Streaming Algorithms with Few State Changes

RAJESH JAYARAM, Google Research, USA
DAVID P. WOODRUFF, Carnegie Mellon University and Google Research, USA
SAMSON ZHOU, Texas A&M University, USA

In this paper, we study streaming algorithms that minimize the number of changes made to their internal state (i.e., memory contents). While the design of streaming algorithms typically focuses on minimizing space and update time, these metrics fail to capture the asymmetric costs, inherent in modern hardware and database systems, of reading versus writing to memory. In fact, most streaming algorithms write to their memory on *every update*, which is undesirable when writing is significantly more expensive than reading. This raises the question of whether streaming algorithms with small space *and* number of memory writes are possible.

We first demonstrate that, for the fundamental F_p moment estimation problem with $p \geq 1$, any streaming algorithm that achieves a constant factor approximation must make $\Omega(n^{1-1/p})$ internal state changes, regardless of how much space it uses. Perhaps surprisingly, we show that this lower bound can be matched by an algorithm which also has near-optimal space complexity. Specifically, we give a $(1+\varepsilon)$-approximation algorithm for F_p moment estimation that use a near-optimal $\widetilde{O}_\varepsilon(n^{1-1/p})$ number of state changes, while simultaneously achieving near-optimal space, i.e., for $p \in [1, 2)$, our algorithm uses poly $\left(\log n, \frac{1}{\varepsilon}\right)$ bits of space for, while for $p > 2$, the algorithm uses $\widetilde{O}_\varepsilon(n^{1-1/p})$ space. We similarly design streaming algorithms that are simultaneously near-optimal in both space complexity and the number of state changes for the heavy-hitters problem, sparse support recovery, and entropy estimation. Our results demonstrate that an optimal number of state changes can be achieved without sacrificing space complexity.

CCS Concepts: • **Theory of computation → Streaming, sublinear and near linear time algorithms**.

Additional Key Words and Phrases: streaming algorithms, memory states, moment estimation

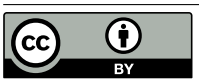
ACM Reference Format:
Rajesh Jayaram, David P. Woodruff, and Samson Zhou. 2024. Streaming Algorithms with Few State Changes. *Proc. ACM Manag. Data* 2, 2 (PODS), Article 82 (May 2024), 28 pages. https://doi.org/10.1145/3651145

1 INTRODUCTION

The streaming model of computation is a central paradigm for computing statistics for datasets that are too large to store. Examples of such datasets include internet traffic logs, IoT sensor networks, financial transaction data, database logs, and scientific data streams (such as huge experiments in particle physics, genomics, and astronomy). In the one-pass streaming model, updates to an underlying dataset are processed by an algorithm one at a time, and the goal is to approximate, collect, or compute some statistic of the dataset while using space that is sublinear in the size of the dataset (see [10, 83] for surveys).

Formally, an insertion-only data stream is modeled by a sequence of updates $u_1, \ldots, u_m$, each of the form $u_t \in [n]$ for $t \in [m]$, where $[n] = \{1, \ldots, n\}$ is the universe size. The updates implicitly define an underlying frequency vector $f \in \mathbb{R}^n$ by $f_i = |\{t \mid u_t = i\}|$, so that the value of each

Authors' addresses: Rajesh Jayaram, Google Research, New York City, NY, USA, rkjayaram@google.com; David P. Woodruff, Carnegie Mellon University and Google Research, Pittsburgh, PA, USA, dwoodruf@cs.cmu.edu; Samson Zhou, Texas A&M University, College Station, TX, USA, samsonzhou@gmail.com.



ACM 2836-6573/2024/5-ART82
https://doi.org/10.1145/3651145

coordinate of the frequency vector is the number of occurrences of the coordinate identity in the data stream.

One of the most fundamental problems in the streaming literature is to compute a $(1+\varepsilon)$ approximation of the F_p moment, defined by $F_p(f) = (f_1)^p + \ldots + (f_n)^p$, where $\varepsilon > 0$ is an accuracy parameter. The frequency moment estimation problem has been the focus of more than two decades of study in the streaming model [2, 5, 12, 15, 17, 32, 50, 55, 58, 60, 62, 68, 74, 76, 92, 94, 95]. In particular, F_p-estimation is used for $p = 0.25$ and $p = 0.5$ in mining tabular data [42], for $p = 1$ in network traffic monitoring [49] and dynamic earth-mover distance approximation [54], and for $p = 2$ in estimating join and self-join sizes [4] and in detecting network anomalies [89].

Another fundamental streaming problem is to compute L_p-heavy hitters: given a threshold parameter $\varepsilon \in (0, 1]$, the L_p-heavy hitters problem is to output a list L containing all $j \in [n]$ such that $f_j \geq \varepsilon \cdot \|f\|_p$, and no $j \in [n]$ with $f_j < \frac{\varepsilon}{2} \cdot \|f\|_p$. The heavy-hitter problem is used for answering iceberg queries [48] in database systems, finding elephant flows and spam prevention in network traffic monitoring [14], and perhaps has an even more extensive history than the F_p moment estimation problem [19–22, 27, 36, 43, 56, 57, 70, 71, 75, 78, 80, 81].

The primary goal of algorithm design for the streaming model is to minimize the space and update time of the algorithm. However, the generic per-update processing time fails to capture the nuanced reality of many modern database and hardware systems, where the *type* of updates which are made on a time step matter significantly for the real-world performance of the algorithm. Specifically, it is typically the case that updates which only require *reads* to the memory contents of the algorithm are significantly faster than updates which also modify the memory of the algorithm, i.e., writes. Thus, while many streaming problems are well understood in terms of their space and update time, little is known about their *write* complexity: namely, the number of state changes they make over the course of the stream.

In this paper, we propose the number of state changes of a streaming algorithm as a complexity-theoretic parameter of interest, and make the case for its importance as a central object of study, in addition to the space and update-time of an algorithm. While there is significant practical motivation for algorithms which update their internal state infrequently (see Section 1.1 for a discussion), from a theoretical perspective it is not clear that having few state changes is even possible. Specifically, most known streaming algorithms write to their memory contents on *every* update of the stream. Moreover, even if algorithms using fewer state changes existed, such algorithms would not be useful if they required significantly more space than prior algorithms (which do not minimize the number of state changes). Thus, we address the following question:

Is it possible to design streaming algorithms which make few updates to their internal state in addition to being space efficient?

Our main contribution is to answer this question positively. Specifically, we demonstrate that algorithms exist which are simultaneously near-optimal in their space and state-change complexity. This demonstrates further that we do not need to pay extra in the space complexity to achieve algorithms with few state changes.

1.1 Motivation for Minimizing State Changes

Asymmetric read/write costs of non-volatile memory. The primary motivation of minimizing state changes arises from the simple observation that different actions performed over the same allocated memory may have different costs. For example, non-volatile memory (NVM) is low latency memory that can be retained when power to the system is turned off and therefore can dramatically increase system performance. Although NVM offers many benefits over dynamic random access memory (DRAM), writing to NVM is significantly more costly than reading from NVM, incurring higher

energy costs, higher latency, and suffering lower per-chip bandwidth [18]. In fact, [79] noted that an NVM cell generally wears out between 10^8 and 10^{12} writes. Hence, in contrast to DRAM, NVM has a significant asymmetry between read and write operation costs [3, 8, 44, 45, 69, 87, 96], which has been the focus of several works in system design [39, 72, 97, 98] and database data structure design [37, 90, 91].

One specific type of non-volatile memory is NAND flash memory, which is an electronic memory storage unit that can be electrically erased, written, and read in blocks that are generally significantly smaller than the entire device. NAND flash memory can be found in a number of common devices, such as smartphones, USB flash drives, memory cards, or solid-state drives. However, [13] notes that, at the time, individual NAND flash memory cells would wear out after 10^4 to 10^6 write/erase operations. Indeed, Apple notes that "all iOS devices and some macOS devices use a solid-state drive (SSD) for permanent storage" and recommends minimizing disk writes to optimize device performance [7]. Although a line of work considered wear leveling to limit memory wear [34, 35, 59], they did not immediately produce high-probability wear guarantees, thus motivating work that focused on hashing algorithms that choose which memory cell to write/overwrite each item, depending on the previous number of writes already incurred by that cell [46]. Subsequently, the development of specific system software, e.g., the garbage collector, virtual memory manager, or virtual machine hypervisor, automatically handled balancing write operations across memory cells over long time horizons, so that an individual cell would not fail much faster than the overall unit. Therefore, subsequent works focused on minimizing the overall number of write operations [18] to the device rather than minimizing the number of write operations to a specific cell.

