$ items. Thus

$$|R(y, Y_r) - R(y, X)| \le |R(y, Y_r) - R(y, Y_{r-d})| + |R(y, Y_{r-d}) - R(y, X)| \le 2^{r+1}\varepsilon m + 2^r \varepsilon m\,,$$

proving the corollary. □

We now prove that if the last several sample streams were good then querying our summary will give us a good result.

Lemma 3.4. *Let $d = \log_2(1/\varepsilon)$ <– and $r = r(t)$. If all $S_r(t), S_{r-1}(t_r), \ldots, S_{r-d}(t_{r-d+1})$ are good, then querying our summary with rank ρ (= querying the active GK summary G_r with $\rho/2^r 32$ if $r \ge 1$, or with ρ if $r = 0$) returns $y = y(\rho)$ such that $|R(y, X) - \rho| \le \varepsilon t$.*

Proof. For $r \geq 1$ we have by Corollary 3.3 that $|R(y,Y_r) - R(y,X)| \leq 2^{r+2}\varepsilon m \leq \varepsilon t/2$. We apply Lemma 2.2 once more at row r, which tells us that $|R(y,Y_r) - \rho| \leq \varepsilon t/2$, and combine these bounds with the triangle inequality.

For $r = 0$, the GK guarantee alone proves the lemma. □

Lastly, we prove that $m = O(\text{poly}(1/\varepsilon))$ suffices to ensure that all of $S_r(t)$, $S_{r-1}(t_r), \ldots, S_{r-d}(t_{r-d+1})$ are good with probability at least $1 - \mathrm{e}^{-1/\varepsilon}$.

Lemma 3.5. *Let $d = \log_2 1/\varepsilon$ and $r = r(t)$. If*

$$m \geq \frac{400000 \ln 1/\varepsilon}{\varepsilon^3}$$

then all of $S_r(t), S_{r-1}(t_r), \ldots, S_{r-d}(t_{r-d+1})$ are good with probability at least $1 - \mathrm{e}^{-1/\varepsilon}$.

Proof. There are at most $1 + \log_2 1/\varepsilon \leq 4/\varepsilon$ of these summary streams total. Lemma 2.2 and the union bound give us

$$P(\text{some } S_r \text{ is bad}) \leq \frac{4}{\varepsilon}\frac{\varepsilon^3}{8}\mathrm{e}^{-1/\varepsilon} \leq \mathrm{e}^{-1/\varepsilon},$$

which implies our claim. □

3.3 Space and time complexity

A minor issue with the algorithm is that, as written in Section 3.1, we do not actually have a bound on the worst-case space complexity of the algorithm; we only have a bound on the space needed at any given point in time. This issue is due to the fact that there are low probability events in which $|S_r|$ can get arbitrarily large and the fact that over n items there are a total of $\Theta(\log n)$ sample streams. The space complexity of the algorithm is $O(\max|S_r|)$, and to bound this value with constant probability using the Chernoff bound appears to require that $\max|S_r| = \Omega(\log\log n)$, which is too big.

Fortunately, fixing this problem is simple. Instead of feeding every sample of S_r into the GK summary G_r, we only feed each next sample if G_r has seen $< 2m$ samples so far. That is, we deterministically restrict G_r to receiving only $2m$ samples. Lemmas 3.1 through Lemma 3.4 condition on the goodness of the sample streams S_r, which ensures that the G_r receive at most $2m$ samples each, and the claim of Lemma 3.5 is independent of the operation of G_r. Therefore, by restricting each G_r to receive at most $2m$ inputs we can ensure that the space complexity is deterministically $O((1/\varepsilon)\log(1/\varepsilon))$ without breaking our error guarantees.

