Approximate Counts and Quantiles over Sliding Windows

Arvind Arasu
Stanford University
arvinda@cs.stanford.edu

Gurmeet Singh Manku
Stanford University
manku@cs.stanford.edu

ABSTRACT

We consider the problem of maintaining ϵ-approximate counts and quantiles over a stream *sliding window* using limited space. We consider two types of sliding windows depending on whether the number of elements N in the window is fixed (*fixed-size* sliding window) or variable (*variable-size* sliding window). In a fixed-size sliding window, both the ends of the window slide synchronously over the stream. In a variable-size sliding window, an adversary slides the window ends independently, and therefore has the ability to vary the number of elements N in the window.

We present various deterministic and randomized algorithms for approximate counts and quantiles. All of our algorithms require $O(\frac{1}{\epsilon}\ \text{polylog}(\frac{1}{\epsilon}, N))$ space. For quantiles, this space requirement is an improvement over the previous best bound of $O(\frac{1}{\epsilon^2}\ \text{polylog}(\frac{1}{\epsilon}, N))$. We believe that no previous work on space-efficient approximate counts over sliding windows exists.

1. INTRODUCTION

Data streams arise in several application domains like high-speed networking, finance, and transaction logs. For many applications, recent elements of a stream are more important than those that arrived a long time ago. This preference for recent elements is commonly expressed using a *sliding window* [2], which identifies a portion of the stream that arrived between "now" and some recent time in the past.

Computation of various statistics over the current contents of the window is an important operation [3, 9, 14, 17]. For many statistics, computing an exact answer requires the storage of the entire current window, which is infeasible for many practical stream rates and window sizes. Also, exact answers are not crucial for many applications; *approximate* answers suffice.

This paper considers the problem of approximate, space-efficient computation of two important statistics over sliding windows—*frequency counts* and *quantiles*. We believe that frequency-counts over sliding windows have not been studied before. Approximate quantiles over sliding windows have been studied by Lin *et al* [17]. Our algorithms beat the algorithms by Lin *et al* in terms of space complexity.

PODS 2004 June 14-16, 2004, Paris, France.

2. DEFINITIONS

Frequency Counts

An *ϵ-approximate frequency count* (or simply *count*) for a bag[1] $\mathcal{B}$ of N elements is a set of $\langle e, \tilde{f}_e \rangle$ pairs ($e \in \mathcal{B}$) such that: (1) the approximate count $\tilde{f}_e$ is smaller than the true frequency f_e of e in $\mathcal{B}$, but by at most ϵN, i.e., $(f_e - \epsilon N) \leq \tilde{f}_e \leq f_e$; (2) any element $e \in \mathcal{B}$ with a frequency $f_e \geq \epsilon N$ appears in the set; other elements may or may not. Clearly, there could be more than one set that satisfies the above two properties.

Approximate counts are important because they can be used to solve approximate *frequent elements* problem, which has applications in data mining and network monitoring [18]. The frequent elements problem over a bag $\mathcal{B}$ seeks the set of elements of $\mathcal{B}$ whose frequency exceeds sN for a given threshold parameter s. The ϵ-approximate frequent elements problem allows some false positives—elements whose frequency is greater than $(s - \epsilon)$ can appear in the answer. However, no false negatives are allowed; all elements whose frequency is greater than sN must appear in the answer. It can be verified that an ϵ-approximate count for $\mathcal{B}$ can be used to find ϵ-approximate frequent elements of $\mathcal{B}$ if $s \geq \epsilon$.

Quantiles

Consider a bag $\mathcal{B}$ of N elements. The *rank* of an element is its position in a sorted arrangement of elements of $\mathcal{B}$. The rank of the minimum element is 1, while that of the maximum element is N. The *ϕ-quantile* ($\phi \in (0, 1]$) of $\mathcal{B}$ is the element with rank $\lceil \phi N \rceil$. The 0.5-quantile is called the *median*. An element is said to be an *ϵ-approximate ϕ-quantile* if its rank is between $\lceil (\phi - \epsilon)N \rceil$ and $\lceil (\phi + \epsilon)N \rceil$; clearly there could be many elements that qualify.

Streams and Sliding Windows

A *data stream* is a continuously arriving sequence of elements drawn from some ordered domain. At any point in time, a sliding window over a stream is a bag of last N elements

[1] A bag is simply a multi-set, allowing multiple occurrences of an element.

of the stream seen so far, for some nonnegative integer N. Our algorithms neither assume any advance knowledge of the domain, nor rely upon the distribution of elements.

We consider two types of sliding windows based on whether N is fixed (*fixed-size* sliding window) or variable (*variable-size* sliding window). Formally, both types of windows are modeled using an adversary who can insert a new element into the window or delete the oldest element from the window, at each time step. For fixed-size windows, the adversary is constrained so that each insertion is followed by a deletion, except at the beginning of the stream when the adversary is required to insert exactly N elements without a deletion. For variable-size windows, the adversary can arbitrarily interleave insertions and deletions, as long as deletions are performed over nonempty windows.

Example 1. Consider the stream $15, 7, 6, 24, 21, 24, \ldots$. A fixed-size sliding window over this stream with size $N = 3$ produces the following sequence of bags: $\{15\}$, $\{15, 7\}$, $\{15, 7, 6\}$, $\{7, 6, 24\}$, $\{6, 24, 21\}$, $\{24, 21, 24\}$, $\ldots$. One possible sequence of bags for a variable-size window over this stream is the following: $\{15\}$, $\{15, 7\}$, $\{7\}$, $\{7, 6\}$, $\{7, 6, 24\}$, $\{7, 6, 24, 21\}$, $\{6, 24, 21\}$, $\ldots$. □

Fixed- and variable-size sliding windows abstract the essential features of many common types of sliding windows like *tuple-based* windows [22], *time-based* windows [22], and *n-of-N* windows [17]. In other words, most algorithms designed for fixed- or variable-size windows can be easily adapted to work for the other types of windows. We present details on how our algorithms can be extended to these other window types in Section 8.

Problem Statement

The problem of maintaining a ϵ-approximate counts (resp. quantiles) over sliding windows can be stated as follows: Maintain "sufficient state" so that, at any point in time, ϵ-approximate counts (resp. ϕ-quantiles, for any $\phi \in (0, 1]$) over the current contents of the sliding window can be computed using the maintained state. The goal is to minimize the space required for maintaining the state.

An algorithm for maintaining ϵ-approximate statistics (counts/quantiles) has advance knowledge of ϵ. An algorithm over fixed-size windows has advance knowledge of N, the size of the window. The input to an algorithm is the sequence of new element insertions into the window and oldest element deletions from the window. For deletions, the identity of the oldest element is *not* provided as input to the algorithm: the algorithm is simply informed that the window boundary has moved.

3. RELATED WORK

Frequency Counting Algorithms

For data streams, the earliest deterministic algorithm for ϵ-approximate frequency counts is by Misra and Gries [21]. Their algorithm requires $\frac{1}{\epsilon}$ space and $O(1)$ amortized processing time per element. The same algorithm has been rediscovered recently by Demaine *et al* [10] and Karp *et al* [16], who reduced the processing time to $O(1)$ in the worst case. Manku and Motwani [18] presented LOSSY COUNTING, a deterministic algorithm that requires $O(\frac{1}{\epsilon} \log \epsilon N)$ space. Though the space requirements are worse than the Misra-Gries algorithm, LOSSY COUNTING is superior when the input is skewed. Further, the algorithm can be adapted to compute association rules [1] over data streams.

In a random sample of size $O(\frac{1}{\epsilon^2} \log (\epsilon\delta)^{-1})$, the relative frequency of any element in the sample differs from its true frequency in the base dataset by at most ϵ. This observation can be exploited to obtain approximate frequency counts in a single pass such that the space requirements are independent of N. For example, Toivonen [26] identifies a candidate set of frequent itemsets in the context of association rule mining [1]. For data streams, Manku and Motwani [18] presented STICKY SAMPLING, a randomized algorithm that requires only $O(\frac{1}{\epsilon} \log(\epsilon\delta)^{-1})$ space, beating the sampling bound. Cormode and Muthukrishnan [7] recently presented randomized algorithms for identifying frequency counts in the presence of both additions and deletions. Their algorithm is not directly applicable to sliding windows where the oldest element is *implicitly* deleted.

In this paper, we propose deterministic and randomized algorithms for approximate frequency counts over sliding windows. Our randomized algorithms are adaptations of STICKY SAMPLING. For fixed-size and variable-size windows, we require $O(\frac{1}{\epsilon} \log(\epsilon\delta)^{-1})$ and $O(\frac{1}{\epsilon} \log(\epsilon\delta)^{-1} \log \epsilon N)$ space respectively. Our deterministic algorithms use the Misra-Gries algorithm as a black-box. For fixed-size and variable-size windows, we require $O(\frac{1}{\epsilon} \log^2 \frac{1}{\epsilon})$ and $O(\frac{1}{\epsilon} \log^2 \frac{1}{\epsilon} \log \epsilon N)$ space respectively.

Deterministic Quantile-Finding Algorithms

The history of quantile-finding algorithms dates back to the early days of computer science. Early work focused on main memory datasets. The celebrated paper of Blum, Floyd, Pratt, Rivest and Tarjan[5], shows that selecting the kth largest element among N elements requires at least $1.5N$ and at most $5.43N$ comparisons. For an account of progress since then, see the survey by Paterson [24].

For large datasets in external memory, Pohl [25] established that any deterministic algorithm that computes the exact median in one pass needs to store at least $N/2$ data elements. Munro and Paterson [23] generalized the idea and showed that for $p \geq 2$, memory to store $\Theta(N^{1/p})$ elements is necessary and sufficient for finding the exact median (or any ϕ-quantile) in p passes.