Asymmetric read/write costs of large data storage systems. Power consumption is a major consideration for large enterprise data storage subsystems, which can often impact the density of servers and the total cost of ownership [38, 99]. In [11], it was observed that given steady hit ratios of read operations, write operations will eventually dominate file system performance and thus be the main source of power consumption. Indeed, [84, 88] noted that there are substantial periods of time where all the traffic in the request stream to the Microsoft data storage centers servicing applications such as Hotmail and Messenger is write traffic.

In addition, for distributed data systems that each serve a number of clients, even when one server is updated, they must periodically notify the other servers about their changes to maintain some level of synchronization across the entire system. Therefore, write operations to a single server can still induce expensive overheads from communication costs between servers, and thus, reducing the number of writes has long been a goal in disk arrays, distributed systems, and cache-coherent multiprocessors.

Challenges with deterministic algorithms. From a theoretical perspective, minimizing the number of internal state changes immediately rules out a large class of deterministic streaming algorithms. For example, counting the stream length can be performed deterministically by simply repeatedly updating a counter over the course of a stream. Similarly, L_1-heavy hitters can be tracked deterministically and in sublinear space using the well-known Misra-Gries data structure [81]. Another example is the common merge-and-reduce technique for clustering in the streaming model, which can be implemented deterministically if the coreset construction in each reduce step is deterministic. Other problems such as maintaining Frequent Directions [51] or L_2 regression in the row arrival model [23, 25] also admit highly non-trivial deterministic algorithms that use sublinear space. However, these approaches all update the algorithm upon each stream update and thus seem inherently at odds with achieving a sublinear number of internal state changes over the course of the stream.

Relationship with sampling. On the other hand, sampling-based algorithms inherently seem useful for minimizing the number of internal state changes. There are a number of problems that admit sublinear-size coresets based on importance sampling, such as clustering [24, 26, 41], graph sparsification [1, 28], linear regression [40], and low-rank approximation [23]. These algorithms generally assign some value quantifying the "importance" of each stream update as it arrives and then sample the update with probability proportional to the importance. Thus if there are few additional operations, then the number of internal state changes can be as small as the overall data structure maintained by the streaming algorithm. On the other hand, it is not known that space-optimal sampling algorithms exist for a number of other problems that admit sublinear-space streaming algorithms, such as F_p estimation, L_p-heavy hitters, distinct elements, and entropy estimation. Hence, a natural question is to ask whether there exist space-optimal streaming algorithms for all of these problems that also incur a small number of internal state changes.

1.2 Our Contributions

In this work, we initiate the study of streaming algorithms which minimize state changes, and demonstrate the existence of algorithms which achieve optimal or near-optimal space bounds while simultaneously achieving an optimal or near optimal number of internal state changes.

Heavy-hitters. We first consider the L_p-heavy hitters problem, where the goal is to output estimates $\widehat{f_j}$ to the frequency f_j of every item $j \in [n]$ such that $\left|\widehat{f_j} - f_j\right| \leq \frac{\varepsilon}{2} \cdot \|f\|_p$, given an input accuracy parameter $\varepsilon \in (0, 1)$. Note that under such a guarantee, along with a 2-approximation of $\|f\|_p$, we can automatically output a list that contains all $j \in [n]$ such that $f_j \geq \varepsilon \cdot \|f\|_p$ but also no index $j \in [n]$ such that $f_j < \frac{\varepsilon}{4} \cdot \|f\|_p$. We defer discussion of how to obtain a 2-approximation to $\|f\|_p$ for the moment and instead focus on the additive error guarantee for all $\widehat{f_j}$. Our main result for the heavy hitters problem is the following:

Theorem 1.1. *Given a constant $p \geq 1$, there exists a one-pass insertion-only streaming algorithm that has $O\left(n^{1-1/p}\right) \cdot \text{poly}\left(\log(nm), \frac{1}{\varepsilon}\right)$ internal state changes, and solves the L_p-heavy hitter problem, i.e., it outputs a frequency vector $\widehat{f}$ such that*

$$\mathbf{Pr}\left[\|\widehat{f} - f\|_\infty \leq \frac{\varepsilon}{2} \cdot \|f\|_p\right] \geq \frac{2}{3}.$$

For $p \in [1, 2)$, the algorithm uses $O\left(\frac{\log^{9+3p}(mn)}{\varepsilon^{4+4p}}\right)$ bits of space, while for $p > 2$, the algorithm uses $\tilde{O}\left(\frac{1}{\varepsilon^{4+2p}} n^{1-2/p}\right)$ bits of space.

We next give a lower bound showing that any approximation algorithm achieving a $(2 - \Omega(1))$-approximation to F_p requires $\Omega(n^{1-1/p})$ state updates.

Theorem 1.2. *Let $\varepsilon \in (0, 1)$ be a constant and $p \geq 1$. Any algorithm that solves the L_p-heavy hitters problem with threshold ε with probability at least $\frac{2}{3}$ requires at least $\frac{1}{2\varepsilon} n^{1-1/p}$ state updates.*

Together, Theorem 1.2 and Theorem 1.1 show that we achieve a near-optimal number of internal state changes. Furthermore, [9, 67] showed that for any $p > 0$, the L_p-heavy hitters problem requires $\Omega\left(\frac{1}{\varepsilon^p} \log n\right)$ words of space, while [12, 52, 64] showed that for $p > 2$ and even for constant $\varepsilon > 0$, the L_p-heavy hitters problem requires $\Omega\left(n^{1-2/p}\right)$ words of space. Therefore, Theorem 1.1 is near-optimal for all $p \geq 1$, for both the number of internal state updates and the memory usage.

Moment estimation. We then consider the F_p moment estimation problem, where the goal is to output an estimate of $F_p(f) = (f_1)^p + \ldots + (f_n)^p$ of a frequency vector $f \in \mathbb{R}^n$ implicitly defined through an insertion-only stream. Our main result is the following:

Theorem 1.3. *Given a constant $p \geq 1$, there exists a one-pass insertion-only streaming algorithm that has $\tilde{O}\left(n^{1-1/p}\right)$ internal state changes, and outputs $\widehat{F_p}$ such that*

$$\mathbf{Pr}\left[\left|\widehat{F_p} - F_p\right| \leq \varepsilon \cdot F_p\right] \geq \frac{2}{3}.$$

For $p \in [1,2)$, the algorithm uses $O\left(\frac{\log^{9+3p}(mn)}{\varepsilon^{4+4p}}\right)$ bits of space, while for $p > 2$, the algorithm uses $\tilde{O}\left(\frac{1}{\varepsilon^{4+2p}} n^{1-2/p}\right)$ space.

We next give a lower bound showing that any approximation algorithm achieving $(2 - \Omega(1))$-approximation to F_p requires $\Omega(n^{1-1/p})$ state updates.

Theorem 1.4. *Let $\varepsilon \in (0,1)$ be a constant and $p \geq 1$. Any algorithm that achieves a $2 - \varepsilon$ approximation to F_p with probability at least $\frac{2}{3}$ requires at least $\frac{1}{2} n^{1-1/p}$ state updates.*

Theorem 1.4 shows that our algorithm in Theorem 1.3 achieves a near-optimal number of internal state changes. Moreover, it is known that any one-pass insertion-only streaming algorithm that achieves $(1+\varepsilon)$-approximation to the F_p moment estimation problem requires $\Omega\left(\frac{1}{\varepsilon^2} + \log n\right)$ bits of space [5, 92] for $p \in [1,2]$ and $\Omega\left(\frac{1}{\varepsilon^2} n^{1-2/p}\right)$ bits of space [94] for $p > 2$, and thus Theorem 1.3 is also near-optimal in terms of space for all $p \geq 1$.

1.3 Technical Overview

Heavy-hitters. We first describe our algorithm for L_p-heavy hitters using near-optimal space and a near-optimal number of internal state changes. For ease of discussion, let us assume that $F_p = \widetilde{\Theta}_\varepsilon(n)$, so that the goal becomes to estimate the frequencies of the coordinates $j \in [n]$ with $f_j \geq \varepsilon \cdot n^{1/p}$, given an input accuracy parameter $\varepsilon \in (0,1)$.

