Maintaining Significant Stream Statistics over Sliding Windows

L.K. Lee * H.F. Ting *

Abstract

In this paper, we introduce the Significant One Counting problem. Let ε and θ be respectively some user-specified error bound and threshold. The input of the problem is a stream of bits. We need to maintain some data structure that allows us to estimate the number of 1-bits in a sliding window of size n such that whenever there are at least θn 1-bits in the window, the relative error of the estimate is guaranteed to be at most ε. When $\theta = 1/n$, our problem becomes the Basic Counting problem proposed by Datar *et al.* [*ACM-SIAM Symposium on Discrete Algorithms* (2002), pp. 635–644]. We prove that any data structure for the Significant One Counting problem must use at least $\Omega(\frac{1}{\varepsilon}\log^2\frac{1}{\theta} + \log \varepsilon\theta n)$ bits of memory. We also design a data structure for the problem that matches this memory bound and supports constant query and update time. Note that for fixed θ and ε, our data structure uses $O(\log n)$ bits of memory, while any data structure for the Basic Counting problem needs $\Omega(\log^2 n)$ bits in the worst case.

1 Introduction

Data streams are common in many applications such as network monitoring, telecommunications and financial monitoring. For example, data packets in network monitoring and stock transactions in financial monitoring are received and processed in the form of a data stream. There are many data structures and algorithms proposed for estimating statistics of data streams [3, 10, 11, 12, 13, 15, 16, 17]. Because of practical considerations, any algorithm for data streams must satisfy the following requirements: (1) it can only process the stream in one pass; (2) it uses a very small amount of working memory; and (3) it has small update and query times.

This paper focuses on the sliding window model for data stream algorithms [6]. In this model, the data stream is infinite. The data items in the stream are received one by one, and the statistics are computed over a *sliding window* of size n (not over the whole stream), which covers the n most recent data items received. Many problems related to sliding window have been proposed and studied [1, 2, 6, 7, 14]. One such problem is the *Basic Counting* problem, which asks for designing some data structure that allows us to estimate, at any time, the number of 1-bits in the sliding window such that the relative error of the estimate is bounded by ε, i.e., an estimate $\hat{m}$ of the actual number m of 1-bits such that $m - \varepsilon m \le \hat{m} \le m + \varepsilon m$. Datar *et al.* [6] gave the first solution for the problem. Their data structure uses $O(\frac{1}{\varepsilon}\log^2(\varepsilon n))$ bits of memory and has $O(\log n)$ update time and $O(1)$ query time. They also proved that any data structure for the problem must use $\Omega(\frac{1}{\varepsilon}\log^2(\varepsilon n))$ bits of memory. Later, Gibbons *et al.* [14] gave an improved data structure for the problem; it uses the same $O(\frac{1}{\varepsilon}\log^2(\varepsilon n))$ bits of memory and has $O(1)$ query time, but the update time is reduced to $O(1)$. Note that the data structure of Gibbons *et al.* has optimal time and space complexity. Unfortunately, the $\Theta(\frac{1}{\varepsilon}\log^2(\varepsilon n))$ memory requirement is still too much to be practical in many applications.

In this paper, we break the $\Theta(\frac{1}{\varepsilon}\log^2(\varepsilon n))$ memory requirement barrier, not by giving a better data structure (which is impossible), but by defining a new problem. After analyzing the lower bound proof of Datar *et al.*, we observe that the $\Theta(\frac{1}{\varepsilon}\log^2(\varepsilon n))$ bound comes from the fact that any correct data structure should cover all cases; in particular, it needs to guarantee the relative error bound even when the window has only a few 1-bits. However, this is not a requirement for many applications. For these applications, if the actual number of 1-bits in the window is small, it is enough to know that it is small and an estimate with a bounded relative error is not required. For example, in telecommunications, it is usually the case that users with usage above some threshold are charged by usage while the rest are charged by some fixed rate. Thus, we need good usage estimate for a user only when the number of his/her calls is larger than the threshold. Another example is in financial monitoring: microfinance institutions want good estimates only for stocks with lots of recent transactions; they are not interested in those with few transactions. These applications are called *threshold accounting* in [9], and for them we introduce the *Significant One Counting* problem, which is defined as follows:

Given a stream of bits, maintain some data

*Department of Computer Science, The University of Hong Kong, Pokfulam, Hong Kong, {lklee, hfting}@cs.hku.hk

> structure that allows us to make, at any time, an estimate $\hat{m}$ of the number m of 1-bits in the sliding window (of size n) such that if $m \geq \theta n$, we have $|\hat{m} - m| \leq \varepsilon m$.

Here, $0 < \theta < 1$ and $0 < \varepsilon < 1$ are two user-specified values. We call θ the *threshold* and ε the *relative error bound*. We assume that $\varepsilon \geq \frac{1}{\theta n}$; otherwise the error bound will force us to find the exact number of 1-bits in the stream. Note that when $\theta = 1/n$, our problem becomes the Basic Counting problem. We prove in this paper a lower bound on the size of any data structure for the Significant One Counting problem. We show that any data structure for the problem must have size at least $\Omega(\frac{1}{\varepsilon}\log^2\frac{1}{\theta} + \log\varepsilon\theta n)$ bits. Furthermore, we give an optimal data structure for the problem. It has the following properties:

1. It has constant update and query time.
2. It uses $O(\frac{1}{\varepsilon}\log^2\frac{1}{\theta} + \log\varepsilon\theta n)$ bits of memory, which matches the lower bound.
3. It allows us to find an estimate $\hat{m}$ of the number m of 1-bits in the sliding window such that
   - if $m \geq \theta n$, we have $m \leq \hat{m} \leq m + \varepsilon m$, and
   - if $m < \theta n$, we have $m \leq \hat{m} \leq m + \varepsilon\theta n$.

Note that our data structure has better relative error guarantees than what is required by the problem. Furthermore, our data structure becomes an optimal data structure for the Basic Counting problem when we set θ to $1/n$. On the other hand, for any fixed θ, our data structure has size only $O(\frac{1}{\varepsilon} + \log(\varepsilon n))$ bits.

Our data structure is simple and easy to implement, and can be used directly to solve a practical problem on data stream, namely *finding frequent items in a sliding window*. The problem is an extension to the classical problem of finding frequent items in the entire data stream, which has been studied extensively [4, 5, 8, 18, 19, 20] and is defined as follows: Let Σ be a set of t items and $0 < \theta < 1$ be a constant. The input of the problem is a data stream of items in Σ. The problem asks for designing some data structure that allows us to determine, at any time, the set of *frequent items* in the stream, i.e., those items whose number of occurrences in the stream is no less than θN where N is the total number of items currently in the stream. It is easy to see that at any time, there can be at most $1/\theta$ frequent items. Karp *et al.* [19] proved that any data structure solving the problem needs at least $\Omega(t\log\frac{N}{t})$ bits of memory. Demaine *et al.* [8] and Karp *et al.* [19] described data structures that allow us to identify a set of at most $1/\theta$ items that contains all the frequent items. However, this set may also contain non-frequent items whose number of occurrences in the stream is much smaller than θN.