For large-sized data streams, where only one pass is allowed, the lower bound of $N/2$ for the exact median [25] motivated the definition of *approximate* quantiles in the hope of reducing space to $o(N)$. Manku *et al* [19] defined the notion of ϵ-approximate quantiles and devised a deterministic one-pass algorithm that requires only $O(\frac{1}{\epsilon} \log^2 \epsilon N)$ space. However, the algorithm requires advance knowledge of an upper bound for N. Greenwald and Khanna [15] improved the space requirements to $O(\frac{1}{\epsilon} \log \epsilon N)$. Their algorithm does not require advance knowledge of N. Moreover, experiments indicate that their algorithm requires only $O(\frac{1}{\epsilon})$ space if the input is a random permutation of elements.

For sliding windows, Lin *et al* [17] recently devised the first algorithms for ϵ-approximate quantiles. Their space

bounds are $O(\frac{1}{\epsilon^2} + \frac{1}{\epsilon}\log(\epsilon^2 N))$ for fixed-size windows and $O(\frac{1}{\epsilon^2}\log^2(\epsilon N))$ for variable-size windows. In this paper, we improve upon both bounds, our space requirements being $O(\frac{1}{\epsilon}\log\frac{1}{\epsilon}\log N)$ and $O(\frac{1}{\epsilon}\log\frac{1}{\epsilon}\log\epsilon N\log N)$ respectively.

Randomized Quantile-Finding Algorithms

For identifying exact quantiles in main memory, a simple linear time randomized algorithm was presented by Floyd and Rivest [12]. For approximate quantiles, randomization reduces the space requirements significantly. The key insight is the well-known fact that the ϕ-quantile of a random sample of size $O(\frac{1}{\epsilon^2}\log(\epsilon\delta)^{-1})$ is an ϵ-approximate ϕ-quantile of N elements with probability at least $1-\delta$. This observation has been exploited, for example, by DeWitt *et al* [11] to identify splitters of large datasets in the context of distributed sorting. For data streams, a randomized quantile-finding algorithm was proposed by Manku *et al* [20] that requires only $O(\frac{1}{\epsilon}\log^2(\frac{1}{\epsilon}\log(\epsilon\delta)^{-1}))$ space, beating the sampling bound. Further improvement in space is possible, as described in Section 6. Recently Cormode and Muthukrishnan [8] proposed a *sketching* technique called CM-sketches that can be used to maintain ϵ-approximate quantiles over streams and updateable relations. Their approach requires an advance knowledge of the domain U from which the elements of the stream are drawn. The space requirement for their approach is $O(\frac{1}{\epsilon}\log^2|U|\log(|U|/(\epsilon\delta)))$, where $|U|$ denotes the size of the domain U.

Related Problems

A problem related to approximate counting is the top-k problem, also known as the *Hot Item* problem, where the goal is to identify k items which are most frequent. Algorithms for the problem over data streams have been developed by Charikar *et al* [6], Cormode and Muthukrishnan [7] and Gibbons and Matias [13].

Sliding window algorithms have been developed for a variety of problems: bit-counting (Datar *et al* [9]), sampling (Babcock *et al* [3]), variance and k-medians (Datar *et al* [4], distinct values and bit-counts (Gibbons and Tirthapura [14]) and quantiles (Lin *et al* [17]).

4. SLIDING-WINDOW SKETCHES

Our presentation focuses on the data structures maintained by our algorithms, which we call *sliding-window sketches*. In this section, we define two general classes of sliding window sketches called *unbounded-window sketches* and *bounded-window sketches*. We then describe how these sketches can be used to derive algorithms for maintaining approximate statistics over fixed- and variable-size windows.

Recall from Section 2 that a sliding window can be defined as a sequence of two operations: insertion of a new stream element into the window when it arrives, and deletion of the oldest element in the window.

Intuitively, a sliding-window sketch is maintained over a stream and allows some approximate statistic to be computed over a sliding window of the stream. Specifically, a sliding-window sketch supports three operations: (1) QUERY, which computes approximate statistics over the current contents of the window; (2) INSERT, which updates the sketch when a new element is inserted into the window; and (3) DELETE, which updates the sketch when the oldest element is deleted from the window.

For the definitions that follow, assume that the sketches are maintained over the stream of elements $e_1, e_2, e_3, \ldots$.

Definition 1. An *unbounded-window sketch* $\mathcal{S}$ is defined using a fixed error parameter ϵ, and is denoted $\mathcal{S}_\epsilon$. $\mathcal{S}_\epsilon$ allows ϵ-approximate statistics to be computed over a variable-size window of the stream. The contents of $\mathcal{S}_\epsilon$, when the current size of the stream is m and the current size of the window is N, is denoted $\mathcal{S}_\epsilon(m,N)$. $\mathcal{S}_\epsilon(m,N)$ supports three operations:

1. QUERY: Returns ϵ-approximate value of the maintained statistic over the bag of elements $\{e_{m-N+1}, e_{m-N+2}, \ldots, e_m\}$.
2. INSERT: Constructs $\mathcal{S}_\epsilon(m+1,N+1)$ from $\mathcal{S}_\epsilon(m,N)$ and the inserted element e_{m+1}.
3. DELETE: Constructs $\mathcal{S}_\epsilon(m,N-1)$ from $\mathcal{S}_\epsilon(m,N)$.

Definition 2. A *bounded-window sketch* $\mathcal{S}$ is defined using two parameters—an error parameter ϵ and a *max-window* parameter W—and is denoted $\mathcal{S}_{W,\epsilon}$. $\mathcal{S}_{W,\epsilon}$ allows approximate statistics to be computed over a variable-size window of the stream, but the size of the window N is constrained to be $\leq W$. The contents of $\mathcal{S}_{W,\epsilon}$, when the current size of the stream is m and the current size of the window is N, is denoted $\mathcal{S}_{W,\epsilon}(m,N)$. $\mathcal{S}_{W,\epsilon}(m,N)$ supports three operations:

1. QUERY: Returns $(\epsilon W/N)$-approximate value of the maintained statistic over the bag of elements $\{e_{m-N+1}, e_{m-N+2}, \ldots, e_m\}$.
2. INSERT: Constructs $\mathcal{S}_{W,\epsilon}(m+1,N+1)$ from $\mathcal{S}_{W,\epsilon}(m,N)$ and the inserted element e_{m+1}, if $N+1 \leq W$.
3. DELETE: Constructs $\mathcal{S}_{W,\epsilon}(m,N-1)$ from $\mathcal{S}_{W,\epsilon}(m,N)$.

Both bounded- or unbounded-window sketches allows approximate statistics to be computed over variable-size windows. For bounded-window sketches, the size of the window is constrained to be less than a fixed max-window parameter W, while there are no such constraints for unbounded-window sketches.

We can use bounded- and unbounded-window sketches to derive algorithms for maintaining statistics over sliding windows as follows: An ϵ-approximate algorithm over a fixed-size window of size N uses a bounded-window sketch $\mathcal{S}_{N,\epsilon}$ (i.e., the max-window parameter $W=N$). Insertions to the window are handled using the INSERT operation and deletions using the DELETE operation of the sketch. From the definition of fixed-size windows, it follows that after $m \geq N$

L	$= \log_2(4/\epsilon)$	Highest level
ϵ_ℓ	$= \frac{\epsilon}{2(2L+2)} 2^{L-\ell}$	Error at level ℓ
N_ℓ	$= \frac{\epsilon W}{4} 2^\ell$	Size of level-ℓ block

Table 1: List of symbols used by $\mathcal{C}_{W,\epsilon}$ and $\mathcal{Q}_{W,\epsilon}$.

elements of the stream have been seen, the sketch maintained by the algorithm is of the form $\mathcal{S}_{N,\epsilon}(m, N)$. By definition, this sketch allows $\epsilon N/N = \epsilon$-approximate statistics to be computed over the previous N elements[2].

Similarly, an ϵ-approximate algorithm over variable-size windows uses an unbounded-window sketch $\mathcal{S}_\epsilon$, and handles insertions and deletions from the window using the INSERT and DELETE operations, respectively. Therefore, when the current size of the stream is m and the current size of the window is N, the sketch maintained by the algorithm is $\mathcal{S}_\epsilon(m, N)$. By definition, this sketch allows ϵ-approximate statistics to be computed over the current elements in the window.

Note that the definition of bounded-window sketches is more general than required by our fixed-size window algorithms. For example, $\mathcal{S}_{W,\epsilon}(m, N)$ allows N to vary between 0 and W, while the fixed-size window algorithms always maintain $W = N$. The generality is in fact, exploited for constructing unbounded-window sketches from bounded-window sketches, as we will describe in Section 7.

We can define randomized versions of these two types of sketches in a straightforward manner: A *randomized bounded-window sketch* $\mathcal{S}_{W,\epsilon,\delta}(m, N)$ is similar to a bounded-window sketch $\mathcal{S}_{W,\epsilon}(m, N)$ except that the approximation guarantee of the QUERY operation holds with probability at least $(1 - \delta)$. A *randomized unbounded-window sketch* $\mathcal{S}_{\epsilon,\delta}(m, N)$ is defined similarly.

5. DETERMINISTIC, BOUNDED-WINDOW SKETCHES

In this section, we present two deterministic, bounded-window sketches for counts and quantiles called $\mathcal{C}_{W,\epsilon}$ and $\mathcal{Q}_{W,\epsilon}$, respectively, where W denotes the maximum window width and ϵ the error parameter.