We first define a subroutine SampleAndHold based on sampling a number of items into a reservoir Q. As we observe updates in the stream, we sometimes update the contents of Q and sometimes observe that some stream updates are to coordinates that are being held by the reservoir. For the items that have a large number of stream updates while they are being held by the reservoir, we create separate counters for these items.

We first describe the intuition for $p \geq 2$. We create a reservoir Q of size $\kappa = \widetilde{O}_\varepsilon(n^{1-2/p})$ and sample each item of the stream into the reservoir Q with probability roughly $\frac{1}{\widetilde{\Theta}_\varepsilon(n^{1/p})}$. Note that at some point we may attempt to sample an item of the stream into the reservoir Q when the latter is already full. In this case, we choose a uniformly random item of Q to be replaced by the item corresponding to the stream update.

Our algorithm also checks each stream update to see if it matches an item in the reservoir; if there is a match, we create an explicit counter tracking the frequency of the item. In other words, if $j \in [n]$ arrives as a stream update and $j \in Q$ is in the reservoir, then our algorithm SampleAndHold creates a separate counter for j to count the number of subsequent instances of j.

Now for a heavy hitter $j \in [n]$, we have $f_j \geq \varepsilon \cdot n^{1/p}$ and thus since the sampling probability is $\frac{1}{\widetilde{\Theta}_\varepsilon(n^{1/p})}$, then we can show that j will likely be sampled into our reservoir Q at some point. In fact, since the reservoir Q has size $\kappa = \widetilde{O}_\varepsilon(n^{1-2/p})$, then in expectation, j will be retained by the reservoir for roughly $\widetilde{\Omega}_\varepsilon(n^{1-1/p})$ stream updates before it is possibly replaced by some other item. Moreover, since $f_j \geq \varepsilon \cdot n^{1/p}$, then we should expect another instance of j to arrive in $\widetilde{\Omega}_\varepsilon(n^{1-1/p})$ additional stream updates, where the expectation is taken over the randomness of the sampled positions, under the assumption that $F_p = \Theta(n)$ and thus the stream length is at most $O(n)$. Therefore, our

algorithm will create a counter for tracking j before j is removed from the reservoir Q. Furthermore, we can show that the counter for j is likely created sufficiently early in the stream to provide a $(1+\varepsilon)$-approximation to the frequency f_j of j. Then to decrease the number of internal state updates, we can use Morris counters to approximate the frequency of subsequent updates for each tracked item.

Counter maintenance for heavy-hitters. The main issue with this approach is that too many counters can be created. As a simple example, consider when all items of each coordinate arrive together, one item after the other. Although this is a simple case for counting heavy-hitters, our algorithm will create counters for almost any item that it samples, and although our reservoir uses space $\widetilde{O}_\varepsilon(n^{1-2/p})$, the total number of items sampled over the course of the stream is $\widetilde{O}_\varepsilon(n^{1-1/p})$ and thus the number of created counters can also be $\widetilde{O}_\varepsilon(n^{1-1/p})$, which would be too much space. We thus create an additional mechanism to remove half of the counters each time the number of counters becomes too large, i.e., exceeds $\widetilde{O}_\varepsilon(n^{1-2/p})$. In particular, we remove the counters with the smallest tracked frequencies, which overcomes per-counter analysis in similar algorithms based on sampling [29, 31].

Since our algorithm samples each item of the stream of length $O(n)$ with probability $\frac{1}{\widetilde{O}_\varepsilon(n^{1/p})}$, then we expect our reservoir to have $\widetilde{\Omega}_\varepsilon(n^{1-1/p})$ internal state changes. On the other hand, the counters can increment each time another instance of the tracked item arrives. To that end, we replace each counter with an approximate counter that has a small number of internal state changes. In particular, by using Morris counters, the number of internal state changes for each counter is $\text{poly}\left(\log n, \frac{1}{\varepsilon}, \log\frac{1}{\delta}\right)$ times over the course of the stream. Therefore, the total number of internal state changes is $\widetilde{\Omega}_\varepsilon(n^{1-1/p})$ while the total space used is $\widetilde{O}_\varepsilon(n^{1-2/p})$.

For $p \in [1,2)$, we instead give the reservoir Q a total of $\kappa = \text{poly}_\varepsilon(\log n)$ size, so that the total space is $\text{poly}_\varepsilon(\log n)$ while the total number of internal state changes remains $\widetilde{\Omega}_\varepsilon(n^{1-1/p})$.

Removal of moment assumptions. To remove the assumption that $F_p = \widetilde{O}_\varepsilon(n)$, we note that if each element of the stream of length m is sampled with probability $q < 1$, then the expected number of sampled items is qm, but the p-th power of the expected number of sampled items is $(qm)^p$. Although this is not the p-th moment of the stream, we nevertheless can expect the F_p moment of the stream to decrease at a faster rate than the number of sampled items. Thus we create $L = O(\log(nm))$ substreams so that for each $\ell \in [L]$, we subsample each stream update $[m]$ with probability $\frac{1}{2^{\ell-1}}$. For one of these substreams J_ℓ, we will need have $F_p(J_\ell) = \widetilde{O}_\varepsilon(n)$. We show that we can estimate the frequency of the heavy-hitters in the substream J_ℓ and then rescale by the inverse sampling rate to achieve a $(1+\varepsilon)$-approximation to the frequency of the heavy-hitters in the original stream.

It then remains to identify the correct stream ℓ such that $F_p(J_\ell) = \widetilde{O}_\varepsilon(n)$. A natural approach would be to approximate the moment of each substream, to identify such a correct stream. However, it turns out that our F_p moment estimation algorithm will ultimately use our heavy hitter algorithm as a subroutine. Furthermore, other F_p moment estimation algorithms, e.g., [5, 50, 55, 74], use a number of internal state changes that is linear in the stream length and it is unclear how to adapt these algorithms to decrease the number of internal state changes. Instead, we note that with high probability, the estimated frequency of each heavy-hitter by our algorithm can only be an underestimate. This is because if we initialize counters throughout the stream to track the heavy hitters, then our counters might miss some stream updates to the heavy hitters, but it is not possible to overcount the frequency of each heavy hitter, i.e., we cannot count stream updates that do not exist. Moreover, this statement is still true, up to a $(1+\varepsilon)$ faactor, when we use approximate

counters. Therefore, it suffices to use the maximum estimation for the frequency for each heavy hitter, across all the substreams. We can then use standard probability boosting techniques to simultaneously accurately estimate all L_p-heavy hitters.

Moment estimation. Given our algorithm for finding $(1+\varepsilon)$-approximations to the frequencies of L_p-heavy hitters, we now adapt a standard subsampling framework [58] to reduce the F_p approximation problem to the problem of finding the L_p-heavy hitters. The framework has subsequently been used in a number of different applications, e.g., [16, 30, 33, 73, 77, 93, 94] and has the following intuition.

For ease of discussion, consider the level set Γ_i consisting of the coordinates $j \in [n]$ such that $f_j \in \left(\frac{F_p}{2^i}, \frac{F_p}{2^{i+1}}\right]$ for each i, though we remark that for technical reasons, we shall ultimately define the level sets in a slightly different manner. Because the level sets partition the universe $[n]$, then if we define the contribution $C_i := \sum_{j\in\Gamma_i}(f_k)^p$ of a level set Γ_i to be the sum of the contributions of all their coordinates, then we can decompose the moment F_p into the sum of the contributions of the level sets, $F_p = \sum_i C_i$. Moreover, it suffices to accurately estimate the contributions of the "significant" level sets, i.e., the level sets whose contribution is at least a poly $\left(\varepsilon, \frac{1}{\log(nm)}\right)$ fraction of the F_p moment, and crudely estimate the contributions of the insignificant level sets.