Instead of finding the frequent items in the entire stream, we are interested in finding the frequent items in the sliding window of size n, i.e., those items that occur no less than θn times in the window. We can use our data structure for the Significant One Counting problem to identify these frequent items as follows: Let $0 < \varepsilon < 1$ be some fixed constant.

> We maintain t instances of our data structure for Significant One Counting (with threshold θ and relative error bound ε), one for each item in Σ. When an item e is received, the data structure for e is updated as if a 1-bit was received, and all the other data structures are updated as if a 0-bit was received. When there is a query on the set of frequent items, we return the set S of items whose estimates are no less than θn.

Note that by definition, every frequent item appears no less than θn times in the sliding window, and by Property (3) of our data structure, its estimate is no less than θn. Hence, the set S includes all the frequent items. Furthermore, it is easy to verified that (i) $|S| \leq \frac{1}{(1-\epsilon)\theta}$ and (ii) for any non-frequent item in S, its number of occurrences in the sliding window is at least $(1-\varepsilon)\theta n$. Since θ and ε are fixed constants, each of the t data structures uses $O(\frac{1}{\varepsilon}\log^2\frac{1}{\theta} + \log(\varepsilon\theta n)) = O(\log n)$ bits and the whole implementation uses $O(t\log n)$ bits. Finally, we note that in addition to identifying the set of frequent items, our solution also gives estimates of the number of occurrences of the frequent items with a bounded relative error; this property is important for application such as usage-based pricing in telecommunications [9].

Our paper is organized as follows. In section 2, we prove a lower bound of $\Omega(\frac{1}{\varepsilon}\log^2\frac{1}{\theta} + \log\varepsilon\theta n)$ bits of memory for any (deterministic) data structure for the Significant One Counting problem. In section 3, we introduce a one-level data structure. In section 4, we improve the memory usage by using a multi-level data structure. We present our optimal data structure in section 5.

2 Lower Bound

In this section, we derive two lower bounds on the size of any data structure for the Significant One Counting problem; one is $\Omega(\log\varepsilon\theta n)$ and the other is $\Omega(\frac{1}{\varepsilon}\log^2\frac{1}{\theta})$. Combining these two results we conclude a lower bound of $\Omega(\frac{1}{\varepsilon}\log^2\frac{1}{\theta} + \log\varepsilon\theta n)$, which is claimed in the previous section. Note that our results have assumed some

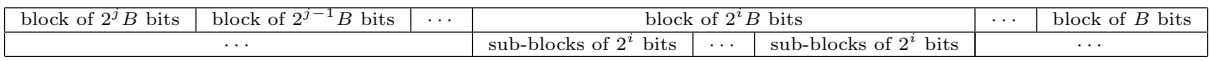

Figure 1: The bits in the sliding window are listed from left to right, where the rightmost bit is the most recent bit, and divided into different blocks. A block of size 2^iB is further divided into B contiguous sub-blocks of size 2^i bits.

bounds on the values of the threshold θ and the relative error bound ϵ.

LEMMA 2.1. *Suppose that $\theta \leq \frac{1}{2}$ and $\varepsilon \leq \frac{1}{5}$. Then, any data structure for the Significant One Counting problem with threshold θ and relative error bound ε has size $\Omega(\log \varepsilon\theta n)$ bits.*

Proof. Suppose that the data stream has a sequence of n 0-bits followed by a sequence P of $\varepsilon\theta n$ 1-bits. For the sake of contradiction, assume that the data structure uses $o(\log \varepsilon\theta n)$ bits of memory. Thus, there are two 1-bits in sequence P, say the x-th 1-bit and the $(x+y)$-th 1-bit where $x, y \in [1, \varepsilon\theta n - 1]$, such that when they are received, the memory states are the same, denoted by M.

When the x-th 1-bit is received, since $x < \varepsilon\theta n < n$, the actual number of 1-bits in the sliding window is $m_1 = x$. Consider that following this x-th 1-bit is $y\delta$ 1-bits, where $\delta = \lceil (4\varepsilon\theta n)/y \rceil$. Note that after the arrivals of each y 1-bits, the memory state will also be M. After the arrivals of the $y\delta$ 1-bits, the memory state is therefore M. Since $\theta \leq \frac{1}{2}$ and $\varepsilon \leq \frac{1}{5}$, it follows that $x + y\delta \leq x + (4\varepsilon\theta n + y) < 6\varepsilon\theta n < n$. Thus, the actual number of 1-bits in the sliding window is $m_2 = x + y\delta$.

Consider the two cases that a sequence S of $(\theta n - x)$ 1-bits are received after (i) the xth 1-bit is received, and (ii) the $y\delta$ 1-bits following the x-th 1-bit are received. For (i), the actual number of 1-bits in the sliding window is $m_1' = m_1 + (\theta n - x) = x + (\theta n - x) = \theta n$. For (ii), since $\theta \leq \frac{1}{2}$ and $\varepsilon \leq \frac{1}{5}$, we have $m_2 + (\theta n - x) = x + y\delta + (\theta n - x) = \theta n + y\delta < \theta n + 4\varepsilon\theta n + y < \theta n + 5\varepsilon\theta n \leq n$ and the actual number of 1-bits in the sliding window is thus $m_2' = m_2 + (\theta n - x) = \theta n + y\delta \geq \theta n + 4\varepsilon\theta n$. Note that when using the data structure to estimate the number of 1-bits of the data streams described in (i) and (ii), both streams will cause the data structure change to memory state M immediately before receiving the sequence S. Then, after receiving the whole sequence S, we should return the same estimate. Thus the data structure will give an estimate with absolute error at least $|m_1' - m_2'|/2 \geq (4\varepsilon\theta n)/2 = 2\varepsilon\theta n$ for one of the two streams. Consequently, the relative error is at least

$$\begin{aligned} \frac{2\varepsilon\theta n}{\max\{m_1', m_2'\}} &= \frac{2\varepsilon\theta n}{\theta n + y\lceil \frac{4\varepsilon\theta n}{y} \rceil} > \frac{2\varepsilon\theta n}{\theta n + 4\varepsilon\theta n + y} \\ &> \frac{2\varepsilon\theta n}{\theta n + 5\varepsilon\theta n} \geq \frac{2\varepsilon\theta n}{2\theta n} = \varepsilon \end{aligned}$$

which contradicts that the relative error of the estimate is no more than ε. Therefore, $\Omega(\log \varepsilon\theta n)$ bits of memory is required.