We assume that both $1/\epsilon$ and W are powers of 2 to avoid floors and ceilings in expressions. In order to design $\mathcal{C}_{W,\epsilon}$ for arbitrary W and ϵ, we identify W' and ϵ' such that: (a) W' and $1/\epsilon'$ are powers of 2; (b) $W \leq W' \leq 2W$; and (c) $W'\epsilon' \leq W\epsilon$. We then use the sketch $\mathcal{C}_{W',\epsilon'}$ in the place of $\mathcal{C}_{W,\epsilon}$. By construction, $\mathcal{C}_{W',\epsilon'}$ is more accurate than $\mathcal{C}_{W,\epsilon}$, and we can show that both have the same asymptotic space complexity. We also assume that $W \gg (1/\epsilon)$. Otherwise, a simple construction of $\mathcal{C}_{W,\epsilon}$ and $\mathcal{Q}_{W,\epsilon}$ is to store all the elements in the current window, and use these to compute exact counts and quantiles.

[2]The sketch does not allow ϵ-approximate statistics to be computed until $m \geq N$ elements are seen. In order to be able to compute ϵ-approximate sketches at all times, the algorithm should use an unbounded-sketch for the first N elements and switch to a bounded-sketch subsequently.

Let S denote the stream over which the sketches $\mathcal{C}_{W,\epsilon}$ and $\mathcal{Q}_{W,\epsilon}$ are maintained, and let $e_1, e_2, \ldots$ denote the sequence of elements of S. Therefore, $\mathcal{C}_{W,\epsilon}(m, N)$ (resp. $\mathcal{Q}_{W,\epsilon}(m, N)$) denotes the contents of the sketch $\mathcal{C}_{W,\epsilon}$ (resp. $\mathcal{Q}_{W,\epsilon}$) when the current size of S is m and the current window size is N.

$\mathcal{C}_{W,\epsilon}$ and $\mathcal{Q}_{W,\epsilon}$ conceptually make $L+1$ copies of the stream, where $L = \log_2(4/\epsilon)$. We say that these copies are at different *levels*, which are numbered sequentially as $0, 1, \ldots, L$. For each level, the copy of the stream for that level is divided into non-overlapping *blocks*. Within each level-ℓ, all blocks have the same size. We denote the size of level-ℓ blocks by N_ℓ. We set $N_0 = \frac{\epsilon W}{4}$[3] and for $\ell \geq 1$, $N_\ell = 2N_{\ell-1}$. It follows that $N_\ell = 2^\ell N_0 = \frac{\epsilon W}{4} 2^\ell$. Note that $N_L = W$. Within a level, blocks are numbered $0, 1, 2, \ldots$; smaller numbered blocks contain older elements. For example, in level-0 block 0 contains the first $\frac{\epsilon W}{4}$ elements, block 1 the next $\frac{\epsilon W}{4}$, and so on. In general, block b in level-ℓ contains the bag of elements e_i, $i \in [b2^\ell \frac{\epsilon W}{4} + 1, (b+1)2^\ell \frac{\epsilon W}{4}]$. Figure 1 schematically illustrates blocks and levels.

At any given point in time, a block is assigned to one of four states, depending on m, the number of elements of S so far, and N, the current window size. Consider block b of level-ℓ, which contains the bag of elements e_i, $i \in [b2^\ell \frac{\epsilon W}{4} + 1, (b+1)2^\ell \frac{\epsilon W}{4}] = [l, r]$. This block is ACTIVE if all its elements belong to the current window ($m - N < l < r \leq m$), EXPIRED if at least one of its elements is older than the last N elements ($l \leq m - N$), UNDER-CONSTRUCTION if some of its elements belong to the current window, and all the remaining, yet to arrive ($m - N < l \leq m$ and $r > m$), and INACTIVE if none of its elements has arrived ($l > m$). Each block (upto level-L) goes through the sequence of states INACTIVE, UNDER-CONSTRUCTION, ACTIVE, and EXPIRED.

$\mathcal{C}_{W,\epsilon}(m, N)$ and $\mathcal{Q}_{W,\epsilon}(m, N)$ are collections of *block-level sketches*, one sketch per block that is currently ACTIVE or UNDER-CONSTRUCTION (i.e., when the current size of the stream is m and the current size of the window is N).

A sketch for a level-ℓ ACTIVE block has an associated error parameter $\epsilon_\ell = \frac{\epsilon}{2(2L+2)} 2^{(L-\ell)}$, and allows ϵ_ℓ-approximate statistics (counts/quantiles) to be computed over the elements belonging to that block. The error parameter decreases as the level increase, so sketches for higher level blocks are more accurate than sketches for lower level blocks. The sketches for the ACTIVE blocks are *merged* by the QUERY operation to produce approximate statistics (count/quantiles) over the current contents of the window. The details of the block-level sketches and the merge operation differ for $\mathcal{C}_{W,\epsilon}(m, N)$ and $\mathcal{Q}_{W,\epsilon}(m, N)$, and we present them separately in Sections 5.1 and 5.2 respectively.

In order to construct a sketch for a block when it becomes ACTIVE, $\mathcal{C}_{W,\epsilon}$ and $\mathcal{Q}_{W,\epsilon}$ run a traditional one-pass algorithm (Misra-Gries [21] for $\mathcal{C}_{W,\epsilon}$ and Greenwald-Khanna [15] for $\mathcal{Q}_{W,\epsilon}$) over the elements of a block when it is in state UNDER-CONSTRUCTION. When all the elements of a block arrive (i.e., the state of the block changes from UNDER-CONSTRUCTION

[3]There is no sanctity to the number 4 in $\frac{\epsilon W}{4}$. It can be replaced with a different constant provided other constants in related expressions are changed suitably.

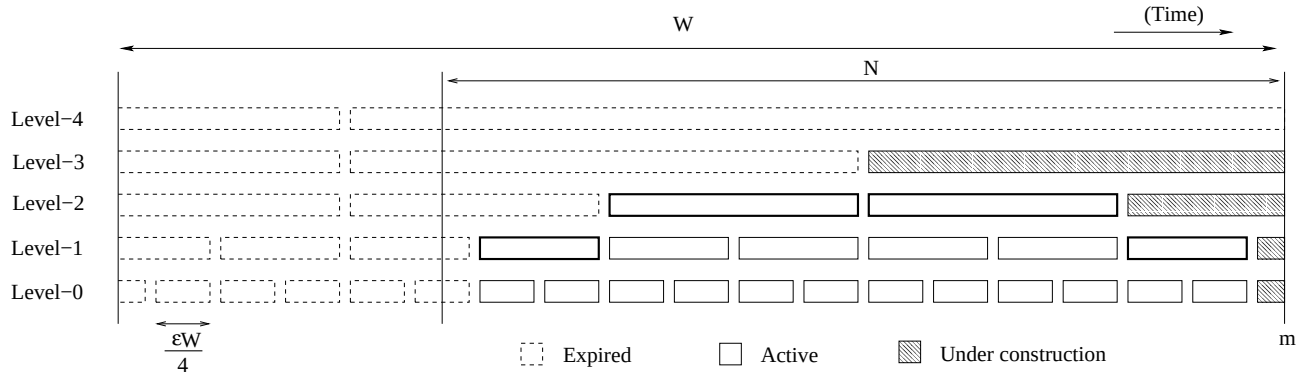

Figure 1: Levels and Blocks used in $\mathcal{C}_{W,\epsilon}(m,N)$ and $\mathcal{Q}_{W,\epsilon}(m,N)$

to ACTIVE), the one-pass algorithm is "queried" to produce a sketch for the block. Again, the details of running the one-pass algorithm and querying it differ for $\mathcal{C}_{W,\epsilon}$ and $\mathcal{Q}_{W,\epsilon}$, and we present them separately in Sections 5.1 and 5.2. Note that it is sufficient to start the one-pass algorithm for a block when the first element of the block arrives, i.e., when the block changes from the INACTIVE to the UNDER-CONSTRUCTION state. In order to implement this conceptual operation, $\mathcal{C}_{W,\epsilon}$ and $\mathcal{Q}_{W,\epsilon}$ maintain sketches corresponding to blocks UNDER-CONSTRUCTION. The sketch for a block is essentially the state maintained by the instance of the one-pass algorithm for that block.

5.1 Details of $\mathcal{C}_{W,\epsilon}$

Recall that $\mathcal{C}_{W,\epsilon}(m,N)$ is a collection of sketches corresponding to blocks that are ACTIVE or UNDER-CONSTRUCTION, when the current length of the stream is m and the current window size is N.

The sketch stored by $\mathcal{C}_{W,\epsilon}(m,N)$ for a level-ℓ ACTIVE block is just an ϵ_ℓ-approximate count over the elements belonging to the block. In other words, the sketch is a set of $\langle e, \tilde{f}_e\rangle$ pairs, where e denotes an element belonging to the block and $\tilde{f}_e$ denotes an approximate frequency count for the element. Further the set satisfies the following two properties: (1) the approximate count $\tilde{f}_e$ is smaller than the true frequency f_e of element e in the block, but by at most $\epsilon_\ell N_\ell$, i.e., $(f_e - \epsilon_\ell N_\ell) \leq \tilde{f}_e \leq f_e$; (2) any element e belonging to the block with a frequency $f_e \geq \epsilon_\ell N_\ell$ appears in the set.

In order to construct a sketch for a level-ℓ block when it becomes ACTIVE, $\mathcal{C}_{W,\epsilon}$ runs an instance of Misra-Gries algorithm over the elements of the block with error parameter ϵ_ℓ, when the block is UNDER-CONSTRUCTION. When the block becomes ACTIVE, $\mathcal{C}_{W,\epsilon}$ stores the output of the Misra-Gries algorithm as the sketch for the block. The output is simply the state used by the Misra-Gries algorithm, which is known to be $O(\frac{1}{\epsilon})$ [21]. Thus the space requirements for both ACTIVE and UNDER-CONSTRUCTION blocks at level ℓ are $O(\frac{1}{\epsilon_\ell})$.