[58] observed that the contributions of the significant level sets can be estimated by approximating the frequencies of the heavy hitters for substreams induced by subsampling the universe at exponentially smaller rates. We emphasize that whereas we previously subsampled updates of the stream $[m]$ for heavy hitters, we now subsample elements of the universe $[n]$. That is, we create $L = O(\log(nm))$ substreams so that for each $\ell \in [L]$, we subsample each element of the universe $[n]$ into the substream with probability $\frac{1}{2^{\ell-1}}$. For example, a single item with frequency $F_p^{1/p}$ will be a heavy hitter in the original stream, which is also the stream induced by $\ell = 1$. On the other hand, if there are n items with frequency $(F_p/n)^{1/p}$, then they will be L_p-heavy hitters at a subsampling level where in expectation, there are roughly $\Theta\left(\frac{1}{\varepsilon^p}\right)$ coordinates of the universe that survive the subsampling. Then [58] notes that $(1+\varepsilon)$-approximations to the contributions of the surviving heavy-hitters can then be rescaled inversely by the sampling rate to obtain good approximations of the contributions of each significant level set. The same procedure also achieves crude approximations for the contributions of insignificant level sets, which overall suffices for a $(1+\varepsilon)$-approximation to the F_p moment.

The key advantage in adapting this framework over other F_p estimation algorithms, e.g., [5, 50, 55, 74] is that we can then use our heavy hitter algorithm FullSampleAndHold to provide $(1+\varepsilon)$-approximations to the heavy hitters in each substream while guaranteeing a small number of internal state changes.

1.4 Preliminaries

We use $[n]$ to denote the set $\{1, 2, \ldots, n\}$ for an integer $n > 0$. We write $\text{poly}(n)$ to denote a fixed univariate polynomial in n and similarly, $\text{poly}(n_1, \ldots, n_k)$ to denote a fixed multivariate polynomial in $n_1, \ldots, n_k$. We use $\tilde{O}(f(n_1, \ldots, n_k))$ for a function $f(n_1, \ldots, n_k)$ to denote $f(n_1, \ldots, n_k) \cdot \text{poly}(\log f(n_1, \ldots, n_k))$. For a vector $f \in \mathbb{R}^n$, we use f_i for $i \in [n]$ to denote the i-th coordinate of f.

Model. In our setting, an insertion-only stream S consists of a sequence of updates $u_1, \ldots, u_m$. In general, we do not require m to be known in advance, though in some cases, we achieve algorithmic improvements when a constant-factor upper bound on m is known in advance; we explicitly clarify the setting in these cases. For each $t \in [m]$, we have $u_t \in [n]$, where without loss of generality, we

use $[n]$ to represent an upper bound on the universe, which we assume to be known in advance. The stream S defines a frequency vector $f \in \mathbb{R}^n$ by $f_i = |\{t \mid u_t = i\}|$, so that for each $i \in [n]$, the i-th value of the frequency vector S is how often i appears in the data stream S. Observe that through a simple reduction, this model suffices to capture the more general case where some coordinate is increased by some amount in $\{1, \dots, M\}$ for some integer $M > 0$ in each update.

For a fixed algorithm $\mathcal{A}$, suppose that at each time $t \in [m]$, the algorithm $\mathcal{A}$ maintains a memory state σ_t. Let $|\sigma_t|$ denote the size of the memory state, in words of space, where each word of space is assumed to used $O(\log n + \log m)$ bits of space. We say the algorithm $\mathcal{A}$ uses s words of space or equivalently, $O(s \log n + \log m)$ bits of space, if $\max_{t\in[m]} |\sigma_t| \le s$. For each $t \in [m]$, let X_t be the indicator random variable for whether the algorithm $\mathcal{A}$ changed its memory state at time t. That is, $X_t = 1$ if $\sigma_t \neq \sigma_{t-1}$ and $X_t = 0$ if $\sigma_t = \sigma_{t-1}$, where we use the convention $\sigma_0 = \emptyset$. Then we say the total number of internal changes by the algorithm $\mathcal{A}$ is $\sum_{t=1}^m x_t$.

We also require the following definition of Morris counters to provide approximate counting.

Theorem 1.5 (Morris counters). *[82, 85] There exists an insertion-only streaming algorithm (Morris counter) that uses space (in bits) $O\left(\log\log n + \log\frac{1}{\varepsilon} + \log\log\frac{1}{\delta}\right)$ and outputs a $(1+\varepsilon)$-approximation to the frequency of an item i, with probability at least $1-\delta$. Moreover, the algorithm is updated at most* $\mathrm{poly}\left(\log n, \frac{1}{\varepsilon}, \log\frac{1}{\delta}\right)$ *times over the course of the stream.*

2 HEAVY HITTERS

In this section, we first describe our algorithm for identifying and accurately approximating the frequencies of the L_p-heavy hitters.

2.1 Sample and Hold

A crucial subroutine for our F_p estimation algorithm is the accurate estimation of heavy hitters. In this section, we first describe such a subroutine SampleAndHold for approximating the frequencies of the L_p-heavy hitters under the assumption that F_p is not too large.

We now describe our algorithm for the case $p \ge 2$. Our algorithm creates a reservoir Q of size $\kappa = \widetilde{O}_\varepsilon(n^{1-2/p})$ and samples each item of the stream into the reservoir Q with probability roughly $\frac{1}{\widetilde{\Theta}_\varepsilon(n^{1/p})}$. If the reservoir Q is full when an item of the stream is sampled, then a uniformly random item of Q is replaced with the stream update. Thus if the stream has length n, then we will incur $\widetilde{O}_\varepsilon(n^{1-2/p})$ internal state changes due to the sampling. For stream length m, we set the sampling probability to be roughly $\frac{\widetilde{O}_\varepsilon(n^{1-1/p})}{m}$.

Our algorithm also checks each stream update to see if it matches an item in the reservoir and creates a counter for the item if there is a match. In other words, if $j \in [n]$ arrives as a stream update and $j \in Q$ is in the reservoir, then our algorithm SampleAndHold creates a separate counter for j to count the number of subsequent instances of j. In addition, we remove half of the counters each time the number of counters becomes too large, i.e., exceeds $\widetilde{O}_\varepsilon(n^{1-2/p})$. In particular, we remove the counters with the smallest tracked frequencies. To reduce the number of internal state changes, we use Morris counters rather than exact counters for each item.

For $p \in [1, 2)$, we set $\kappa = \mathrm{poly}_\varepsilon(\log n)$ to be the size of the reservoir Q, so that the total space is $\mathrm{poly}_\varepsilon(\log n)$ while the total number of internal state changes remains $\widetilde{\Omega}_\varepsilon(n^{1-1/p})$. Our algorithm SampleAndHold appears in Algorithm 1.

We first analyze how many additional counters are available at a time when a heavy hitter j is sampled.

Algorithm 1 SampleAndHold

Input: Stream $s_1, \ldots, s_m$ of items from $[n]$, accuracy parameter $\varepsilon \in (0,1)$, $p \geq 1$
Output: Accurate estimation of an L_p heavy hitter frequency

$\kappa_1 \leftarrow \Theta\left(\frac{\log^{11+3p}(mn)}{\varepsilon^{4+4p}}\right)$, $\gamma \leftarrow 2^{20}p$
if $m \geq n$ **then**
    $\varrho \leftarrow \frac{\gamma^2 n^{1-1/p}\log^4(nm)}{\varepsilon^2 m}$, $\kappa_2 \leftarrow \Theta\left(\frac{n^{1-2/p}\log^{11+3p}(mn)}{\varepsilon^{4+4p}}\right)$
else if $m < n$ **then**
    $\varrho \leftarrow \frac{\gamma^2 m^{1-1/p}\log^4(nm)}{\varepsilon^2 m}$, $\kappa_2 \leftarrow \Theta\left(\frac{m^{1-2/p}\log^{11+3p}(mn)}{\varepsilon^{4+4p}}\right)$
$\kappa \leftarrow \kappa_1$ if $p \in [1,2)$, $\kappa \leftarrow \kappa_2$ if $p \geq 2$
$k \sim \mathrm{Uni}([200p\kappa\log^2(nm), 202p\kappa\log^2(nm)])$
for $t = 1$ to $t = m$ **do**
    $q_i \leftarrow \emptyset$ for $i \in [k]$
    if there is a Morris counter for s_t **then**
        Update the Morris counter
    else if there exists $i \in [k]$ with $q_i = s_t$ **then** ▷item is in the reservoir
        Start a Morris counter for s_t ▷hold a counter for the item
    else
        Pick $\mu_t \in [0,1]$ uniformly at random
        if $\mu_t < \varrho$ **then** ▷with probability ϱ
            Pick $i \in [k]$ uniformly at random
            $q_i \leftarrow s_t$
    if there exist k active Morris counters initialized between time $t - 2^z$ and $t - 2^{z+1}$ for integer $z > 0$ **then** ▷too many counters
        $k \leftarrow \mathrm{Uni}([200p\kappa\log^2(nm), 202p\kappa\log^2(nm)])$
        Retain the $\frac{k}{2}$ counters initialized between time $t - 2^z$ and $t - 2^{z+1}$ for integer $z > 0$ with largest approximate frequency
return the estimated frequencies by the Morris counters