The proof of the following lemma adapts the proof techniques given in [6].

LEMMA 2.2. *Suppose that $\theta \leq \frac{1}{16}$ and $\varepsilon \geq \frac{1}{4\theta n}$. Any data structure for the Significant One Counting problem with threshold θ and relative error bound ε uses $\Omega(\frac{1}{\varepsilon}\log^2 \frac{1}{\theta})$ bits of memory.*

Proof. Let $\varepsilon = \frac{1}{k}$ for some integer k. Since $\varepsilon \geq \frac{1}{4\theta n}$, $k \leq 4\theta n$. We list the bits in the stream from left to right, where the rightmost bit is the most recent bit. A window of size n is divided from right to left into blocks of size B, $2B$, $4B$, $8B$, ..., 2^jB, where $B \geq \frac{k}{4}$ and $j = \lfloor \log \frac{n}{B} \rfloor - 1$. For a block of size 2^iB, it is further divided into B contiguous sub-blocks of size 2^i bits, as shown in Figure 1.

Among the B blocks, $k/4$ of them are all 1-bits, while the rest are all 0-bits. Let $d = \lceil \log(\frac{4\theta n}{k} + 1) \rceil$. Note that $d = \lceil \log(\frac{4\theta n + k}{k}) \rceil \leq \lceil \log(\frac{8\theta n}{k}) \rceil$. For a block of size $s > 2^dB$, there are $\binom{B}{k/4}$ possible arrangements for the $k/4$ sub-blocks of 1-bits in the B sub-blocks. For a block of size $s \leq 2^dB$, $k/4$ sub-blocks of the B sub-blocks are fixed to be the sub-blocks of 1-bits. Let b denote the number of blocks of size larger than 2^dB. Therefore, there are $\binom{B}{k/4}^b$ possible arrangements for the sub-blocks of 1-bits in those blocks of size $s > 2^dB$. Also, we have $b \geq j - d \geq (\lfloor \log \frac{n}{B} \rfloor - 1) - \lceil \log(\frac{8\theta n}{k}) \rceil > ((\log \frac{n}{B} - 1) - 1) - (\log \frac{8\theta n}{k} + 1) = \log(\frac{k}{8\theta B}) - 3$.

Consider any two of the $\binom{B}{k/4}^b$ arrangements, say x and y. From right to the left, we can find the first sub-block (the z-th sub-block in the block of size 2^fB, where $z \in [1, B]$ and $f \in [d+1, j]$) such that without loss of generality, it is the c-th sub-block of 1-bits, where $c \in [1, k/4]$, in the block of size 2^fB in arrangement x, and it is a sub-block of 0-bits in arrangement y. Assume, for the sake of contradiction, that after receiving arrangements x and y respectively, the memory states are the same, denoted by M. Then, $n - ((2^f - 1)B + 2^f z)$ 0-bits are input such that the z-th sub-block in the block of size 2^fB is the leftmost sub-block in the sliding window. After receiving this sequence of 0-bits, for arrangement x, the number of 1-bits in the sliding window is $m_1 = (2^f - 1)(k/4) + 2^f c$,

bit stream	1	0	1	1	1	1	1	1	1	1	0	0
stream position (sp)	1	2	3	4	5	6	7	8	9	10	11	12
1-rank	1		2	3	4	5	6	7	8	9		
interesting 1-bit label				1			2			3		
anchor label			1			2			3			4
	Expired								Sliding Window			

bit stream	0	1	0	0	1	0	1	1	1	1	1
stream position (sp)	13	14	15	16	17	18	19	20	21	22	23
1-rank		10			11		12	13	14	15	16
interesting 1-bit label							4			5	
anchor label			5			6			7		
	Sliding Window										

Figure 2: An example bit stream with $\omega = 3$ and sliding window of size $n = 15$.

while for arrangement y, the number of 1-bits in the sliding window is $m_2 = (2^f - 1)(k/4) + 2^f(c-1)$. As the memory states in both cases are also M before receiving the sequence of 0-bits, we should give the same estimate after receiving the sequence of 0-bits. Thus, we will give an estimate with an absolute error at least $|m_1 - m_2|/2 = 2^{f-1}$ for one of the two cases. Consequently, the relative error is at least

$$\begin{aligned}\frac{2^{f-1}}{\max\{m_1, m_2\}} &= \frac{2^{f-1}}{(2^f-1)(\frac{k}{4}) + 2^f c} \geq \frac{2^{f-1}}{2^{f+1}(\frac{k}{4}) - \frac{k}{4}} \\ &> \frac{2^{f-1}}{2^{f-1} \times k} = \frac{1}{k} = \varepsilon\end{aligned}$$

which contradicts that the relative error of the estimate is no more than ε. Thus, we have to differentiate between any two of the $\binom{B}{k/4}^b$ arrangements.

Therefore, the required memory is at least

$$\begin{aligned}\log \binom{B}{\frac{k}{4}}^b &\geq \log \left(\frac{B}{\frac{k}{4}}\right)^{kb/4} \\ &> \frac{k}{4}\left(\log\left(\frac{k}{8\theta B}\right) - 3\right)\left(\log \frac{4B}{k}\right) \\ &= \frac{k}{4}\left(\log \frac{k}{64\theta B}\right)\left(\log \frac{4B}{k}\right)\end{aligned}$$

Since $\theta \leq \frac{1}{16}$ and $B \geq \frac{k}{4}$, by choosing $B = \frac{k}{16}\sqrt{\frac{1}{\theta}}$, the required memory is more than

$$\frac{k}{4}\log^2\left(\frac{1}{4}\sqrt{\frac{1}{\theta}}\right) = \frac{k}{16}\log^2\left(\frac{1}{16\theta}\right)$$

Therefore, $\Omega(\frac{1}{\varepsilon}\log^2\frac{1}{\theta})$ bits of memory is required.

Combining the above two lemmas, we have the following theorem.