LEMMA 1. *(Merge Operation for Counts) Using a collection of s sketches over disjoint bags of $N_1, N_2, \ldots N_s$ elements each, with error parameters $\epsilon_1, \epsilon_2, \ldots \epsilon_s$, respectively, we can compute an ϵ-approximate count for the union of the bags with error parameter $\epsilon = \frac{\epsilon_1 N_1 + \epsilon_2 N_2 + \cdots + \epsilon_s N_s}{N_1 + N_2 + \cdots + N_s}$.*

PROOF. Let $\mathcal{B}_1, \ldots, \mathcal{B}_s$ denote the s bags and $\mathcal{A}_1, \ldots, \mathcal{A}_s$ denote the s sketches over these bags. By definition, each $\mathcal{A}_i$ is an ϵ_i-approximate count over the elements of $\mathcal{B}_i$. We construct an output approximate count $\mathcal{A}$ for the union of the bags as follows: An element e occurs as part of $\mathcal{A}$ *iff* e occurs as part of some $\mathcal{A}_i$ $(1 \leq i \leq s)$, i.e., $\mathcal{A}_i$ contains a pair of the form $\langle e, \tilde{f}_{ei}\rangle$. If so, $\mathcal{A}$ contains the pair $\langle e, \tilde{f}_e\rangle$. The approximate count $\tilde{f}_e$ is the sum of the approximate counts $\tilde{f}_{ei}$ of e stored in each $\mathcal{A}_i$. (If e does not occur in $\mathcal{A}_i$, $\tilde{f}_{ei}$ is defined to be 0.) By definition of $\mathcal{A}_i$, $f_{ei} - \epsilon_i N_i \leq \tilde{f}_{ei} \leq f_{ei}$, where f_{ei} is the true frequency of e in bag $\mathcal{B}_i$. It follows that $f_e - (\epsilon_1 N_1 + \ldots + \epsilon_s N_s) \leq \tilde{f}_e \leq f_e$, where $f_e = f_{e1} + \ldots + f_{es}$ is the true frequency of e in the union of the bags. If the (true) frequency f_e of an element e in the union of the bags is at least $\epsilon_1 N_1 + \ldots + \epsilon_s N_s$, then the frequency f_{ei} of e in at least one of the bags $\mathcal{B}_i$ exceeds $\epsilon_i N_i$, implying that e occurs in $\mathcal{A}_i$. By construction, e occurs in $\mathcal{A}$ as well. Therefore $\mathcal{A}$ is an approximate count for the union of the bags with error parameter $\frac{\epsilon_1 N_1 + \epsilon_2 N_2 + \cdots + \epsilon_s N_s}{N_1 + N_2 + \cdots + N_s}$. □

THEOREM 1. *$\mathcal{C}_{W,\epsilon}(m,N)$ allows $\frac{\epsilon W}{N}$-approximate counts to be computed over the bag of elements $\{e_{m-N+1}, \ldots, e_m\}$. Further, $\mathcal{C}_{W,\epsilon}(m,N)$ uses $O(\frac{1}{\epsilon}\log^2\frac{1}{\epsilon})$ space.*

PROOF. *Approximation*: Let $\mathcal{B}_w = \{e_{m-N+1}, \ldots, e_m\}$ denote the bag of elements in the current window. Let α be the smallest integer such that $\alpha\frac{\epsilon W}{4} \geq m - N$. Let β be the largest integer such that $\beta\frac{\epsilon W}{4} \leq m$. Let $\mathcal{B}_{ig}$ (for bag of ignored elements) denote the bag of all elements e_i such that $m - N < i \leq \alpha\frac{\epsilon W}{4}$ or $\beta\frac{\epsilon W}{4} < i \leq m$. Intuitively, $\mathcal{B}_{ig}$ denotes the bag of elements that belong to the current window but do not belong to any ACTIVE block. Let $\mathcal{B}_{con}$ (for considered elements) denote the bag of all elements e_i such that $\alpha\frac{\epsilon W}{4} + 1 \leq i \leq \beta\frac{\epsilon W}{4}$. $\mathcal{B}_{con}$ denotes the bag of elements belonging to the current window that also belong to some ACTIVE block. Note that $\mathcal{B}_w = \mathcal{B}_{ig} \cup \mathcal{B}_{con}$.

We claim without proof that $\mathcal{B}_{con}$ can be expressed as a union of k non-overlapping ACTIVE blocks such that $k \leq (2L+2)$. (See Figure 1 for an illustration.) Let $\mathcal{B}_1, \ldots, \mathcal{B}_k$ denote the bag of elements belonging to these blocks. Let $\mathcal{A}_1, \ldots, \mathcal{A}_k$ denote the sketches stored by $\mathcal{C}_{W,\epsilon}(m,N)$ for these blocks. Let $\mathcal{A}_{ig}$ denote the trivial (empty set) sketch for the bag of elements $\mathcal{B}_{ig}$ with error parameter $\epsilon_{ig} = 1$. We

use Lemma 1 to compute an approximate count $\mathcal{A}$ over the bag of elements $\mathcal{B}_w = \mathcal{B}_1 \cup \ldots \cup \mathcal{B}_k \cup \mathcal{B}_{ig}$ using the sketches $\mathcal{A}_1, \ldots, \mathcal{A}_k, \mathcal{A}_{ig}$.

We now prove that $\mathcal{A}$ is an $\frac{\epsilon W}{N}$-approximate count for $\mathcal{B}_w$. Let ϵ_i denote the approximation parameter for sketch $\mathcal{A}_i$. Let N_i denote the number of elements in $\mathcal{B}_i$, and let N_{ig} denote the number of elements in $\mathcal{B}_{ig}$. We first note that for any level-ℓ, $\epsilon_\ell N_\ell = \frac{\epsilon W}{2(2L+2)}$, which is independent of ℓ. Since each $\mathcal{A}_i$ is some block-level sketch, $\epsilon_i N_i = \frac{\epsilon W}{2(2L+2)}$ for $(1 \leq i \leq k)$. Using Lemma 1 the approximation parameter for $\mathcal{A}$ (say $\epsilon_\mathcal{A}$) is given by:

$$\begin{aligned}
\epsilon_\mathcal{A} &= (\epsilon_{ig} N_{ig} + \sum_{i=1}^{k} \epsilon_i N_i)/N && (1)\\
&= (N_{ig} + k \cdot \frac{\epsilon W}{2(2L+2)})/N && (2)\\
&\leq (N_{ig} + \frac{\epsilon W}{2})/N && (3)\\
&\leq (\frac{\epsilon W}{2} + \frac{\epsilon W}{2})/N && (4)\\
&\leq \frac{\epsilon W}{N} && (5)
\end{aligned}$$

Equation 3 follows from the fact that $k \leq (2L+2)$, and Equation 4 from the fact that $N_{ig} \leq 2\frac{\epsilon W}{4}$ (which can be shown from the definition of $\mathcal{B}_{ig}$).

Space requirement: The space required by $\mathcal{C}_{W,\epsilon}(m, N)$ is the space required to store the block-level sketches for blocks that are currently ACTIVE or UNDER-CONSTRUCTION. As we argued earlier, a sketch for a level-ℓ block that is either ACTIVE or UNDER-CONSTRUCTION requires $O(\frac{1}{\epsilon_\ell})$ space.

From the definition of bounded-window sketches (Definition 2), we have $N \leq W$. Therefore there are at most $\frac{N}{\epsilon W/4} \leq \frac{4}{\epsilon}$ ACTIVE blocks at level-0, $\frac{2}{\epsilon}$ ACTIVE blocks at level-1, and so on. In general, we can show that there are at most $2^{L-\ell}$ ACTIVE blocks at level-ℓ. Hence, the total space required to store the sketches corresponding to all the ACTIVE blocks is $O(\sum_{\ell=0}^{L} 2^{L-\ell}/\epsilon_\ell)$. Substituting $\epsilon_\ell = \frac{\epsilon}{2(2L+2)} 2^{(L-\ell)}$, the required space is $O(\sum_{\ell=0}^{L} 2(2L+2)/\epsilon)$, which is $O(\sum_{\ell=0}^{L} \frac{L}{\epsilon}) = O(\frac{L^2}{\epsilon}) = O(\frac{1}{\epsilon}\log^2\frac{1}{\epsilon})$.

By definition, there is at most one block that is UNDER-CONSTRUCTION for each level. Therefore, the total space required for sketches corresponding to blocks UNDER CONSTRUCTION is $O(\sum_{\ell=0}^{L} 1/\epsilon_\ell)$. Since, $\epsilon_\ell \propto 2^{-\ell}$, $O(\sum_{\ell=0}^{L} 1/\epsilon_\ell)$ is geometric and is $O(\frac{1}{\epsilon_L}) = O(\frac{1}{\epsilon}\log\frac{1}{\epsilon})$. Therefore the total space required by $\mathcal{C}_{W,\epsilon}(m, N)$ is $O(\frac{1}{\epsilon}\log^2\frac{1}{\epsilon})$. □

5.2 Details of $\mathcal{Q}_{W,\epsilon}$

When a level-ℓ block is under UNDER-CONSTRUCTION, $\mathcal{Q}_{W,\epsilon}$ conceptually runs an instance of the Greenwald-Khanna algorithm [15] with error parameter $\frac{\epsilon_\ell}{2}$ over the elements of the block. When the block becomes ACTIVE, $\mathcal{Q}_{W,\epsilon}$ computes ϕ-quantiles for $\phi = \langle \epsilon_\ell, 2\epsilon_\ell, \ldots, 1\rangle$ using the instance of the Greenwald-Khanna algorithm, and stores the output sequence as the sketch for the block. It is a known result [15] that this sequence can be used to compute ϵ_ℓ-approximate quantiles over the elements of this block.