Lemma 2.1. *Let $j \in [n]$ be an item with $(f_j)^p \geq \frac{\varepsilon^2 \cdot F_p}{2^{10}\gamma\log^2(nm)}$ and let $J = \{t \in [m] \mid s_t = j\}$. Let $k \in [200p\kappa\log^2(nm), 202p\kappa\log^2(nm)]$ be chosen uniformly at random. Let v be the last time that j is sampled by the algorithm. Then with probability at least $1 - \frac{1}{50p\log^2(nm)}$, there are at most $k - \kappa$ counters at time v.*

Proof. Consider any fixing of the random samples of the algorithm and the random choices of k, before time v. Let $T_1 < T_2 < \ldots$ be the sequence of times when the counters are reset. Note that since $k \in [200p\kappa\log^2(nm), 202p\kappa\log^2(nm)]$, then each time the counters are reset, between $[100p\kappa^2(nm), 101p\kappa\log^2(nm)]$ counters are newly allocated. Note that the sequence could be empty, in which case our claim is vacuously true. For each T_i, consider the times $\mathcal{T}_i$ at which additional counters would be created after T_i is fixed if there were no limit to the number of counters. Moreover, let u_i be the choice of k at time T_i. Note that the first $100p\kappa\log^2(nm)$ of the times in $\mathcal{T}_i$ are independent of the choice of u_i, while latter times in $\mathcal{T}_i$ may not actually be sampled due to the choice of u_i. Let T_w be the first time for which which v appears in the first $101p\kappa\log^2(nm)$ terms of the $\mathcal{T}_w$. Then with probability at most $\frac{1}{100p\log^2(mn)}$, the choice of u_w will be within κ indices after v in the sequence $\mathcal{T}_i$. On the other hand, the choice of u_w could cause T_{w+1} to be before v, e.g., if v is the $(101p\kappa\log^2(nm))$-th term and $u_w = 100\kappa\log^2(nm)$, in which case the same argument shows

that with probability at most $\frac{1}{100p\log^2(mn)}$, the choice of u_{w+1} will be within κ indices after v in the sequence $\mathcal{T}_{w+1}$. Note that since $u_w + u_{w+1} \geq 200p\kappa^2(nm)$, then v must appear before T_{w+2}. Therefore by a union bound, with probability at least $1 - \frac{1}{50p\log^2(nm)}$, there are at most $k - \kappa$ counters at time v. □

Accurate estimation of the heavy-hitter frequencies can be used in a number of applications such as moment estimation, L_p sampling [6, 61, 63], cascaded norms [65, 66], etc. We next upper bound how many additional counters are created between the time a heavy hitter j is sampled until it becomes too large to delete.

Lemma 2.2. *Suppose $m \geq n$ and $F_p = O\left(\frac{n\log^3(nm)}{\varepsilon^{4p}}\right)$. Let $j \in [n]$ be an item with $(f_j)^p \geq \frac{\varepsilon^2 \cdot F_p}{2^{10}\gamma\log^2(nm)}$. Let $J = \{t \in [m] \mid s_t = j\}$ and let $u \in J$ be chosen uniformly at random and suppose that the algorithm samples j at time u. Then for $p \in [1, 2]$, over the choice of u and the internal randomness of the algorithm, the probability that fewer than $\kappa_1 = O\left(\frac{\log^{11+3p}(mn)}{\varepsilon^{4+4p}}\right)$ new counters are generated after u and before $\frac{\varepsilon^4 \cdot F_p}{2^{10}\gamma\log^2(nm)}$ additional instances of j arrive is at most $1 - \frac{1}{100p\log(nm)}$.*

Similarly for $p \in [1, 2)$, over the choice of u and the internal randomness of the algorithm, the probability that fewer than $\kappa_2 = O\left(\frac{n^{1-2/p}\log^{11+3p}(mn)}{\varepsilon^{4+4p}}\right)$ new counters are generated after u and before $\frac{\varepsilon^4 \cdot F_p}{2^{10}\gamma\log^2(nm)}$ additional instances of j arrive is at most $1 - \frac{1}{100p\log(nm)}$.

Proof. Let $L = O(p\log(nm))$ and let $\ell \in [L]$ be fixed. Let W_ℓ be the items with frequency in $[2^{\ell-1}, 2^\ell)$ so that

$$W_\ell = \{i \in [n] \mid f_i \in [2^{\ell-1}, 2^\ell)\}.$$

and observe that $|W_\ell| \leq \frac{F_p}{2^{p\ell}}$. Let $\varrho = \frac{\gamma^2 n^{1-1/p}\log^4(nm)}{\varepsilon^2 m}$ and $X = \frac{\varepsilon^2 F_p^{1/p}}{2^{14}\gamma^2\log^4(nm)}$.

We define $B_1, \ldots, B_\alpha$ to be blocks that partition the stream, so that the i-th block includes the items of the stream after the $(i-1)$-th instance of f_j, up to and including the i-th instance of f_j. We therefore have $\alpha = f_j + 1$.

Observe that since $(f_j)^p \geq \frac{\varepsilon^2 \cdot F_p}{2^{10}\gamma\log^2(nm)}$, then we have $f_j = \beta X$ for some $\beta > 1$. Since $|W_\ell| \leq \frac{F_p}{2^{p\ell}}$, then the expected number of unique items in W_ℓ contained in a block is at most $\frac{F_p}{2^{p\ell}\alpha}$, conditioned on any fixing of indices that are sampled. Thus in a block, the conditional expectation of the number of stream updates that correspond to items in W_ℓ is at most $\frac{F_p}{2^{(p-1)\ell}\alpha}$, and so the expected number of items in W_ℓ that are sampled in a block is at most $\frac{\varrho F_p}{2^{(p-1)\ell}\alpha}$. Therefore, the expected number of retained items in the previous and following 2^i blocks for $i \leq \ell$ is at most

$$\frac{\varrho F_p}{2^{(p-1)\ell}\alpha} \cdot \min(2^i, 2^\ell) \leq \frac{\varrho F_p}{2^{(p-2)\ell}\alpha} \leq \frac{2\varrho F_p}{2^{(p-2)\ell}X},$$

since $\alpha = f_j + 1$ and by assumption $(f_j)^p \geq \frac{\varepsilon^2 \cdot F_p}{2^{10}\gamma\log^2(nm)}$, but $X = \frac{\varepsilon^2 F_p^{1/p}}{2^{14}\gamma^2\log^4(nm)}$. For $i > \ell$, note that since W_ℓ only contains elements of frequency 2^ℓ, then no elements of W_ℓ will be retained over j once the consideration is for $2^i > 2^\ell$ blocks.

For $p \in [1, 2)$, note that $2^\ell \leq F_p^{1/p}$, so that $\frac{1}{2^{(p-2)\ell}} = 2^{(2-p)\ell} \leq F_p^{2/p-1}$. Therefore, after 2^i blocks, the expected number of items in W_ℓ is at most $\frac{2\varrho F_p}{2^{(p-2)\ell}X} \leq \frac{2\varrho F_p^{2/p}}{X}$. For $\varrho = \frac{\gamma^2 n^{1-1/p}\log^4(nm)}{\varepsilon^2 m}$ and $X = \frac{\varepsilon^2 F_p^{1/p}}{2^{14}\gamma^2\log^4(nm)}$, we have that the expected number of retained items in the previous and following

2^i blocks is at most

$$O\left(\frac{2\gamma^4 n^{1-1/p} F_p^{1/p} \log^8(nm)}{\varepsilon^4 m}\right) = O\left(\frac{\log^{8+3p}(nm)}{\varepsilon^{4+4p}}\right),$$

for $m \geq n$ and $F_p = O\left(\frac{n\log^{3p}(nm)}{\varepsilon^{4p}}\right)$. By Markov's inequality, we have that with probability $1 - \frac{1}{100p\log^3(nm)}$ over a random time u, the number of counters for the items with frequency in $[2^{\ell-1}, 2^\ell)$ is at most $\kappa_1 = O\left(\frac{\log^{11+3p}(nm)}{\varepsilon^{4+4p}}\right)$ across 2^i blocks before and after u. Thus, by a union bound over all $i = O(\log m)$ and $L = O(p\log(nm))$ choices of ℓ, we have that with probability at least $1 - \frac{1}{100p\log(nm)}$, κ_1 new counters are generated after u.