The assumption in the streaming setting is that new items arrive over the input stream X at a high rate, so both the worst-case per-item processing time as well as the amortized time to process n items are important. For our per-item time complexity, the limiting factor is the duplication step that occurs at the end of each time $t_r = 2^{r-1}m$, which makes the worst-case per-item processing time as large as $\Theta(n)$. Instead, at time t_r we could generate Q_{r-1} and store it in $O(1/\varepsilon)$ words, and then on each arrival $t = 2^{r-1}m+1, \ldots, 2^r m$ we could insert both x_t and also the next item in R_r. By the time $t_{r+1} = 2t_r$ that we generate Q_r, all items in R_r will have been inserted into J_r. Thus the worst-case per-item time complexity

ε	The error parameter of our algorithm as a whole.
$\varepsilon/8$	The error parameter used in internal GK summaries.
m	The size of our random sample in each row. Following Lemma 2.2 we use $m \geq \frac{400000\ln(1/\varepsilon)}{\varepsilon^3}$.
$r = 0, 1, \ldots$	The current row.
$2^r 32m$	The number of stream items summarized by row r.
$2^{r\geq 1}m$	The time at which we create row r.
$X(2^r 32m)$	The first $2^r 32m$ items of X, which are also the stream items summarized by row r.
A_0	The prefix stream of row 0, which is empty.
$A_{r\geq 1}$	The prefix stream of row r, which contains $2^r m$ items. By the time $2^r m$ at which we create row r these items have already expired.
R_r	The replacement prefix of row r, which is generated by row $r-1$ to act as a proxy for the expired items A_r.
B_r	The suffix stream of row r, which is everything in $X(2^r 32m)$ except for A_r.
J_r	The result of joining R_r and B_r. J_r is to (R_r, B_r) as $X(2^r 32m)$ is to (A_r, B_r).
$S_{r\geq 1}$	The result of applying Bernoulli sampling on J_r at rate $1/2^r 32$. This rate is chosen so that the expected size of S_r is m.
Q_r	The result of querying the GK summary for row r with the $8/\varepsilon$ evenly-spaced queries $\varepsilon/8, 2\varepsilon/8, 3\varepsilon/8, \ldots, 1$. Each item in Q_r is duplicated $2^r m\varepsilon/8$ times to get R_{r+1}. The duplication number $2^r m\varepsilon/8$ is chosen so that R_{r+1} and A_{r+1} have the same size.
$q = 1, 2, \ldots, 8/\varepsilon$	The indices of evenly-spaced queries into Q_r for the current value of r.
ρ_q	The evenly-spaced rank queries into Q_r.
C_r	The continuation of the potentially infinite stream X beyond the items in B_r; that is, $x(2^r 32m+1), x(2^r 32m+2), \ldots$
Y_r	The potentially infinite stream that is the result of joining R_r, B_r, and C_r. Equivalently, the stream defined by replacing A_r in X with R_r.
$R(z, S_r)$	The rank of item z in the Bernoulli sample S_r.
$R(z, Y_r)$	The rank of item z in the stream Y_r.
$2^r 32R(z, S_r)$	This is $R(z, S_r)$ scaled up so that it is approximately the size of $R(z, Y_r)$.
α_r, β_r	Low-probability events that can kill our algorithm if they occur. We proved that these are low-probability events in Lemma 2.2
$y(\rho_q)$	The item returned by G_r on a query of ρ_q.

Table 1: A table summarizing the variables used in the algorithm description of Section 3.1 and the statements and proofs of Section 3.2.

is $O((1/\varepsilon)T_{\mathrm{GK}}^{\max})$, where $T_{\mathrm{GK}}^{\max}$ is the worst-case per-item time to query or insert into one of our GK summaries. Over $2^r 32m$ items there are at most $2m$ insertions into any one GK summary, so the amortized time over n items in either case is

$$O\left(\frac{m\log(n/m)}{n}T_{\mathrm{GK}}\right),$$

where T_{GK} is the amortized per-item time to query or insert into one of our GK summaries. Algorithm 2 includes the changes of this section, with the relevant lines highlighted to show the difference from Algorithm 1.