From the construction above, a sketch for a level-ℓ ACTIVE block is just a sequence of $\frac{1}{\epsilon_\ell}$ elements, which requires $O(\frac{1}{\epsilon_\ell})$ space. An instance of Greenwald-Khanna algorithm over elements of a level-ℓ block (with error parameter $\frac{\epsilon_\ell}{2}$) requires $O(\frac{1}{\epsilon_\ell}\log\epsilon_\ell N_\ell)$ space, which is the space required for a sketch corresponding to a level-ℓ block UNDER-CONSTRUCTION.

LEMMA 2. *(Merge Operation for Approximate Quantiles) Using a collection of s sketches over disjoint bags of $N_1, N_2, \ldots N_s$ elements each, with error parameters $\epsilon_1, \epsilon_2, \ldots \epsilon_s$, respectively, we can compute ϵ-approximate quantiles for the union of the bags with error parameter $\epsilon = \frac{\epsilon_1 N_1 + \epsilon_2 N_2 + \cdots + \epsilon_s N_s}{N_1 + N_2 + \cdots + N_s}$.*

PROOF. For simplicity, we assume that $1/\epsilon_i$ $(1 \leq i \leq s)$ is an integer. Also, assume that the sketches are constructed as described earlier, i.e., the ϵ_i-sketch for the i^{th} bag contains $\frac{\epsilon_i}{2}$-approximate ϕ-quantiles for $\phi = \langle \epsilon_i, 2\epsilon_i, \ldots, 1\rangle$, though the lemma is valid for the more general class of sketches introduced in [15]. An approximate ϕ-quantile, $\phi \in (0, 1]$, over the union of the bags can be computed as follows: Associate a weight of $\epsilon_i N_i$ with each element of the i-th sketch. Sort the elements of all sketches and pick the element with the property that the sum of weights of all preceding elements is $< \lceil \phi(N_1 + \ldots + N_s)\rceil$, but the sum including the element's weight is $\geq \lceil \phi(N_1 + \ldots + N_s)\rceil$. We claim without proof that this element is an ϵ-approximate ϕ-quantile for the union of the bags, where $\epsilon = \frac{\epsilon_1 N_1 + \epsilon_2 N_2 + \cdots + \epsilon_k N_k}{N_1 + N_2 + \cdots + N_k}$. □

THEOREM 2. *$\mathcal{Q}_{W,\epsilon}(m, N)$ allows $\frac{\epsilon W}{N}$-approximate quantiles to be computed over the elements $\{e_{m-N+1}, \ldots, e_m\}$. Further, $\mathcal{Q}_{W,\epsilon}(m, N)$ uses $O(\frac{1}{\epsilon}\log\frac{1}{\epsilon}\log W)$ space.*

PROOF. *Approximation*: The proof is identical to that of Theorem 1, with Lemma 1 replaced by Lemma 2.

Space requirement: The space required by $\mathcal{Q}_{W,\epsilon}(m, N)$ is the space required to store the block-level sketches for blocks that are currently ACTIVE or UNDER-CONSTRUCTION. As we argued earlier, a sketch for a level-ℓ ACTIVE block requires $O(\frac{1}{\epsilon_\ell})$ space, while a level-ℓ block UNDER-CONSTRUCTION requires $O(\frac{1}{\epsilon_\ell}\log\epsilon_\ell N_\ell)$ space.

The analysis of the space required for the block-level sketches corresponding to all the ACTIVE blocks is identical to the analysis that we presented for $\mathcal{C}_{W,\epsilon}$: There are at most $2^{L-\ell}$ ACTIVE blocks at level-ℓ. Hence, the total space required to store the sketches corresponding to all the ACTIVE blocks is $O(\sum_{\ell=0}^{L} 2^{L-\ell}/\epsilon_\ell)$. Substituting $\epsilon_\ell = \frac{\epsilon}{2(2L+2)} 2^{(L-\ell)}$, the required space is $O(\sum_{\ell=0}^{L} 2(2L+2)/\epsilon)$, which is $O(\sum_{\ell=0}^{L} \frac{L}{\epsilon}) = O(\frac{L^2}{\epsilon}) = O(\frac{1}{\epsilon}\log^2\frac{1}{\epsilon})$.

There is at most one block that is UNDER-CONSTRUCTION for each level. Therefore the total space required for sketches corresponding to the blocks UNDER-CONSTRUCTION is $O(\sum_{\ell=0}^{L} \frac{1}{\epsilon_\ell}\log\epsilon_\ell N_\ell)$. For any level-$\ell$, $\epsilon_\ell N_\ell = \frac{\epsilon W}{2(2L+2)}$, which is independent of ℓ. So, the above sum reduces to $O(\log\frac{\epsilon W}{2(2L+2)} \sum_{\ell=0}^{L} \frac{1}{\epsilon_\ell})$. Since, $\epsilon_\ell \propto 2^{-\ell}$, $\sum_{\ell=0}^{L} O(\frac{1}{\epsilon_\ell})$ is geometric and is $O(\frac{1}{\epsilon_L}) = O(\frac{1}{\epsilon}\log\frac{1}{\epsilon})$. The total space required for sketches for blocks UNDER-CONSTRUCTION is therefore

$O(\frac{1}{\epsilon}\log\frac{1}{\epsilon}\log\frac{\epsilon N}{\log(1/\epsilon)})$. The overall space is $O(\frac{1}{\epsilon}\log^2\frac{1}{\epsilon} + \frac{1}{\epsilon}\log\frac{1}{\epsilon}\log\frac{\epsilon W}{\log(2/\epsilon)})$ which can be simplified to the more conservative expression $O(\frac{1}{\epsilon}\log\frac{1}{\epsilon}\log W)$. □

6. RANDOMIZED, BOUNDED-WINDOW SKETCHES

In this section, we present two randomized, bounded-window sketches—$\mathcal{RC}_{W,\epsilon,\delta}$ for counts and $\mathcal{RQ}_{W,\epsilon,\delta}$ for quantiles. $\mathcal{RQ}_{W,\epsilon,\delta}$ is a minor variation of $\mathcal{Q}_{W,\epsilon}$, while $\mathcal{RC}_{W,\epsilon,\delta}$ is significantly different from $\mathcal{C}_{W,\epsilon}$. For the remainder of this section, assume a stream S with elements $e_1, e_2, \ldots$.

6.1 Counts

$\mathcal{RC}_{W,\epsilon,\delta}$ is based on a sampling technique called STICKY SAMPLING proposed in [18]. Let r denote the integer satisfying $1/2^r \leq (\frac{1}{\epsilon}\log(\epsilon\delta)^{-1})/W < 1/2^{r-1}$. $\mathcal{RC}_{W,\epsilon,\delta}$ logically partitions the stream into *blocks* of size 2^r. For each block, $\mathcal{RC}_{W,\epsilon,\delta}$ samples one element that belongs to the block uniformly at random.

When the current length of the stream is m and the current size of the window is N, the contents of $\mathcal{RC}_{W,\epsilon,\delta}(m,N)$ are as follows: For each element e_i $(m-N < i \leq m)$, belonging to the last N positions, that was chosen as a sample for its block, store the triple $\langle e_i, \tilde{f}_i, i\rangle$. The count $\tilde{f}_i$ denotes the number of occurrences of e_i until it is sampled again, or if e_i is not sampled subsequently, the number of occurrences of e_i until the current end of the stream. Formally, $\tilde{f}_i$ is defined as the number of elements $e_j = e_i$ $(i \leq j \leq m)$ such that there does not exist an element $e_k = e_i$ $(i < k \leq j)$ which was chosen as a sample for its block.

THEOREM 3. *$\mathcal{RC}_{W,\epsilon,\delta}(m,N)$ allows $\frac{\epsilon W}{N}$-approximate counts to be computed over the elements $\{e_{m-N+1}, \ldots, e_m\}$ with probability at least $(1-\delta)$. Further, $\mathcal{RC}_{W,\epsilon,\delta}(m,N)$ uses $O(\frac{1}{\epsilon}\log(\epsilon\delta)^{-1})$ space.*

PROOF. *Approximation:* An approximate count $\mathcal{A}$ over the bag of elements $\{e_{m-N+1}, \ldots, e_m\}$ is computed using $\mathcal{RC}_{W,\epsilon,\delta}(m,N)$ in a natural way: For each distinct element e that occurs as part of a triple in $\mathcal{RC}_{W,\epsilon,\delta}(m,N)$, $\mathcal{A}$ contains a pair $\langle e, \tilde{f}_e\rangle$. The approximate count $\tilde{f}_e$ for element e is the sum of all $\tilde{f}_i$ such that $\langle e, \tilde{f}_i, i\rangle$ is a triple in $\mathcal{RC}_{W,\epsilon,\delta}(m,N)$.