For $p \geq 2$, we have that the expected number of retained items across 2^i blocks is at most $\frac{2\varrho F_p}{2^{(p-2)\ell}X} \leq \frac{2\varrho F_p}{X}$. For $\varrho = \frac{\gamma^2 n^{1-1/p}\log^4(nm)}{\varepsilon^2 m}$ and $X = \frac{\varepsilon^2 F_p^{1/p}}{2^{14}\gamma^2\log^4(nm)}$, we have that the expected number of retained items across 2^i blocks is at most

$$O\left(\frac{2\gamma^4 (nF_p)^{1-1/p}\log^8(nm)}{\varepsilon^4 m}\right) = O\left(\frac{n^{1-2/p}\log^{8+3p}(mn)}{\varepsilon^{4+4p}}\right),$$

for $m \geq n$ and $F_p = O\left(\frac{n\log^{3p}(nm)}{\varepsilon^{4p}}\right)$. By Markov's inequality, we have that with probability $1 - \frac{1}{100p\log^3(nm)}$ over a random time u, the number of counters for the items with frequency in $[2^{\ell-1}, 2^\ell)$ is at most $\kappa_2 = O\left(\frac{n^{1-2/p}\log^{11+3p}(mn)}{\varepsilon^{4+4p}}\right)$ across 2^i blocks after u. Moreover, observe that the number of counters generated within 2^z timesteps is certainly at most the number of counters generated within 2^z blocks. Thus, by a union bound over all $z = O(\log m)$ and $L = O(p\log(nm))$ choices of ℓ, we have that with probability at least $1 - \frac{1}{100p\log(nm)}$, κ_2 new counters are generated after u. □

By Lemma 2.1 and Lemma 2.2, we have:

Lemma 2.3. *Suppose $m \geq n$ and $F_p = O\left(\frac{n\log^3(nm)}{\varepsilon^{4p}}\right)$. Let $j \in [n]$ be an item with $(f_j)^p \geq \frac{\varepsilon^2 \cdot F_p}{2^{10}\gamma\log^2(nm)}$. Then with probability at least $1 - \frac{1}{100p\log(nm)}$, the counters are not reset between the times at which j is sampled and $\frac{\varepsilon^4 \cdot F_p}{2^{10}\gamma\log^2(nm)}$ occurrences of j arrive after it is sampled.*

We now claim that a heavy hitter j will be sampled early enough to obtain a good approximation to its overall frequency.

Lemma 2.4. *Suppose $F_p = O\left(\frac{n\log^3(nm)}{\varepsilon^{4p}}\right)$. Let $j \in [n]$ be an item with $(f_j)^p \geq \frac{\varepsilon^2 \cdot F_p}{2^{10}\gamma\log^2(nm)}$. Then with probability at least 0.99, SampleAndHold outputs $\widehat{f_j}$ such that*

$$\left(1 - \frac{\varepsilon}{8\log(nm)}\right) \cdot (f_j)^p \leq (\widetilde{f_j})^p \leq \left(1 + \frac{\varepsilon}{8\log(nm)}\right) \cdot (f_j)^p.$$

Proof. Note that for the purposes of analysis, we can assume $m \geq n$, since otherwise if $m < n$, then SampleAndHold essentially redefines n to be the number of unique items in the induced stream by setting ϱ and κ appropriately, even though the overall universe can be larger. Note that $m \leq 2\left(F_p\right)^{1/p} \cdot n^{1-1/p}$.

By assumption, we have $(f_j)^p \geq \frac{\varepsilon^2 \cdot F_p}{2^7 \gamma \log^2(nm)}$ and thus, we certainly have $f_j \geq \frac{\varepsilon \cdot (F_p)^{1/p}}{2^7 \gamma \log^2(nm)}$ for $p \geq 2$ and $\varepsilon \in (0,1)$. Let T be the set of the first $\frac{\varepsilon}{16\log(nm)}$ fraction of occurrences of j, so that

$$|T| \geq \frac{\varepsilon^2 \cdot (F_p)^{1/p}}{2^{11}\gamma \log^3(nm)}.$$

We claim that with probability at least $\frac{2}{3}$, a Morris counter for j will be created by the stream as it passes through T. Indeed, observe that since each item of the stream is sampled with probability

$$\varrho = \frac{\gamma^2 n^{1-1/p} \log^4(nm)}{\varepsilon^2 m} \geq \frac{\gamma^2 \log^4(nm)}{2\varepsilon^2 (F_p)^{1/p}},$$

then we have with high probability, an index in T is sampled. By Lemma 2.3, the index will not be removed by the counters resetting.

Since T is the set of the first $\frac{\varepsilon}{16\log(nm)}$ fraction of occurrences of j, then the Morris counter is used for at least $\left(1 - \frac{\varepsilon}{16\log(nm)}\right) f_j$ occurrences of j. We use Morris counters with multiplicative accuracy $\left(1 + O\left(\frac{\varepsilon}{\log(nm)}\right)\right)$. Hence by Theorem 1.5, we obtain an output $\widehat{f_j}$ such that

$$\left(1 - \frac{\varepsilon}{8\log(nm)}\right) \cdot (f_j)^p \leq (\widetilde{f_j})^p \leq \left(1 + \frac{\varepsilon}{8\log(nm)}\right) \cdot (f_j)^p.$$

□

2.2 Full Sample and Hold

We now address certain shortcomings of Lemma 2.4 – namely, the assumption that $F_p = O\left(\frac{n\log^3(nm)}{\varepsilon^{4p}}\right)$ and the fact that Lemma 2.4 only provides constant success probability for each heavy hitter $j \in [n]$, but there can be many of these heavy-hitters.

Algorithm 2 FullSampleAndHold

Input: Stream $s_1, \ldots, s_m$ of items from $[n]$, accuracy parameter $\varepsilon \in (0,1)$, $p \geq 2$
Output: Accurate estimations of L_p heavy hitter frequencies

1: $R \leftarrow O(\log n)$, $Y = O(\log m)$
2: **for** $r \in [R], x \in [Y]$ **do**
3: Let $J_x^{(r)}$ be a (nested) subset of $[m]$ subsampled at rate $p_x := \min(1, 2^{1-x})$
4: Let $m_x^{(r)}$ be the length of the stream $J_x^{(r)}$
5: Run $\text{SampleAndHold}_x^{(r)}$ on $J_x^{(r)}$
6: Let $\widehat{f_j^{(r,x)}}$ be the estimated frequency for j by $\text{SampleAndHold}_x^{(r)}$
7: $\widehat{f_j^x} \leftarrow \text{median}_{r\in[R]} \widehat{f_j^{(r,x)}}$
8: Let $\ell = \min\{x \in [X] \mid m_x \geq (\widehat{f_j^x})^p\}$
9: **return** $\widehat{f_j^\ell}$

We first show subsampling allows us to find a substream that satisfies the required assumptions. We can then boost the probability of success for estimating the frequency of each heavy hitter using a standard median-of-means argument.