THEOREM 2.1. *Suppose that $\theta \leq \frac{1}{16}$ and $\frac{1}{4\theta n} \leq \varepsilon \leq \frac{1}{5}$. Any data structure for the Significant One Counting problem with threshold θ and relative error bound ε requires $\Omega(\frac{1}{\varepsilon}\log^2\frac{1}{\theta} + \log \varepsilon\theta n)$ bits of memory.*

3 A One-Level Data Structure

In this section, we describe a simple data structure for the Significant One Counting problem. First, we need some definitions. Consider any bit stream. Each bit in the stream has a *stream position* (sp), the first bit of the stream has sp = 1, the second one has sp = 2, and etc. Each 1-bit has a *1-rank*, which is its rank among all the 1-bits. Figure 2 shows an example bit stream. Given a positive integer ω, a 1-bit with its 1-rank divisible by ω is called an *interesting 1-bit* (with respect to ω). Each interesting 1-bit has an *interesting 1-bit label* to denote its rank among all the interesting 1-bits. In Figure 2, we have $\omega = 3$. The 1-bit in sp 10 has 1-rank 9, which is divisible by ω. Thus, it is an interesting 1-bit. As it is the third interesting 1-bit, its interesting 1-bit label is 3. Recall that n is the size of the sliding window. If p is the current sp, i.e., the most recent sp, those bits with sp in $[p - n + 1, p]$ are in the sliding window while other bits with sp less than $p - n + 1$ are expired.

3.1 Basic Idea Given the bit stream, the size n of the sliding window and a positive integer ω, we can estimate the number of 1-bits in the sliding window with a bounded absolute error as follows:

> We store the (interesting 1-bit label, sp) pairs of all interesting 1-bits (with respect to ω) in the sliding window in a linked list and keep a variable *count* to count the number of 1-bits arrived after the last interesting 1-bit (or the beginning of the stream if there is no such interesting 1-bit). Let s denote the size of the linked list. The estimate $\hat{m}$ of the number of 1-bits is computed by $\hat{m} = s \times \omega + count$.

For instance, in Figure 2, the linked list contains the (interesting 1-bit label, sp) pairs of all interesting 1-bits in the sliding window, i.e., $\langle(3, 10), (4, 19), (5, 22)\rangle$. The size s of the linked list is 3. As there is one 1-bit arrived after the last interesting 1-bit with label 5, the variable *count* is 1. Therefore, the estimate

$\hat{m} = s \times \omega + count = 3 \times 3 + 1 = 10$.

Intuitively, an interesting 1-bit b in the linked list represents ω 1-bits between its previous interesting 1-bit b_p (not including the 1-bit b_p) and the interesting 1-bit b (including the 1-bit b). A 1-bit in the sliding window will either be counted by $s \times \omega$ or *count*, and this implies $\hat{m} \geq m$. To get an upper bound on $\hat{m}$, we note that if *count* counts an expired 1-bit, there is no interesting 1-bit in the sliding window and thus we have $s = 0$. As $count \leq \omega - 1$, $\hat{m}$ counts at most $\omega - 1$ expired 1-bits, so that $\hat{m} < m + \omega$. If *count* does not count any expired 1-bit, since only the least recent interesting 1-bit b_l in the sliding window can represent expired 1-bits and b_l is not expired, $s \times \omega$ counts at most $\omega - 1$ expired 1-bits. Therefore, we also have $\hat{m} < m + \omega$. In conclusion, we have $m \leq \hat{m} < m + \omega$ and the absolute error of the estimate $\hat{m}$ is at most ω.

The above method can be generalized: A *linked list with dilution i* stores the (interesting 1-bit label, sp) pairs of all interesting 1-bits in the sliding window with its interesting 1-bit label divisible by 2^i. We keep three auxiliary variables: *pos* is the current sp, *nb* is the number of interesting 1-bits in the stream and *count* is the number of 1-bits arrived after the last interesting 1-bit (or the beginning of the stream if there is no such interesting 1-bit). For a linked list with dilution i, let s be the size of the linked list and nb' be the interesting 1-bit label in the tail of the linked list, i.e., the most recent interesting 1-bit in the linked list. If there is no such interesting 1-bit, we set $nb' = 0$. The estimate $\hat{m}$ is computed by $\hat{m} = s \times 2^i\omega + (nb - nb') \times \omega + count$.

For instance, in Figure 2, the *linked list with dilution 2* contains the (interesting 1-bit label, sp) pairs of all interesting 1-bits in the sliding window with its interesting 1-bit label divisible by 2, i.e., $\langle (4, 19) \rangle$. Thus, we have $s = 1$ and $nb' = 4$. Also, the auxiliary variables are $pos = 23$, $nb = 5$ and $count = 1$. Therefore, the estimate is $\hat{m} = s \times 2\omega + (nb - nb') \times \omega + count = 1 \times 6 + (5 - 4) \times 3 + 1 = 10$.

LEMMA 3.1. *The linked list with dilution i returns an estimate $\hat{m}$ of the number m of 1-bits in the sliding window such that $m \leq \hat{m} \leq m + 2^i\omega$, i.e., the absolute error of the estimate is at most $2^i\omega$.*

Proof. Intuitively, an interesting 1-bit b in a linked list with dilution i represents $2^i\omega$ 1-bits between its previous interesting 1-bit b_p with interesting 1-bit label divisible by 2^i (not including the 1-bit b_p) and the interesting 1-bit b (including the 1-bit b). Let $count'$ denote the number of 1-bits arrived after the interesting 1-bit with label nb' (or the beginning of the stream if $nb' = 0$). We have $count' = (nb - nb') \times \omega + count$.

A 1-bit in the sliding window will either be counted by $s \times 2^i\omega$ or $count'$, and this implies $\hat{m} \geq m$. To get an upper bound on $\hat{m}$, we note that if $count'$ counts an expired 1-bit, there is no interesting 1-bit in the sliding window and $s = 0$. Since $count' \leq (2^i - 1)\omega + (\omega - 1) < 2^i\omega$, we have $\hat{m} < m + 2^i\omega$. Suppose that $count'$ does not count any expired 1-bit. As only the least recent interesting 1-bit b_l in the linked list can represent expired 1-bits and b_l is not expired, $s \times 2^i\omega$ counts at most $2^i\omega - 1$ expired 1-bits. Therefore, we also have $\hat{m} < m + 2^i\omega$. In conclusion, we have $m \leq \hat{m} < m + 2^i\omega$ and the absolute error is at most $2^i\omega$.

Since the sliding window is of size n, we can represent a sp in the sliding window by a modulo $2n$ number without ambiguity. Also, there are at most $\lceil n/\omega \rceil$ interesting 1-bits in the sliding window so that an interesting 1-bit label can be represented by a modulo $\lceil 2n/\omega \rceil$ number. Therefore, a sp needs $O(\log n)$ bits and an interesting 1-bit label needs $O(\log(n/\omega))$ bits. Hence, each (interesting 1-bit label, sp) pair in the linked list needs $O(\log(n/\omega) + \log n) = O(\log n)$ bits of memory.