Consider an element e whose true frequency f_e in the current window exceeds ϵW. Assume that e does not occur as part of $\mathcal{A}$ or, if it does occur, $\tilde{f}_e < f_e - \epsilon W$, where $\tilde{f}_e$ is the approximate count of e stored in $\mathcal{A}$. We can show that this happens only if the earliest ϵW occurrences of e in the current window are not sampled. Let the earliest ϵW occurrences of e in the current window be spread over k different blocks. Let $c_1, \ldots, c_k$ denote the cumulative frequencies of these ϵW occurrences in the k blocks. Then the probability that the first ϵW occurrences of e are not sampled is equal to $\Pi_{i=1}^{k}(1-c_i/2^r)$. Since $1-c_i/2^r < (1-1/2^r)^{c_i}$, this probability is smaller than $\Pi_{i=1}^{k}(1-1/2^r)^{c_i} = (1-1/2^r)^{\epsilon W} < e^{-\epsilon W/2^r} < (\epsilon\delta)$.

Since $N \leq W$ (from the definition of bounded-window sketches), there are at most $\frac{N}{\epsilon W} \leq \frac{1}{\epsilon}$ elements whose frequency in the current window exceeds ϵW. The probability that some element e among these (at most) $\frac{1}{\epsilon}$ elements has $\tilde{f}_e < f_e - \epsilon W$ is less than $\epsilon\delta\frac{1}{\epsilon} = \delta$.

Therefore with a probability at least $(1-\delta)$, all the elements of e whose frequency f_e in the current window exceeds ϵW appear as a pair $\langle e, \tilde{f}_e\rangle$ in $\mathcal{A}$, such that $\tilde{f}_e \geq f_e - \epsilon W$. Further, for any pair $\langle e', \tilde{f}_{e'}\rangle$ that appears in $\mathcal{A}$, we can easily show that $\tilde{f}_{e'} \leq f_{e'}$, where $f_{e'}$ is the true frequency of the element e' in the current window. It follows that $\mathcal{A}$ is an $\frac{\epsilon W}{N}$-approximate count over the current window.

Space Requirement: The number of triples stored in $\mathcal{RC}_{W,\epsilon,\delta}(m,N)$ is less than $\frac{N}{2^r}+1$, which requires $O(\frac{W}{2^r}) = O(\frac{1}{\epsilon}\log(\epsilon\delta)^{-1})$ space. □

The sampling technique used in $\mathcal{RC}_{W,\epsilon,\delta}$ is slightly different from the original STICKY SAMPLING. In the original STICKY SAMPLING each new element is sampled independently with probability $1/2^r$. The space requirement for this sampling scheme is $O(\frac{1}{\epsilon}\log(\epsilon\delta)^{-1})$ in expectation. With 1-in-2^r sampling, as described above, the space requirements become worst-case.

6.2 Quantiles

$\mathcal{RQ}_{W,\epsilon,\delta}$ is identical to $\mathcal{Q}_{W,\epsilon}$ described in Section 5, except that it uses a randomized algorithm that we call RQALG instead of the deterministic Greenwald-Khanna algorithm. We first describe RQALG, before presenting details of $\mathcal{RQ}_{W,\epsilon,\delta}$.

A randomized 1-pass algorithm for computing ϵ-approximate quantiles over a stream of length M works as follows: We maintain a sample of size $O(\frac{1}{\epsilon^2}\log(\epsilon\delta)^{-1})$ using Vitter's reservoir sampling [27]. It is well-known that exact quantiles of the sample are ϵ-approximate quantiles of the stream, with probability at least $1-\delta$. An algorithm that is easier to code is to select 1 out of 2^k successive elements with $k = \lfloor\log_2(M/(\frac{1}{\epsilon^2}\log\delta^{-1}))\rfloor$. Quantiles computed over the resulting samples are indeed ϵ-approximate, as shown in [20]. The space requirements for both Vitter's scheme and the 1-in-2^k scheme, are $O(\frac{1}{\epsilon^2}\log(\epsilon\delta)^{-1})$. For small values of ϵ, the inverse-square dependence on ϵ leads to blowup in space requirements, making the algorithms impractical. Two approaches have been devised in recent years to combat this problem: (a) Gibbons and Matias [13] show how samples can be *compressed* if the stream abounds with duplicates, and (b) Manku *et al* [19] suggest a two-stage pipeline: the output of 1-out-of-2^k sampling can be fed to a deterministic quantile-finding algorithm, with the error parameter being $\epsilon/2$ for the two stages. By employing the Greenwald-Khanna algorithm [15] in the second stage, we arrive at a randomized algorithm requiring only $O(\frac{1}{\epsilon}\log(\frac{1}{\epsilon}\log(\epsilon\delta)^{-1}))$ space[4]. We call this algorithm RQALG(ϵ,δ), parameterized by ϵ and δ.

$\mathcal{RQ}_{W,\epsilon,\delta}(m,N)$ stores block-level sketches for blocks that are currently ACTIVE or UNDER-CONSTRUCTION, exactly like $\mathcal{Q}_{W,\epsilon}(m,N)$. When a level-$\ell$ block is UNDER-CONSTRUCTION,

[4]If M is not known in advance, k is not fixed but grows logarithmically with M. However, we claim that it is possible to employ the "adaptive sampling scheme" Manku *et al* [20] in conjunction with a modified Greenwald-Khanna algorithm to arrive at the same bound.

$\mathcal{RQ}_{W,\epsilon,\delta}$ runs an instance of RQALG($\frac{\epsilon_\ell}{2}, \frac{\delta}{(2L+2)}$) over the elements of the block. When the block becomes ACTIVE, $\mathcal{RQ}_{W,\epsilon,\delta}$ computes ϕ-quantiles for $\phi = \langle \epsilon_\ell, 2\epsilon_\ell, \ldots, 1\rangle$ using the above instance of RQALG.

THEOREM 4. *$\mathcal{RQ}_{W,\epsilon,\delta}(m,N)$ allows $\frac{\epsilon W}{N}$-approximate quantiles to be computed over the elements $\{e_{m-N+1}, \ldots, e_m\}$ with probability at least $(1-\delta)$. Further, $\mathcal{RQ}_{W,\epsilon,\delta}(m,N)$ uses $O(\frac{1}{\epsilon}\log\frac{1}{\epsilon}\log(\frac{1}{\epsilon}\log\Delta^{-1}))$ space, where $\Delta = (\epsilon\delta)/\log(1/\epsilon)$.*

PROOF. (Sketch) Algorithm RQALG guarantees that each block-level sketch has the desired error parameter with probability at least $(1-\frac{\delta}{(2L+2)})$. In order to compute an approximate quantile, at most $(2L+2)$ block-level sketches are merged. With probability at least $(1-\delta)$, each of these have the desired error parameter.

Space requirements for ACTIVE blocks are $O(\frac{1}{\epsilon}\log^2\frac{1}{\epsilon})$ (see Theorem 2 for a proof). Space-requirements for blocks UNDER-CONSTRUCTION are only $O(\frac{1}{\epsilon}(\log\frac{1}{\epsilon})\log(\frac{1}{\epsilon}\log\Delta^{-1}))$, where $\Delta = O(\epsilon\delta/\log(1/\epsilon))$. Summing the two, we get the overall space-complexity. □

7. UNBOUNDED-WINDOW SKETCHES

We present a general technique for constructing unbounded-window sketches from bounded-window sketches that satisfy a certain property. All of our bounded-window sketches presented in Sections 5 and 6 satisfy this property, and therefore we can obtain a unbounded-window sketch corresponding to each one of our bounded-window sketches.

7.1 General Technique

Let $\mathcal{F}$ denote a bounded-window sketch that satisfies the following property:

> PROPERTY P: For any positive integers k and m, and real $\epsilon \in (0,1]$, $\mathcal{F}_{2^{k+1},\epsilon}(m,2^k)$ can be constructed using $\mathcal{F}_{2^k,\epsilon}(m,2^k)$.

For any error parameter ϵ, an unbounded-window sketch $\mathcal{V}_\epsilon$ can be constructed from $\mathcal{F}$ as follows: $\mathcal{V}_\epsilon(m,N)$ is the collection of the following $\lfloor\log_2\epsilon N\rfloor$ bounded-window sketches, where k is the integer satisfying $2^{k-1} \le N < 2^k$:

$$\{\mathcal{F}_{2^k,\frac{\epsilon}{2}}(m,N), \mathcal{F}_{2^{k-1},\frac{\epsilon}{2}}(m,2^{k-1}), \ldots, \mathcal{F}_{\frac{2}{\epsilon},\frac{\epsilon}{2}}(m,\frac{2}{\epsilon})\}$$

The three basic operations QUERY, INSERT, and DELETE of $\mathcal{V}_\epsilon(m,N)$ are performed as follows:

QUERY: To compute ϵ-approximate statistics for the current window, we use the QUERY operation of the sketch $\mathcal{F}_{2^k,\frac{\epsilon}{2}}(m,N)$. By definition, the QUERY operation of $\mathcal{F}_{2^k,\frac{\epsilon}{2}}(m,N)$ produces $\frac{(\epsilon/2)2^k}{N}$-approximate statistics. Since $N > 2^k/2$, the approximation $\frac{(\epsilon/2)2^k}{N} < \epsilon$, as required.

INSERT: We compute $\mathcal{V}_\epsilon(m+1,N+1)$ from $\mathcal{V}_\epsilon(m,N)$ when an element e_{m+1} is inserted as follows: We insert the element e_{m+1} into the sketch $\mathcal{F}_{2^k,\frac{\epsilon}{2}}(m,N)$ using its INSERT operation to produce $\mathcal{F}_{2^k,\frac{\epsilon}{2}}(m+1,N+1)$. For all the remaining sketches $\{\mathcal{F}_{2^{k-1},\frac{\epsilon}{2}}(m,2^{k-1}), \ldots, \mathcal{F}_{\frac{2}{\epsilon},\frac{\epsilon}{2}}(m,\frac{2}{\epsilon})\}$, we first perform a DELETE operation and then insert the element e_{m+1} using their INSERT operations. This sequence of operations results in the following collection of sketches:

$$\{\mathcal{F}_{2^k,\frac{\epsilon}{2}}(m+1,N+1), \mathcal{F}_{2^{k-1},\frac{\epsilon}{2}}(m+1,2^{k-1}), \ldots, \mathcal{F}_{\frac{2}{\epsilon},\frac{\epsilon}{2}}(m+1,\frac{2}{\epsilon})\}$$

Further, if $N+1 = 2^k$, we construct $\mathcal{F}_{2^{k+1},\frac{\epsilon}{2}}(m+1,2^k)$ from $\mathcal{F}_{2^k,\frac{\epsilon}{2}}(m+1,2^k)$ and add it to the collection of sketches. This step is possible since $\mathcal{F}$ satisfies PROPERTY P.