Initially, allocate space for G_0. Mark row 0 as live and active.
for $t = 1, 2, \ldots$ **do**
  foreach *live row* $r \geq 0$ **do**
    with probability $1/2^r 32$ **do**
      Insert x_t into G_r if G_r has seen $< 2m$ insertions.
    if $r \geq 1$ *and* $2^{r-1}m < t \leq 2^r m$ *and* G_r *has seen* $< 2m$ *insertions* **then**
      with probability $1/2^r 32$ **do**
        Also insert item $t - 2^{r-1}m$ of R_r into G_r.
  if $t = 2^{r-1}m$ *for some* $r \geq 1$ **then**
    Allocate space for G_r. Mark row r as live.
    Query G_{r-1} with $\rho_1, \ldots, \rho_{8/\varepsilon}$ to get $Q_{r-1} = y_1, \ldots, y_{8/\varepsilon}$.
    Store Q_{r-1}, to implicitly define R_r.
  if $t = 2^r 16m$ *for some* $r \geq 1$ **then**
    Unmark row $r-1$ as active and mark row r as active.
    Unmark row $r-1$ as live and free space for G_{r-1}.
on *query* ρ **do**
  Let $r = r(t)$ be the active row.
  Query G_r for rank $\rho/2^r 32$ (if $r \geq 1$) or for rank ρ (if $r = 0$).
  Return the result.

Algorithm 2: Procedural listing of the algorithm in Section 3.3. The changes between Section 3.1 and Section 3.3 are that G_r never has more than $2m$ insertions and that R_r is paired with items in B_r.

The modifications of this section notwithstanding, from a practical perspective we expect that the number of items required for our algorithm to begin to outperform a plain GK summary is likely to be prohibitive, even if we optimize the number of live rows and the value of m for any given ε, if we are required to maintain our provable guarantees.

4 Discussion

Our starting point is a very natural idea used in Manku et al. [8]: downsample the input stream and feed the resulting sample stream into a deterministic summary data structure (compare our Figure 1 with figure

1 on page 254 of [8]). At a very high level, we are simply replacing their deterministic $O((1/\varepsilon)\log^2(\varepsilon n))$ MRL summary [7] with the deterministic $O((1/\varepsilon)\log(\varepsilon n))$ GK summary [4].

However, our implementation of this idea differs conceptually from the implementation of Manku et al. in two important ways. First, we use the GK algorithm strictly as a black box, whereas Manku et al. peek into the internals of their MRL algorithm, using its algorithm-specific interface (NEW, COLLAPSE, OUTPUT) rather than the more generic interface (INSERT, QUERY). At an equivalent level, dealing with the GK algorithm is already unpleasant—the space complexity analysis in [4] is quite involved, and in fact a simpler analysis of the GK algorithm is an open problem [2]. Using the generic interface, our implementation could just as easily replace the GK boxes in the diagram in Figure 2 with MRL boxes; or, for the bounded universe model, with boxes running the q-digest summary of Shrivastava et al. [10].

The second way in which our algorithm differs critically from that of Manku et al. is that we operate on *streams* rather than on stream *items*. We use this approach in our proof strategy too; the key step in our error analysis, Lemma 3.1, is a statement about (what to us are) static objects, so we can trade out the complexity of dealing with time-varying data structures for a simple induction. We believe that developing streaming algorithms with analyses that hinge on analyzing streams rather than just stream items is likely to be a useful design approach for many problems.

Acknowledgements We thank the anonymous reviewers for *Theory of Computing* and *RANDOM 2015* for providing historical context, pointers and insights into previous and related work, and suggestions for clarifications. Research supported in part by NSF grant 1619348, DARPA, US-Israel BSF grant 2012366, OKAWA Foundation Research Award, IBM Faculty Research Award, Xerox Faculty Research Award, B. John Garrick Foundation Award, Teradata Research Award, and Lockheed-Martin Corporation Research Award. The views expressed are those of the authors and do not reflect position of the Department of Defense or the U.S. Government.