To further reduce the memory usage, notice that the difference of the sp of two consecutive interesting 1-bits is no less than the difference of their 1-ranks, i.e., ω. If there is an interesting 1-bit in sp p, there is no other interesting 1-bit in the sp range $[(\lceil \frac{p}{\omega} \rceil - 1)\omega + 1, (\lceil \frac{p}{\omega} \rceil)\omega]$. Therefore, for an interesting 1-bit in sp p, we can use $p' = \lceil \frac{p}{\omega} \rceil$ to denote its position in the stream without ambiguity, and we will define the set of all possible p' as *anchors*. As p is a modulo $2n$ number, the memory to represent the position of an interesting 1-bit can then be reduced from $O(\log n)$ bits to $O(\log(n/\omega))$ bits and each pair in the linked list needs only $O(\log(n/\omega))$ bits of memory.

Given a positive integer ω, if a sp is divisible by ω, this sp is defined as an *anchor*. Each anchor has an *anchor label* for identification of different anchors. In Figure 2, when $\omega = 3$, those sp's divisible by ω (e.g. $3, 6, 9$) are anchors. The first anchor (sp 3) has an anchor label 1.

For an interesting 1-bit in sp p, we use the anchor label $\lceil \frac{p}{\omega} \rceil$ to denote its position. For instance, in Figure 2, we use the anchor with anchor label $\lceil \frac{4}{3} \rceil = 2$ to denote the position of the interesting 1-bit in sp 4. Intuitively, we shift the sp p of an interesting 1-bit to the next anchor p_a, i.e., sp $p_a = \lceil \frac{p}{\omega} \rceil \omega$. Let p_c be the current sp and assume p_c is not an anchor. If an interesting 1-bit b arrives after the last anchor, the position of b is the anchor label $p' = \lceil \frac{p_c}{\omega} \rceil$. Since $p_c < \lceil \frac{p_c}{\omega} \rceil \omega$, we will denote the position of this interesting 1-bit by the anchor $p'\omega$ with anchor label p', which does not occur yet and is not considered expired. Since there are at most $\lceil \frac{n}{\omega} \rceil + 1$ anchors which are not expired, we use a

modulo ($\lceil \frac{2n}{\omega} \rceil + 2$) number to represent the anchor label.

3.2 The One-Level Data Structure We keep four auxiliary variables: *na* is the number of anchors in the stream, *dp* is the difference of the last anchor and the current sp, *nb* is the number of interesting 1-bits in the stream and *count* is the number of 1-bits arrived after the last interesting 1-bit (or the beginning of the stream if there is no such interesting 1-bit).

A *linked list with dilution i* now stores the (interesting 1-bit label, anchor label) pairs of all interesting 1-bits in the sliding window with its interesting 1-bit label divisible by 2^i. For a linked list with dilution i, we compute the estimate by the same way as before. Let s be the size of the linked list and nb' be the interesting 1-bit label in the tail of the linked list. If there is no such interesting 1-bit, we set $nb' = 0$. The estimate $\hat{m}$ is computed by $\hat{m} = s \times 2^i\omega + (nb - nb') \times \omega + count$.

For instance, in Figure 2, a linked list with dilution 2 contains the (interesting 1-bit label, anchor label) pair of all interesting 1-bits in the sliding window with the interesting 1-bit label divisible by 2, i.e., $\langle (2,3), (4,7) \rangle$. (The interesting 1-bit with interesting 1-bit label 2 is included since the anchor with anchor label 3, i.e., sp 9, is in the sliding window.)

Since the variable *pos* is replaced by variables *na* and *dp*, to determine whether the anchor with anchor label na_h in the head of the linked list is expired, the current position p is first computed by $p = na \times \omega + dp$. If $na_h \times \omega \leq p - n$, the head is expired. (Since the anchor label is represented by a modulo ($\lceil \frac{2n}{\omega} \rceil + 2$) number, the actual operation to determine whether na_h is expired is: if $na_h \times \omega > p$, the head is expired when $(na_h - (\lceil \frac{2n}{\omega} \rceil + 2)) \times \omega \leq p - n$; otherwise, the head is expired when $na_h \times \omega \leq p - n$. For simplification, operation for modulo numbers is not explicitly used.)

THEOREM 3.1. *For a linked list with dilution i, the above procedure returns the estimate $\hat{m}$ of the number m of 1-bits in the sliding window such that $m \leq \hat{m} \leq m + 2^{i+1}\omega$, i.e., the absolute error of the estimate is at most $2^{i+1}\omega$.*

Proof. If the current sp is p and the least recent anchor a in the sliding window is sp p_a, we have $p_a \geq p - n + 1$. Thus, only expired interesting 1-bits with sp at least $p - n + 1 - (\omega - 1) = p - n - \omega + 2$ can have its position shifted to anchor a. As there is at most one interesting 1-bit in the sp range $[p - n - \omega + 2, p - n]$, at most one expired interesting 1-bit will be treated as in the sliding window.

In a linked list with dilution i, at most one expired interesting 1-bit is stored in the linked list. Let s_0 be the number of interesting 1-bits in the sliding window with label divisible by 2^i, and $\hat{m}_0$ be the estimate obtained from a linked list with dilution i by the procedure for Lemma 3.1. Therefore, we have $s_0 \leq s \leq s_0 + 1$. Since $\hat{m} = s \times 2^i\omega + (nb - nb') \times \omega + count \geq s_0 \times 2^i\omega + (nb - nb') \times \omega + count = \hat{m}_0$, by Lemma 3.1, $\hat{m} \geq \hat{m}_0 \geq m$. On the other hand, $\hat{m} = s \times 2^i\omega + (nb - nb') \times \omega + count \leq (s_0 + 1) \times 2^i\omega + (nb - nb') \times \omega + count = \hat{m}_0 + 2^i\omega$. By Lemma 3.1, it follows that $\hat{m} \leq \hat{m}_0 + 2^i\omega \leq m + 2^i\omega + 2^i\omega = m + 2^{i+1}\omega$. Therefore, the absolute error of the estimate is at most $2^{i+1}\omega$.

By Theorem 3.1, the estimate obtained from a linked list with dilution 0 has absolute error at most 2ω. Thus, for $m \geq n/2^l$, the relative error is at most $2\omega/m \leq 2\omega/(n/2^l) = 2^{l+1}\omega/n$. Let $\omega_{l,\varepsilon}$ be the value of ω such that the relative error of the estimate is at most ε for $m \geq n/2^l$. Therefore, we have $2^{l+1}\omega_{l,\varepsilon}/n = \varepsilon$, i.e., $\omega_{l,\varepsilon} = \varepsilon n/2^{l+1}$.