Lemma 2.5. *Let $j \in [n]$ be an item with $(f_j)^p \geq \frac{\varepsilon^2 \cdot F_p}{2^{10}\gamma \log^2(nm)}$. Then with probability $1 - \frac{1}{\text{poly}(n)}$, FullSampleAndHold outputs $\widehat{f_j}$ such that*

$$\left(1 - \frac{\varepsilon}{8\log(nm)}\right) \cdot (f_j)^p \leq (\widetilde{f_j})^p \leq \left(1 + \frac{\varepsilon}{8\log(nm)}\right) \cdot (f_j)^p.$$

Proof. Consider a fixed $j \in [n]$ with $(f_j)^p \geq \frac{\varepsilon^2 \cdot F_p}{2^{10}\gamma \log^2(nm)}$. Let $q > 0$ be the integer such that $\frac{f_j}{2^q} \in \left[\frac{400}{\varepsilon^2}, \frac{800}{\varepsilon^2}\right]$. Let A_j be the random variable denoting the number of occurrences of j in J_q and note that over the randomness of the sampling, we have $\mathbb{E}\left[A_j\right] = \frac{f_j}{2^q} \leq \frac{400}{\varepsilon^2}$. We also have $\mathbb{E}\left[A_j^2\right] \leq \frac{f_j}{2^q} + \frac{(f_j)^2}{2^{2q}}$, so that $\mathbb{V}\left[A_j\right] \leq \frac{400}{\varepsilon^2}$. By Chebyshev's inequality, we have that the number of occurrences of j in $J_q^{(r)}$ is a $(1+\varepsilon)$-approximation of $\frac{f_j}{2^q}$ with probability at least 0.9. We similarly have that with probability at least 0.9, $m_x \in \left[\frac{(1-\varepsilon)m}{2^q}, \frac{(1+\varepsilon)m}{2^q}\right]$ and thus $(A_j)^p \leq \frac{800^p}{\varepsilon^{2p}} \cdot m_q$.

Since $(f_j)^p \geq \frac{\varepsilon^2 \cdot F_p}{2^{10}\gamma \log^2(nm)}$, then by a Chernoff bound, we have that for any $y \in [n]$, the number of occurrences of y in $J_q^{(r)}$ is at most $\frac{\xi \log^3(nm)}{\varepsilon^4}$ for some constant $\xi > 1$, with high probability. Thus by a union bound, we have that with high probability,

$$F_p(J_q^{(r)}) \leq \frac{\xi' n \log^{3p}(nm)}{\varepsilon^{4p}},$$

for some sufficiently large constant $\xi' > 1$, which satisfies the assumptions of Lemma 2.4. Hence we have with probability at least 0.99, $\widehat{f_j^{(r,q)}}$ is a $(1+\varepsilon)$-approximation to the number of occurrences of j in $J_q^{(r)}$. Thus, $\widehat{f_j^{(r,q)}} \leq \frac{2 \cdot 800^p}{\varepsilon^{2p}} \cdot m_q$ with probability at least 0.7. Observe that since $(f_j)^p \geq \frac{\varepsilon^2 \cdot F_p}{2^{10}\gamma \log^2(nm)}$, then for any $y \in [n]$, we have that the expected number of occurrences of y in $J_q^{(r)}$ is at most $\frac{\xi \log^2(nm)}{\varepsilon^4}$, for some sufficiently large constant $\xi > 1$. Hence by standard Chernoff bounds, we have that the median satisfies

$$\widehat{f_j^{(q)}} = \underset{r \in [R]}{\text{median}}\, \widehat{f_j^{(r,q)}} \leq \frac{2 \cdot 800^p}{\varepsilon^{2p}} \cdot m_q,$$

with high probability.

Moreover, observe that (1) for any stream with subsampling rate $\frac{1}{2^x} > \frac{1}{2^q}$, we similarly have that the number of occurrences of j in J_x is a $(1+\varepsilon)$-approximation of $\frac{f_j}{2^x}$ with high probability and (2) SampleAndHold cannot overestimate the frequency of j. Thus, $\widehat{f_j^{(\ell)}}$ output by FullSampleAndHold satisfies

$$\left(1 - \frac{\varepsilon}{8\log(nm)}\right) \cdot (f_j)^p \leq (\widetilde{f_j})^p \leq \left(1 + \frac{\varepsilon}{8\log(nm)}\right) \cdot (f_j)^p$$

with high probability. □

3 F_p ESTIMATION

In this section, we present insertion-only streaming algorithms for F_p estimation with a small number of internal state changes. We first observe that an F_p estimation algorithm by [62] achieves a small number of internal state changes for $p < 1$. We then build upon our L_p-heavy hitter algorithm to achieve an F_p estimation algorithm that achieves a small number of internal state changes for $p > 1$.

3.1 F_p Estimation, $p < 1$

As a warm-up, we first show that the F_p estimation streaming algorithm of [62] uses a small number of internal state changes for $p < 1$. We first recall the following definition of the p-stable distribution:

Definition 3.1 (p-stable distribution). *[100] For $0 < p \leq 2$, the p-stable distribution $\mathcal{D}_p$ exists and satisfies $\sum_{i=1}^n Z_i x_i \sim \|x\|_p \cdot Z$ for $Z, Z_1, \ldots, Z_n \sim \mathcal{D}_p$ and any vector $x \in \mathbb{R}^n$.*

A standard method [86] for generating p-stable random variables is to first generate $\theta \sim \text{Uni}\left(\left[-\frac{\pi}{2}, \frac{\pi}{2}\right]\right)$ and $r \sim \text{Uni}([0,1])$ and then set

$$X = f(r,\theta) = \frac{\sin(p\theta)}{\cos^{1/p}(\theta)} \cdot \left(\frac{\cos(\theta(1-p))}{\log \frac{1}{r}}\right)^{\frac{1}{p}-1}.$$

The F_p estimation streaming algorithm of [62] first generates a sketch matrix $D \in \mathbb{R}^{k \times n}$, where $k = O\left(\frac{1}{\varepsilon^2}\right)$ and each entry of D is generated from the p-stable distribution. Observe that D can be viewed as k vectors $D^{(1)}, \ldots, D^{(k)} \in \mathbb{R}^n$ of p-stable random variables. For $i \in [k]$, suppose we maintained $\langle D^{(i)}, x\rangle$, where x is the frequency vector induced by the stream. Then it is known [55] that with constant probability, the median of these inner products is a $(1+\varepsilon)$-approximation to F_p.

[62] notes that each vector $D^{(i)}$ can be further decomposed into a vector $D^{(i,+)}$ containing the positive entries of $D^{(i)}$ and a vector $D^{(i,-)}$ containing the negative entries of $D^{(i)}$. Since $D^{(i)} = D^{(i,+)} + D^{(i,-)}$, then it suffices to maintain $\langle D^{(i,+)}, x\rangle$ and $\langle D^{(i,-)}, x\rangle$ for each $i \in [k]$. For insertion-only streams, all entries of x are non-negative, and so the inner products $\langle D^{(i,+)}, x\rangle$ and $\langle D^{(i,-)}, x\rangle$ are both monotonic over the course of the stream, which permits the application of Morris counters. Thus the algorithm of [62] instead uses Morris counters to approximately compute $\langle D^{(i,+)}, x\rangle$ and $\langle D^{(i,-)}, x\rangle$ to within a $(1 + O(\varepsilon))$-multiplicative factor. The key technical point is that [62] shows that

$$\left|\langle D^{(i,-)}, x\rangle\right| + \left|\langle D^{(i,-)}, x\rangle\right| = O\left(\|x\|_p\right)$$

for $p < 1$ and so $(1 + O(\varepsilon))$-multiplicative factor approximations to $\langle D^{(i,+)}, x\rangle$ and $\langle D^{(i,-)}, x\rangle$ are enough to achieve a $(1+\varepsilon)$-approximation to $\langle D^{(i)}, x\rangle$. Now the main gain is that using Morris counters to approximate $\langle D^{(i,+)}, x\rangle$ and $\langle D^{(i,-)}, x\rangle$, not only is the overall space usage improved for the purposes of [62], but also for our purposes, the number of internal state updates is much smaller.

As an additional technical caveat, [62] notes that the sketching matrix D cannot be stored in the allotted memory. Instead, [62] notes that by using the log-cosine estimator [68] instead of the median estimator, the entries of D can be generated using $O\left(\frac{\log(1/\varepsilon)}{\log\log(1/\varepsilon)}\right)$-wise independence, so that storing the randomness used to generate D only requires $O\left(\frac{\log(1/\varepsilon)}{\log\log(1/\varepsilon)} \log n\right)$ bits of space.

Theorem 3.2. *For $p \in (0,1]$, there exists a one-pass insertion-only streaming algorithm that uses $\text{poly}\left(\log n, \frac{1}{\varepsilon}, \log\frac{1}{\delta}\right)$ internal state changes, $O\left(\frac{1}{\varepsilon^2}\left(\log\log n + \log\frac{1}{\varepsilon}\right) + \frac{\log(1/\varepsilon)}{\log\log(1/\varepsilon)} \log n\right)$ bits of space, and outputs a $(1+\varepsilon)$-approximation to F_p with high probability.*

3.2 F_p Estimation, $p > 1$

In this section, we present our F_p approximation algorithm for insertion-only streams that only have $\widetilde{\Omega}_\varepsilon(n^{1-1/p})$ internal state updates for $p > 1$.