References

[1] 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, 2013. Preliminary version in PODS'12. [doi:10.1145/2500128] 3

[2] GRAHAM CORMODE: List of Open Problems in Sublinear Algorithms: Problem 2. https://sublinear.info/2. 16

[3] DAVID FELBER AND RAFAIL OSTROVSKY: A randomized online quantile summary in $O(1/\varepsilon \cdot \log(1/\varepsilon))$ words. In *Proc. 19th Internat. Workshop on Randomization and Computation (RANDOM'15)*, pp. 775–785. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2015. [doi:10.4230/LIPIcs.APPROX-RANDOM.2015.775]

[4] MICHAEL GREENWALD AND SANJEEV KHANNA: Space-efficient online computation of quantile summaries. In *2001 ACM SIGMOD Internat. Conf. on Managment of Data*, pp. 58–66. ACM Press, 2001. [doi:10.1145/375663.375670] 4, 5, 16

[5] REGANT Y. S. HUNG AND HINGFUNG F. TING: An $\Omega(1/\varepsilon \cdot \log 1/\varepsilon)$ space lower bound for finding ε-approximate quantiles in a data stream. In *Proceedings of the 4th International Conference on Frontiers in Algorithmics (FAW'10)*, pp. 89–100. Springer, 2010. [doi:10.1007/978-3-642-14553-7_11] 4

[6] ZOHAR KARNIN, KEVIN LANG, AND EDO LIBERTY: Optimal quantile approximation in streams. In *Proc. 57th FOCS*, pp. 71–78. IEEE Comp. Soc. Press, 2016. [doi:10.1109/FOCS.2016.17, arXiv:1603.05346] 4

[7] GURMEET SINGH MANKU, SRIDHAR RAJAGOPALAN, AND BRUCE G. LINDSAY: Approximate medians and other quantiles in one pass and with limited memory. In *1998 ACM SIGMOD Internat. Conf. on Managment of Data*, pp. 426–435. ACM Press, 1998. [doi:10.1145/276304.276342] 4, 16

[8] GURMEET SINGH MANKU, SRIDHAR RAJAGOPALAN, AND BRUCE G. LINDSAY: Random sampling techniques for space efficient online computation of order statistics of large datasets. In *1999 ACM SIGMOD Internat. Conf. on Managment of Data*, pp. 251–262. ACM Press, 1999. [doi:10.1145/304182.304204] 3, 15, 16

[9] J. IAN MUNRO AND MIKE PATERSON: Selection and sorting with limited storage. *Theoret. Comput. Sci.*, 12(3):315–323, 1980. Preliminary version in FOCS'78. [doi:10.1016/0304-3975(80)90061-4] 4

[10] NISHEETH SHRIVASTAVA, CHIRANJEEB BURAGOHAIN, DIVYAKANT AGRAWAL, AND SUBHASH SURI: Medians and beyond: New aggregation techniques for sensor networks. In *Proc. 2nd Internat. Conf. Embedded Networked Sensor Systems (SenSys'04)*, pp. 239–249. ACM Press, 2004. [doi:10.1145/1031495.1031524, arXiv:cs/0408039] 4, 16

[11] VLADIMIR N. VAPNIK AND ALEXEY YA. CHERVONENKIS: On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability & Its Applications*, 16(2):264–280, 1971. [doi:10.1137/1116025] 5

[12] LU WANG, GE LUO, KE YI, AND GRAHAM CORMODE: Quantiles over data streams: experimental comparisons, new analyses, and further improvements. *The VLDB Journal*, 25(4):449–472, 2016. Preliminary version in SIGMOD'13. [doi:10.1007/s00778-016-0424-7] 3, 4

AUTHORS

David Felber
Software engineer
Google, Inc.
dvfelber@ucla.edu

Rafail Ostrovsky
Professor
Department of Computer Science
Department of Mathematics
UCLA
Los Angeles, CA
rafail@cs.ucla.edu
http://web.cs.ucla.edu/~rafail/

ABOUT THE AUTHORS

David Felber completed his Ph. D. in 2015 at UCLA under the supervision of Rafail Ostrovsky. At press time he works as a software engineer at Google.

Rafail (Rafi) Ostrovsky is Professor of Computer Science and Professor of Mathematics at UCLA. He received his Ph. D. in computer science from MIT in 1992. He heads the multidisciplinary Center for Information and Computation Security at the Henry Samueli School of Engineering and Applied Science.