To solve the Significant One Counting problem, we use a linked list with dilution 0. We set $l = \log(1/\theta)$ such that $n/2^l = \theta n$ and $\omega_{l,\varepsilon} = \varepsilon n/2^{l+1} = \varepsilon\theta n/2$. For the auxiliary variables, *na* and *nb* need $O(\log \frac{n}{\omega}) = O(\log \frac{2}{\varepsilon\theta}) = O(\log \frac{1}{\varepsilon\theta})$ bits, and *dp* and *count* need $O(\log \omega) = O(\log \varepsilon\theta n)$ bits. Since each (interesting 1-bit label, anchor label) pair in the linked list needs $O(\log \frac{n}{\omega}) = O(\log \frac{1}{\varepsilon\theta})$ bits and the size of the linked list is at most $\lceil \frac{n}{\omega} \rceil = O(\frac{1}{\varepsilon\theta})$, the linked list requires $O(\frac{1}{\varepsilon\theta} \log \frac{1}{\varepsilon\theta})$ bits of memory. By storing the differences of (interesting 1-bit label, anchor label) pairs in the linked list except for the head of the list, the memory can be reduced to $O(\log \frac{1}{\varepsilon\theta} + \frac{1}{\varepsilon\theta} \log(\frac{2/\varepsilon\theta}{1/\varepsilon\theta})) = O(\log \frac{1}{\varepsilon\theta} + \frac{1}{\varepsilon\theta}) = O(\frac{1}{\varepsilon\theta})$ bits. Therefore, the total memory needed is $O(\frac{1}{\varepsilon\theta} + \log \frac{1}{\varepsilon\theta} + \log \varepsilon\theta n) = O(\frac{1}{\varepsilon\theta} + \log \varepsilon\theta n)$.

4 Improvement in Memory: A Multilevel Data Structure

In the one-level data structure, the absolute error of the estimate is $2\omega_{l,\varepsilon} = \varepsilon\theta n$ regardless of the number m of 1-bits in the sliding window. In fact, if the number m of 1-bits in the sliding window is large, i.e., $m > \theta n$, the absolute error allowed is εm, which is larger than $\varepsilon\theta n$. In this section, we improve the memory. Throughout our discussion, we assume that $\omega = \omega_{l,\varepsilon}$, which is equal to $\varepsilon n/2^{l+1}$. Recall that this value of ω guarantees that the relative error of the estimate given by the one-level data structure (with dilution 0) is at most ε for $m \geq n/2^l$.

LEMMA 4.1. *If $m \geq n/2^{l-i}$, the estimate $\hat{m}$ of the number m of 1-bits in the sliding window obtained from a linked list with dilution i has relative error at most ε.*

Proof. By Theorem 3.1, the estimate $\hat{m}$ has absolute error at most $2^{i+1}\omega = 2^{i+1}\omega_{l,\varepsilon} = (2^{i+1})(\varepsilon n/2^{l+1}) =$

$\varepsilon n/2^{l-i}$. Therefore, if $m \geq n/2^{l-i}$, the relative error is at most $(\varepsilon n/2^{l-i})/m \leq \varepsilon$.

LEMMA 4.2. *Let m be the number of 1-bits in the sliding window. If a linked list with dilution i has size at least $\lceil n/(2^{l-1}\omega_{l,\varepsilon})\rceil+2$ interesting 1-bits, then $m \geq n/2^{l-i-1}$.*

Proof. Let s be the size of the linked list. Recall that $\hat{m} = s \times 2^i\omega + (nb - nb') \times \omega + count$. By Theorem 3.1, if an estimate $\hat{m}$ of m is obtained from a linked list with dilution i, we have $m \leq \hat{m} \leq m + 2^{i+1}\omega_{l,\varepsilon}$. It follows that $m \geq \hat{m} - 2^{i+1}\omega_{l,\varepsilon} = s \times 2^i\omega_{l,\varepsilon} + (nb - nb') \times \omega_{l,\varepsilon} + count - 2^{i+1}\omega_{l,\varepsilon} \geq (\lceil n/(2^{l-1}\omega_{l,\varepsilon})\rceil+2) \times 2^i\omega_{l,\varepsilon} + count - 2^{i+1}\omega_{l,\varepsilon} \geq n/2^{l-i-1}$.

We use the same set of auxiliary variables $\{na, nb, dp, count\}$ and set $\omega = \omega_{l,\varepsilon}$ as before. However, instead of keeping a linked list with dilution 0, we keep l levels of linked list, numbered 0 to $l-1$. The level k linked list is a linked list with dilution k of size at most $\lceil n/(2^{l-1}\omega_{l,\varepsilon})\rceil + 2$, which keeps the (interesting 1-bit label, anchor label) pairs of the $\lceil n/(2^{l-1}\omega_{l,\varepsilon})\rceil + 2$ most recent interesting 1-bits in the sliding window with interesting 1-bit label divisible by 2^k. For each level $i \in [0, l-1]$ linked list, we introduce the variables pair (nb_i, na_i), which is set to be the (interesting 1-bit label, anchor label) pair of most recent interesting 1-bit removed from the linked list. Initially, (nb_i, na_i) is set to $(0, 0)$. To return an estimate $\hat{m}$, we first find the smallest level j which contains the most recent expired interesting 1-bit in (nb_j, na_j) among all levels, i.e., na_j is the anchor label of the most recent expired anchor among all levels. The estimate is then obtained from the level j linked list: Let s be the size of the level j linked list and nb' be the interesting 1-bit label in the tail of the level j linked list. If there is no such interesting 1-bit, we set $nb' = 0$. The estimate $\hat{m}$ is computed by $\hat{m} = s \times 2^j\omega + (nb - nb') \times \omega + count$.

THEOREM 4.1. *Let $l = \log\frac{1}{\theta}$ (or equivalently $\omega_{l,\varepsilon} = \varepsilon\theta n/2$). The above procedure returns an estimate $\hat{m}$ of the number m of 1-bits in the sliding window with a relative error at most ε for $m \geq \theta n$ and an absolute error at most $\varepsilon\theta n$ for $m < \theta n$, using $O(\frac{1}{\varepsilon}\log^2\frac{1}{\theta} + \log\varepsilon\theta n)$ bits of memory.*

Proof. In the above procedure, if a level $i \in [0, l-1]$ linked list has (nb_i, na_i) where na_i is the anchor label of an anchor in the sliding window, then the level i linked list is truncated to have a size at most $\lceil n/(2^{l-1}\omega_{l,\varepsilon})\rceil+2$. Note that when a linked list with dilution $i \in [0, l-1]$ is truncated, we can't apply Theorem 3.1 and Lemma 4.1 on the estimate from this linked list. The number of interesting 1-bits in the sliding window with interesting 1-bit label divisible by 2^{l-1} is at most $\lceil n/2^{l-1}\omega_{l,\varepsilon}\rceil < \lceil n/2^{l-1}\omega_{l,\varepsilon}\rceil + 2$. Therefore, the level $l-1$ linked list can store all the interesting 1-bits in the sliding window with label divisible by 2^{l-1}. Since the level $l-1$ linked list is not truncated, among all the levels, there is at least one level (i.e., level $l-1$) that Theorem 3.1 and Lemma 4.1 can be applied on the estimate from that level.