DELETE: We compute $\mathcal{V}_\epsilon(m,N-1)$ as follows: We perform a DELETE operation for the sketch $\mathcal{F}_{2^k,\frac{\epsilon}{2}}(m,N)$. All other sketches remain unchanged. This results in the collection of sketches:

$$\{\mathcal{F}_{2^k,\frac{\epsilon}{2}}(m,N-1), \mathcal{F}_{2^{k-1},\frac{\epsilon}{2}}(m,2^{k-1}), \ldots, \mathcal{F}_{\frac{2}{\epsilon},\frac{\epsilon}{2}}(m,\frac{2}{\epsilon})\}$$

Further, if $N-1 < 2^{k-1}$, we drop the sketch $\mathcal{F}_{2^k,\frac{\epsilon}{2}}(m,N-1)$ from the collection of sketches.

7.2 Specific Constructions

In this section we show that all the bounded-window sketches that we presented in Sections 5 and 6 satisfy PROPERTY P.

LEMMA 3. *The bounded-window sketches $\mathcal{C}$, $\mathcal{Q}$ and $\mathcal{RQ}$ satisfy* PROPERTY P.

PROOF. We present the proof only for the bounded-window sketch $\mathcal{C}$. The proofs for $\mathcal{Q}$ and $\mathcal{RQ}$ are very similar.

In order to prove that $\mathcal{C}$ satisfies PROPERTY P, we need to show that $\mathcal{C}_{2^{k+1},\epsilon}(m,2^k)$ can be constructed from $\mathcal{C}_{2^k,\epsilon}(m,2^k)$. Intuitively, this construction is feasible since $\mathcal{C}_{2^k,\epsilon}(m,2^k)$ is a more accurate sketch than $\mathcal{C}_{2^{k+1},\epsilon}(m,2^k)$: Both sketches compute statistics over the same window, but $\mathcal{C}_{2^{k+1},\epsilon}(m,2^k)$ allows ϵ-approximate statistics to be computed over the current window, while $\mathcal{C}_{2^{k+1},\epsilon}(m,2^k)$ only allows 2ϵ-approximate statistics to be computed.

For presentation clarity, we assume that no block of $\mathcal{C}_{2^k,\epsilon}(m,2^k)$ is UNDER-CONSTRUCTION. This is equivalent to assuming that m is an exact multiple of 2^k. The proof for the more general case uses essentially the same ideas as the proof presented here.

Let $L = \log_2(\frac{4}{\epsilon})$ denote the number of different levels for blocks: this is the same for both $\mathcal{C}_{2^{k+1},\epsilon}(m,2^k)$ and $\mathcal{C}_{2^k,\epsilon}(m,2^k)$, since L is a function of ϵ alone. For $\ell < L$, we can show that each level-ℓ ACTIVE block of $\mathcal{C}_{2^{k+1},\epsilon}(m,2^k)$ is the same as some level-$(\ell+1)$ ACTIVE block of $\mathcal{C}_{2^k,\epsilon}(m,2^k)$. For this common block, $\mathcal{C}_{2^k,\epsilon}(m,2^k)$ stores a sketch with error parameter $\epsilon_{\ell+1}$, while $\mathcal{C}_{2^{k+1},\epsilon}(m,2^k)$ stores a sketch with error parameter ϵ_ℓ. By definition, $\epsilon_{\ell+1} = \frac{\epsilon_\ell}{2}$. In general, for counts, we can show that a sketch for a bag of elements with error parameter ϵ' can be constructed using a sketch for the same bag with error parameter $\frac{\epsilon'}{2}$. (This is shown formally in Lemma 4.) Therefore, for $\ell < L$, we can construct a

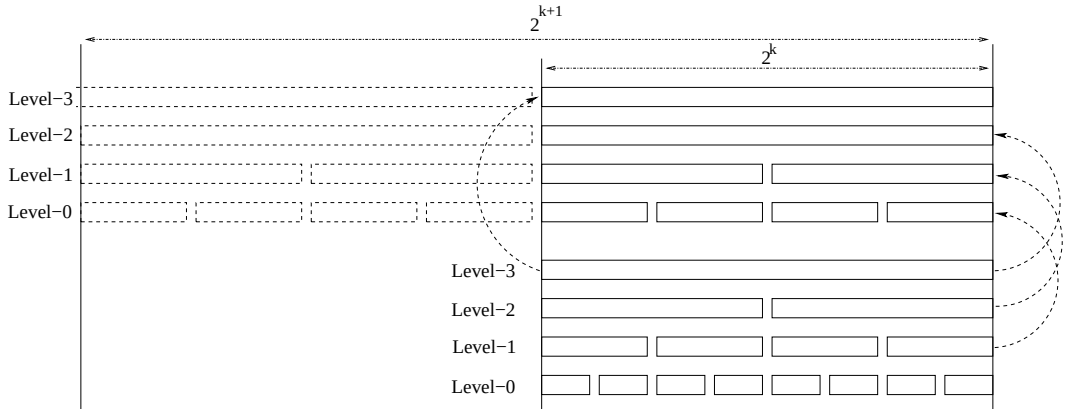

Figure 2: Construction of $\mathcal{C}_{2^{k+1},\epsilon}(m, 2^k)$ from $\mathcal{C}_{2^k,\epsilon}(m, 2^k)$

sketch for a level-ℓ ACTIVE block of $\mathcal{C}_{2^{k+1},\epsilon}(m, 2^k)$ using the sketch maintained for the same block by $\mathcal{C}_{2^k,\epsilon}(m, 2^k)$. (See Figure 2 for an illustration.)

We now consider level-L blocks of $\mathcal{C}_{2^{k+1},\epsilon}(m, 2^k)$. Since the current window size for $\mathcal{C}_{2^{k+1},\epsilon}(m, 2^k)$ is 2^k, and the size of a level-L block in $\mathcal{C}_{2^{k+1},\epsilon}(m, 2^k)$ is 2^{k+1} (by definition), there can be no ACTIVE level-L block in $\mathcal{C}_{2^{k+1},\epsilon}(m, 2^k)$. We consider two possible cases depending on whether $\mathcal{C}_{2^{k+1},\epsilon}(m, 2^k)$ contains a level-L block UNDER-CONSTRUCTION or not. If $\mathcal{C}_{2^{k+1},\epsilon}(m, 2^k)$ does not contain a level-L block UNDER-CONSTRUCTION, we do not need to do anything further, since we have constructed a valid block-level sketch for every block of $\mathcal{C}_{2^{k+1},\epsilon}(m, 2^k)$ that is either UNDER-CONSTRUCTION or ACTIVE.

In order to be able to handle the second case, where $\mathcal{C}_{2^{k+1},\epsilon}(m, 2^k)$ contains a level-L block UNDER-CONSTRUCTION, we make one minor modification to the running of the one-pass Misra-Gries algorithm over blocks UNDER-CONSTRUCTION: When a block becomes active, we do not immediately terminate the algorithm and discard the sketches as described in Section 5.1, but do so only when the next block in the same level reaches UNDER-CONSTRUCTION state. (This change translates to slightly modifying the definitions of ACTIVE and UNDER-CONSTRUCTION blocks, so that, at any point in time, there is always a block in each level in the UNDER-CONSTRUCTION state.) None of the results so far are affected by this modification. With this modification, an instance of the one-pass algorithm, capable of producing an ϵ_L-sketch is running over the elements of the single active level-L block of $\mathcal{C}_{2^k,\epsilon}(m, 2^k)$. We simply continue this algorithm on behalf of the level-L block of $\mathcal{C}_{2^{k+1},\epsilon}(m, 2^k)$ that is UNDER-CONSTRUCTION, and after the next 2^k elements arrive, this algorithm can be used to produce an ϵ_L-sketch. □

LEMMA 4. *For quantiles (resp. counts), a sketch with error parameter ϵ for a bag that uses $O(\frac{1}{\epsilon})$ space can be obtained from a sketch for the same bag that has error parameter $\frac{\epsilon}{2}$.*

PROOF. First consider quantiles. An ϵ-sketch can be constructed by probing $\frac{\epsilon}{2}$ sketch for quantiles $\phi = \epsilon, 2\epsilon, \ldots, 1$ and storing the returned approximate quantiles. Clearly, this sketch uses $O(\frac{1}{\epsilon})$ space.

For counts, retrieve all pairs $\langle e, \tilde{f}_e \rangle$ in the $\frac{\epsilon}{2}$-sketch, and store as part of the ϵ-sketch those pairs with $\tilde{f}_e \geq \epsilon M/2$, where M is the size of the bag. At most $\frac{2}{\epsilon}$ elements can have $\tilde{f}_e \geq \epsilon M/2$. (Recall that the approximate count $\tilde{f}_e \leq$ the frequency). Therefore, size of the constructed sketch is $O(\frac{1}{\epsilon})$. □

LEMMA 5. *The randomized bounded-window sketch $\mathcal{RC}$ satisfies* PROPERTY P.