Note that the interesting 1-bit with label nb_j is the most recent expired 1-bit in the data structure and nb_j is divisible by 2^j since it is removed from the level j linked list. As na_j is the anchor label of an expired anchor, the level j linked list is not truncated so that Theorem 3.1 and Lemma 4.1 can be applied on the estimate from the level j linked list. If $j > 0$, as nb_j is divisible by 2^j, nb_j is also divisible by 2^{j-1}. Since level j is the smallest level containing this interesting 1-bit, level $j-1$ must keep $\lceil n/2^{l-1}\omega_{l,\varepsilon}\rceil+2$ interesting 1-bits in the sliding window. By Lemma 4.2, we have $m \geq n/2^{l-(j-1)-1} = n/2^{l-j}$. Therefore, by Lemma 4.1, the estimate $\hat{m}$ obtained from the level j linked list has relative error at most ε. Note that for $j > 0$, we have $m \geq n/2^{l-j} \geq n/2^{l-1} = 2(n/2^l)$. If $j = 0$, since $\omega = \omega_{l,\varepsilon}$, the estimate obtained from the level 0 linked list has relative error at most ε for $m \geq n/2^l$. When $\omega_{l,\varepsilon} = \varepsilon n/2^{l+1} = \varepsilon\theta n/2$, we have $l = \log(1/\theta)$. Therefore, for $m \geq n/2^l = \theta n$, the procedure returns an estimate $\hat{m}$ with a relative error at most ε. Also, for $m < \theta n$, the estimate is obtained from the level 0 linked list, i.e., a linked list with dilution 0. By Theorem 3.1, the absolute error of the estimate is at most $2\omega = 2\omega_{l,\varepsilon} = \varepsilon\theta n$.

The auxiliary variables still needs $O(\log\frac{1}{\varepsilon\theta} + \log\varepsilon\theta n)$ bits of memory. Since each (interesting 1-bit label, anchor label) pair in the linked list and (nb_i, na_i) for $i \in [0, l-1]$ need $O(\log\frac{n}{\omega}) = O(\log\frac{1}{\varepsilon\theta})$ bits and the size of a linked list is at most $\lceil n/2^{l-1}\omega_{l,\varepsilon}\rceil + 2 = \lceil\frac{4}{\varepsilon}\rceil + 2 = O(\frac{1}{\varepsilon})$, a level i linked list with (nb_i, na_i) requires $O((\frac{1}{\varepsilon}+1)\log\frac{1}{\varepsilon\theta}) = O(\frac{1}{\varepsilon}\log\frac{1}{\varepsilon\theta})$ bits of memory. By storing the differences of (interesting 1-bit label, anchor label) pairs in the linked list except for the head of the linked list, the memory can be reduced to $O(2\log\frac{1}{\varepsilon\theta} + \frac{1}{\varepsilon}\log(\frac{2/\varepsilon\theta}{4/\varepsilon})) = O(\frac{1}{\varepsilon}\log\frac{1}{\theta})$ bits. Therefore, the total memory needed is $O((\frac{1}{\varepsilon}\log\frac{1}{\theta})l + \log\frac{1}{\varepsilon\theta} + \log\varepsilon\theta n + 1) = O(\frac{1}{\varepsilon}\log^2\frac{1}{\theta} + \log\varepsilon\theta n)$ bits.

5 Improvement in Time: The Optimal Data Structure

Note that by setting $l = \log\frac{1}{\theta}$, the data structure described in the last section takes $O(l) = O(\log\frac{1}{\theta})$ time to answer a query because it takes $O(l)$ time to find the smallest level j containing the most recent expired interesting 1-bit with label nb_j among all the levels.

Also, for the per-item processing, we have to insert and remove an interesting 1-bit to/from at most l levels, which also costs $O(l) = O(\log \frac{1}{\theta})$ time. In this section, we improve the per-item processing time and query time to $O(1)$ worst case.

In our optimal data structure, we let $l = \log \frac{1}{\theta}$ and $\omega = \omega_{l,\varepsilon}$, which is $\varepsilon\theta n/2$. We still keep the set of the four auxiliary variables $\{na, nb, dp, count\}$ as in Section 3.2 and keep the l levels of linked list as in the last section. In addition, we use one more auxiliary variable lb to store the interesting 1-bit label of the most recent expired interesting 1-bit removed from the l levels, which is 0 initially. We still have $\omega = \omega_{l,\varepsilon} = \varepsilon\theta n/2$. The estimate $\hat{m}$ is then computed by $\hat{m} = (nb - lb) \times \omega + count$. Also, we insert an interesting 1-bit only to the maximum level containing it. Thus, the size of the level $i \in [0, l-2]$ linked list is halved, i.e., $\frac{1}{2}(\lceil n/2^{l-1}\omega_{l,\varepsilon} \rceil + 2)$, while the size of the level $l-1$ linked list is not changed. Furthermore, we keep one more linked list L of (interesting 1-bit label, anchor label) pairs of all the interesting 1-bit stored in the l levels so that only $O(1)$ time is needed to remove an interesting 1-bit. The updating algorithm is shown as follows, where $\omega = \omega_{l,\varepsilon} = \varepsilon\theta n/2$:

Initially, variables na, nb, dp, $count$ and lb are set to 0.

For per-item processing (when a bit b arrives):

1. Increment dp.
2. If $dp = \omega$ (i.e., $0 \mod \omega$), increment na.
3. Check for expiration: If the head (nb_h, na_h) of the linked list L has expired, i.e., $na_h \times \omega \leq na \times \omega + dp - n$, set $lb = nb_h$ and remove (nb_h, na_h) from L and thus the corresponding level.
4. If $b = 1$, increment $count$.
5. If $b = 1$ and $count = \omega$ (i.e., $count = 0 \mod \omega$),
   - Increment nb.
   - Determine the largest level k such that nb is a multiple of 2^k.
   - If the level k linked list reaches its the maximum size, remove its head from the level k linked list and thus the linked list L at the same time.
   - If $dp = 0$, add (nb, na) to the tail of the level k linked list and the tail of L; otherwise, add $(nb, na+1)$ to the tail of the level k linked list and the tail of L.

For answering a query:

Return an estimate $\hat{m} = (nb - lb) \times \omega + count = (nb - lb) \times (\varepsilon\theta n/2) + count$.

THEOREM 5.1. *The above query algorithm returns an estimate $\hat{m}$ of the number m of 1-bits in the sliding window with a relative error at most ε, i.e., $m \leq \hat{m} \leq m + \varepsilon m$, for $m \geq \theta n$ and an absolute error at most $\varepsilon\theta n$, i.e., $m \leq \hat{m} \leq m + \varepsilon\theta n$, for $m < \theta n$ using $O(\frac{1}{\varepsilon}\log^2 \frac{1}{\theta} + \log \varepsilon\theta n)$ bits of memory with $O(1)$ per-item processing and $O(1)$ query time.*

Proof. Let level j be the minimum level containing the interesting 1-bit with label lb when it is expired. Let nb_t be the interesting 1-bit label in the tail of the level j linked list. The level j linked list thus has size $s = (nb_t - lb)/2^j$. By the procedure for Theorem 3.1, the estimate from the linked list with dilution j is computed by $\hat{m} = s \times 2^j\omega + (nb - nb_t) \times \omega + count = ((nb_t - lb)/2^j) \times 2^j\omega + (nb - nb_t) \times \omega + count = (nb - lb) \times \omega + count$. Therefore, the estimate is equal to that obtained from the level j linked list. Note that in the proof of Theorem 4.1, the interesting 1-bit with label nb_j is the most recent expired 1-bit in the data structure and the interesting 1-bit with label nb_j is removed from the level j linked list. Thus, we have $lb = nb_j$. By Theorem 4.1, the above algorithm returns an estimate $\hat{m}$ of the number m of 1-bits in the sliding window with a relative error at most ε, i.e., $m \leq \hat{m} \leq m + \varepsilon m$, for $m \geq \theta n$ and an absolute error at most $\varepsilon\theta n$, i.e., $m \leq \hat{m} \leq m + \varepsilon\theta n$, for $m < \theta n$.

As the variable lb needs $O(\log \frac{n}{\omega}) = O(\log \frac{1}{\varepsilon\theta})$ bits of memory, the set of auxiliary variables still needs $O(\log \frac{1}{\varepsilon\theta} + \log \varepsilon\theta n)$ bits of memory. Since each (interesting 1-bit label, anchor label) pair in the linked list needs $O(\log \frac{1}{\varepsilon\theta})$ bits and the size of a linked list remains $O(\frac{1}{\varepsilon})$, a linked list requires $O(\frac{1}{\varepsilon}\log \frac{1}{\varepsilon\theta})$ bits of memory. By storing differences of (interesting 1-bit label, anchor label) pairs in the linked list except for the head of the linked list, the memory can be reduced to $O(\log \frac{1}{\varepsilon\theta} + \frac{1}{\varepsilon}\log(\frac{2/\varepsilon\theta}{2/\varepsilon})) = O(\frac{1}{\varepsilon}\log \frac{1}{\theta})$ bits. Therefore, the total memory needed is still $O(\frac{1}{\varepsilon}\log^2 \frac{1}{\theta} + \log \varepsilon\theta n)$. Moreover, in the algorithm, all operations in per-item processing and query can be done in $O(1)$ worst case time.

References

[1] A. Arasu and G. Manku. Approximate counts and quantiles over sliding windows. In *Proc. 23rd ACM Symp. on Principles of Database Systems (PODS)*, 2004.

[2] B. Babcock, M. Datar, and R. Motwani. Sampling from a moving window over streaming data. In *Proc. 13th annual ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pages 633–634, 2002.

[3] B. Babcock, M. Datar, R. Motwani, and L. O'Callaghan. Maintaining variance and k-medians over data stream windows. In *Proc. of 22nd Symp. on Principles of Database Systems (PODS)*, 2003.

[4] M. Charikar, K. Chen, and M. Farach-Colton. Finding frequent items in data streams. In *Proc. 29th International Colloquium on Automata, Languages, and Programming (ICALP)*, 2002.

[5] G. Cormode and S. Muthukrishnan. What's hot and what's not: Tracking most frequent items dynamically. In *Proc. of ACM Principles of Database Systems (PODS)*, 2003.

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

[7] M. Datar and S. Muthukrishnan. Estimating rarity and similarity over data stream windows. In *Proc. of European Symposium of Algorithms (ESA)*, 2002.

[8] E. D. Demaine, A. Lopez-Ortiz, and J. I. Munro. Frequency estimation of internet packet streams with limited space. In *Proc. the European Symposium on Algorithms (ESA)*, pages 348–360, 2002.

[9] C. Estan and G. Varghese. New directions in traffic measurement and accounting. In *Proc. ACM SIGCOMM*, 2002.

[10] J. Fong and M. Strauss. An approximate L^p-difference algorithm for massive data streams. In *Proc. 17th Symp. on Theoretical Aspects of Computer Science (STACS)*, pages 193–204, 2000.

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

[12] P. B. Gibbons and Y. Matias. Synopsis data structures for massive data sets. In J. M. Abello and J. S. Vitter, editors, *External Memory Algorithms*, number 50 in DIMACS: Series in Discrete Mathematics and Theoretical Computer Science, pages 39–70. AMS, 1999. A two page summary appeared as a short paper in SODA'99.

[13] P. B. Gibbons and S. Tirthapura. Estimating simple functions on the union of data streams. In *Proc. 13th ACM Symp. on Parallel Algorithms and Architectures (SPAA)*, pages 281–290, 2001.

[14] P. B. Gibbons and S. Tirthapura. Distributed streams algorithms for sliding windows. In *Proc. ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 63–72, 2002.

[15] A. C. Gilbert, Y. Kotidis, and M. J. Strauss. Surfing wavelets on streams: One-pass summaries for approximate aggregate queries. In *Proc. 27th International Conf. on Very Large Data Bases (VLDB)*, pages 79–88, 2001.

[16] S. Guha, N. Koudas, and K. Shim. Data-streams and histograms. In *Proc. 33rd ACM Symp. on Theory of Computing (STOC)*, pages 471–475, 2001.

[17] P. Indyk. Stable distributions, pseudorandom generators, embeddings and data stream computation. In *Proc. 41st IEEE Symp. on Foundations of Computer Science (FOCS)*, pages 189–197, 2000.

[18] C. Jin, W. Qian, C. Sha, J. X. Yu, and A. Zhou. Dynamically maintaining frequent items over a data stream. In *Proc. of the ACM International Conf. on Information and Knowledge Management (CIKM)*, 2003.

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

[20] A. Manjhi, V. Shkapenyuk, K. Dhamdhere, and C. Olston. Finding (recently) frequent items in distributed data streams. In *Proc. 21st International Conference on Data Engineering (ICDE)*, 2005.