PROOF. (Sketch) We need to show that $\mathcal{RC}_{2^{k+1},\epsilon,\delta}(m, 2^k)$ can be constructed using $\mathcal{RC}_{2^k,\epsilon,\delta}(m, 2^k)$. We can easily show that if $\mathcal{RC}_{2^k,\epsilon,\delta}$ samples one element every 2^r elements, $\mathcal{RC}_{2^{k+1},\epsilon,\delta}$ samples one element every 2^{r+1} elements. The construction of $\mathcal{RC}_{2^{k+1},\epsilon,\delta}(m, 2^k)$ from $\mathcal{RC}_{2^k,\epsilon,\delta}(m, 2^k)$ essentially uses the fact that a 1-in-2^{r+1} sample can be obtained from a 1-in-2^r sample. □

The theorems below follow directly from the general technique for constructing unbounded-window sketches and the space requirements for our bounded-window sketches.

THEOREM 5. *There exists an unbounded-window sketch $\mathcal{V}_\epsilon$ for computing ϵ-approximate counts, such that $\mathcal{V}_\epsilon(m, N)$ requires $O(\frac{1}{\epsilon}\log^2\frac{1}{\epsilon}\log\epsilon N)$ space.*

THEOREM 6. *There exists an unbounded-window sketch $\mathcal{V}_\epsilon$ for computing ϵ-approximate quantiles, such that $\mathcal{V}_\epsilon(m, N)$ requires $O(\frac{1}{\epsilon}\log\frac{1}{\epsilon}\log N\log\epsilon N)$ space.*

THEOREM 7. *There exists an randomized, unbounded-window sketch $\mathcal{V}_{\epsilon,\delta}$ for computing ϵ-approximate counts with probability at least $(1-\delta)$, such that $\mathcal{V}_{\epsilon,\delta}(m, N)$ requires $O(\frac{1}{\epsilon}\log(\epsilon\delta)^{-1}\log\epsilon N)$ space.*

THEOREM 8. *There exists an randomized, unbounded-window sketch $\mathcal{V}_{\epsilon,\delta}$ for computing ϵ-approximate quantiles with probability at least $(1-\delta)$, such that $\mathcal{V}_{\epsilon,\delta}(m,N)$ requires $O(\frac{1}{\epsilon}\log\frac{1}{\epsilon}\log(\frac{1}{\epsilon}\log\Delta^{-1})\log\epsilon N)$ space, where $\Delta = (\epsilon\delta)/\log(1/\epsilon)$.*

8. EXTENSIONS TO OTHER TYPES OF WINDOWS

As we indicated in Section 1, fixed- and variable-size windows capture the essential features of many common types of windows. Tuple-based windows [22] correspond exactly to fixed-size windows.

A *time-based* window [22] is defined for streams whose elements have associated timestamps, indicating their arrival time on the stream. At any given point in time, a time-based window contains the elements of a stream with timestamps in the last T time units, for some fixed parameter T. Since zero, one, or more than one elements can have the same timestamp, the number of elements in a time-based windows can vary. Therefore time-based windows are closely related to variable-size windows[5]. All of our unbounded-window can be extended to handle time-based windows. Briefly, this is done by storing for each block the timestamp of the oldest element that belongs to the block. (Recall that all of our unbounded-window sketches are constructed using bounded-window sketches, which in turn maintain some summary information at the level of blocks, for some definition of blocks.) We omit the details, which are straightforward.

Lin *et al* [17] introduce the notion of n-of-N window model. An algorithm maintaining statistics in the n-of-N window model should be able to compute approximate statistics over the last n elements of the stream for any $n \leq N$. Any algorithm over variable-size windows necessarily has the ability to compute statistics in the n-of-N window, since an adversary for variable-size windows can arbitrarily shrink a window.

9. ACKNOWLEDGMENTS

This work was supported by an SNRC grant and by National Science Foundation grants IIS-0118173, IIS-9817799, and EIA-0137761. The authors thank Mayank Bawa for useful feedback on an initial draft.

10. REFERENCES

[1] R. Agrawal, T. Imielinski, and A. N. Swami. Mining association rules between sets of items in large databases. In *Proc. of the 1993 ACM SIGMOD Intl. Conf. on Management of Data*, pages 207–216, May 1993.

[2] B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom. Models and issues in data stream systems. In *Proc. of the 21st ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Systems*, pages 1–16, June 2002.

[3] B. Babcock, M. Datar, and R. Motwani. Sampling from a moving window over streaming data. In *Proc. of the 13th Annual ACM-SIAM Symp. on Discrete Algorithms*, pages 633–634, Jan. 2002.

[4] B. Babcock, M. Datar, R. Motwani, and L. O'Callaghan. Maintaining variance and k-medians over data stream windows. In *Proc. of the 22nd ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Systems*, pages 234–243, June 2003.

[5] M. Blum, R. W. Floyd, V. R. Pratt, R. L. Rivest, and R. E. Tarjan. Time bounds for selection. *Journal of Computer and System Sciences*, 7(4):448–461, Aug. 1973.

[6] M. Charikar, K. Chen, and M. Farach-Colton. Finding frequent items in data streams. In *Proc. of the 29th Intl. Coll. on Automata, Languages and Programming*, pages 693–703, July 2002.

[7] G. Cormode and S. Muthukrishnan. What's hot and what's not: tracking most frequent items dynamically. In *Proc. of the 22nd ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Systems*, pages 296–306, June 2003.

[8] G. Cormode and S. Muthukrishnan. An improved data stream summary: The count-min sketch and its applications. In *LATIN 2004*, pages 29–38, Apr. 2004.

[9] M. Datar, A. Gionis, P. Indyk, and R. Motwani. Maintaining stream statistics over sliding windows. In *Proc. of the 13th Annual ACM-SIAM Symp. on Discrete Algorithms*, pages 635–644, Jan. 2002.

[10] E. D. Demaine, A. López-Ortiz, and J. I. Munro. Frequency estimation of internet packet streams with limited space. In *Proc. of the 10th Annual European Symp. on Algorithms*, pages 348–360, Sept. 2002.

[11] D. J. DeWitt, J. F. Naughton, and D. A. Schneider. Parallel sorting on a shared-nothing architecture using probabilistic splitting. In *Proc. of the 1st Intl. Conf. on Parallel and Distributed Information Systems*, pages 280–291, Dec. 1991.

[12] R. W. Floyd and R. L. Rivest. Expected time bounds for selection. *Comm. of the ACM*, 18(3):165–172, Mar. 1975.

[13] P. B. Gibbons and Y. Matias. New sampling-based summary statistics for improving approximate query answers. In *Proc. of the 1998 ACM SIGMOD Intl. Conf. on Management of Data*, pages 331–342, June 1998.

[14] P. B. Gibbons and S. Tirthapura. Distributed streams algorithms for sliding windows. In *Proc. of the 14th Annual ACM Symp. on Parallel Algs. and Architectures*, pages 63–72, Aug. 2002.

[15] M. Greenwald and S. Khanna. Space-efficient online computation of quantile summaries. In *Proc. of the 2001 ACM SIGMOD Intl. Conf. on Management of Data*, pages 58–66, May 2001.

[16] R. M. Karp, S. Shenker, and C. H. Papadimitriou. A simple algorithm for finding frequent elements in streams and bags. *ACM Trans. on Database Systems*, 28(1):51–55, Mar. 2003.

[5] There exist statistics, e.g., "the count of all elements in the window", that can be computed using constant space over variable-size windows, but not over time-based windows.

[17] X. Lin, H. Lu, J. Xu, and J. X. Yu. Continuously maintaining quantile summaries of the most recent N elements over a data stream. In *Proc. of the 20th Intl. Conf. on Data Engineering*, Mar. 2004. (to appear).

[18] G. S. Manku and R. Motwani. Approximate frequency counts over data streams. In *Proc. of the 28th Intl. Conf. on Very Large Data Bases*, pages 356–357, Aug. 2002.

[19] G. S. Manku, S. Rajagopalan, and B. G. Lindsay. Approximate medians and other quantiles in one pass and with limited memory. In *Proc. of the 1998 ACM SIGMOD Intl. Conf. on Management of Data*, pages 426–435, June 1998.

[20] G. S. Manku, S. Rajagopalan, and B. G. Lindsay. Random sampling techniques for space efficient online computation of order statistics of large datasets. In *Proc. of the 1999 ACM SIGMOD Intl. Conf. on Management of Data*, pages 251–262, June 1999.

[21] J. Misra and D. Gries. Finding repeated elements. *Sci. Comput. Programming*, 2(2):143–152, Nov. 1982.

[22] R. Motwani, J. Widom, et al. Query processing, approximation, and resource management in a data stream management system. In *Proc. of the 1st Conf. on Innovative Data Systems Research*, pages 245–256, Jan. 2003.

[23] J. I. Munro and M. Paterson. Selection and sorting with limited storage. *Theoretical Computer Science*, pages 315–323, 1980.

[24] M. Paterson. Progress in selection. In *Proc. of the 5th Scandinavian Workshop on Algorithm Theory*, pages 368–379, July 1996.

[25] I. Pohl. A minimum storage algorithm for computing the median. Technical Report IBM Research Report RC 2701 (# 12713), IBM T. J. Watson Center, 1969.

[26] H. Toivonen. Sampling large databases for association rules. In *Proc. of the 22nd Intl. Conf. on Very Large Data Bases*, pages 134–145, Sept. 196.

[27] J. S. Vitter. Random sampling with a reservoir. *ACM Trans. on Mathematical Software*, 11(1):37–57, Mar. 1985.