We first define the level sets of the F_p moment, as well as the contribution of each level set.

Definition 3.3 (Level sets and contribution). *Let $\widetilde{F_p}$ be the power of two such that $F_p \le \widetilde{F_p} < 2F_p$. Given a uniformly random $\lambda \in \left[\frac{1}{2}, 1\right]$, for each $\ell \in [L]$, we define the level set*

$$\Gamma_\ell := \left\{ i \in [n] \mid f_i \in \left[\frac{\lambda \cdot \widetilde{F_p}}{2^\ell}, \frac{2\lambda \widetilde{F_p}}{2^\ell} \right) \right\}.$$

We define the contribution C_ℓ and the fractional contribution ϕ_ℓ of level set Γ_ℓ to be $C_\ell := \sum_{i \in \Gamma_\ell} (f_i)^p$ and $\phi_\ell := \frac{C_\ell}{F_p}$.

For an accuracy parameter ε and a stream of length m, we say that a level set Γ_ℓ is significant if its fractional contribution ϕ_ℓ is at least $\frac{\varepsilon}{2p \log(nm)}$. Otherwise, we say the level set is insignificant.

Our algorithm follows from the framework introduced by [58] and subsequently used in a number of different applications, e.g., [16, 30, 33, 73, 77, 93, 94] and has the following intuition. We estimate the contributions of the significant level sets by approximating the frequencies of the heavy hitters for substreams induced by subsampling the universe at exponentially smaller rates. Specifically, we create $L = O(\log n)$ substreams where for each $\ell \in [L]$, we subsample each element of the universe $[n]$ into the substream with probability $\frac{1}{2^{\ell-1}}$. We rescale $(1+\varepsilon)$-approximations to the contributions of the surviving heavy hitters by the inverse of the sampling rate to obtain good approximations of the contributions of each significant level set.

To guarantee a small number of internal state changes, we use our heavy hitter algorithm FullSampleAndHold to provide $(1+\varepsilon)$-approximations to the heavy hitters in each substream, thereby obtaining good approximations to the contributions of each significant level set. Our algorithm appears in full in Algorithm 3.

Algorithm 3 F_p approximation algorithm, $p \ge 1$

Input: Stream $s_1, \ldots, s_m$ of items from $[n]$, accuracy parameter $\varepsilon \in (0,1)$, $p \ge 2$
Output: $(1+\varepsilon)$-approximation to F_p

$k = O\left(\frac{\log^{8+3p}(mn)}{\varepsilon^{4+4p}}\right)$ for $p \in [1,2]$, $k = \tilde{O}\left(\frac{n^{1-2/p}}{\varepsilon^4}\right)$ for $p > 2$
$L = O(p \log(nm))$, $R = O(\log \log n)$, $\gamma = 2^{20} p$
for $t = 1$ to $t = m$ **do**
  for $(\ell, r) \in [L] \times [R]$ **do**
    Let $m_\ell^{(r)}$ be a 2-approximation to the length of the induced stream ▷Morris counter
    Let $I_\ell^{(r)}$ be a (nested) subset of $[n]$ subsampled at rate $p_\ell := \min(1, 2^{1-\ell})$
    if $s_t \in I_\ell^{(r)}$ **then**
      Send s_t to FullSampleAndHold$_\ell^{(r)}$
Let $H_i^{(r)}$ be the outputs of the Morris counters at level i
Let $\widetilde{M}$ be the power of two such that $m^p \le \widetilde{M} < 2m^p$
Let $S_i^{(r)}$ be the set of ordered pairs $(j, \widehat{f_j})$ of $H_i^{(r)}$ with $\left(\widehat{f_j}\right)^p \in \left[\frac{\lambda \cdot \widetilde{M}}{2^i}, \frac{2\lambda \cdot \widetilde{M}}{2^i}\right]$
for $i = 1$ to $i = L$ **do**
  $\ell \leftarrow \max\left(1, i - \left\lfloor \log \frac{\gamma^2 \log(nm)}{\varepsilon^2} \right\rfloor\right)$
  $\widehat{C_i} \leftarrow \frac{1}{p_\ell} \text{median}_{r \in [R]} \left(\sum_{(j, \widehat{f_j}) \in S_\ell^{(r)}} \left(\widehat{f_j}\right)^p \right)$
return $\widehat{F_p} = \sum_{\ell \in [L]} \widehat{C_\ell}$

We note the following corollary of Lemma 2.5.

Lemma 3.4. *Let $r \in [R]$ be fixed. Suppose $j \in I_\ell^{(r)}$ and $(f_j)^p \geq \frac{\varepsilon^2 \cdot F_p(I_\ell^{(r)})}{2^7 \gamma \log^2(nm)}$. Then with probability at least $\frac{9}{10}$, $H_\ell^{(r)}$ outputs $\widehat{f}_j$ with*

$$\left(1 - \frac{\varepsilon}{8\log(nm)}\right) \cdot (f_j)^p \leq (\widetilde{f}_j)^p \leq \left(1 + \frac{\varepsilon}{8\log(nm)}\right) \cdot (f_j)^p.$$

Lemma 3.5. *Let $\varepsilon \in (0,1)$, Γ_i be a fixed level set and let $\ell := \max\left(1, i - \left\lfloor \log \frac{\gamma \log(nm)}{\varepsilon^2} \right\rfloor\right)$. For a fixed $r \in [R]$, let $\mathcal{E}_1$ be the event that $|I_\ell^{(r)}| \leq \frac{32n}{2^\ell}$ and let $\mathcal{E}_2$ be the event that $F_p(I_\ell^{(r)}) \leq \frac{32F_p}{2^\ell}$. Then conditioned on $\mathcal{E}_1$ and $\mathcal{E}_2$, for each $j \in \Gamma_i \cap I_\ell^{(r)}$, there exists $(j, \widetilde{f}_j)$ in $S_i^{(r)}$ such that with probability at least $\frac{9}{10}$,*

$$\left(1 - \frac{\varepsilon}{8\log(nm)}\right) \cdot (f_j)^p \leq (\widetilde{f}_j)^p \leq \left(1 + \frac{\varepsilon}{8\log(nm)}\right) \cdot (f_j)^p.$$

We now justify the approximation guarantees of our algorithm.

Lemma 3.6. $\mathbf{Pr}\left[\left|\widehat{F_p} - F_p\right| \leq \varepsilon \cdot F_p\right] \geq \frac{2}{3}$.

Putting things together, we give the full guarantees of our F_p estimation algorithm in Theorem 1.3.

3.3 Entropy Estimation

In this section, we describe how to estimate the entropy of a stream using a small number of internal state changes. Recall that for a frequency vector $f \in \mathbb{R}^n$, the Shannon entropy of f is defined by $H(f) = -\sum_{i=1}^n f_i \log f_i$. Observe that any algorithm that obtains a $(1 + O(\varepsilon))$-multiplicative approximation to the function $h(f) = 2^{H(f)}$ also obtains an $O(\varepsilon)$-additive approximation of the Shannon entropy $H(f)$, and vice versa. Hence to obtain an additive ε-approximation to the Shannon entropy, we describe how to obtain a multiplicative $(1+\varepsilon)$-approximation to $h(f) = 2^{H(f)}$.

Lemma 3.7 ([53])**.** *Given an accuracy parameter $\varepsilon > 0$, let $k = \log \frac{1}{\varepsilon} + \log\log m$ and $\varepsilon' = \frac{\varepsilon}{12(k+1)^3 \log m}$. Then there exists an efficiently computable set $\{p_0, \ldots, p_k\}$ such that $p_i \in (0,2)$ for all i, as well as an efficiently computable deterministic function that uses $(1+\varepsilon')$-approximations to $F_{p_i}(f)$ to compute a $(1+\varepsilon)$-approximation to $h(f) = 2^{H(f)}$.*

Section 3.3 in [53] describes how to compute the set $\{p_0, \ldots, p_k\}$ in Lemma 3.7 as follows. We define $\ell = \frac{1}{2(k+1)\log m}$ and the function $g(z) = \frac{\ell(k^2(z-1)+1))}{2k^2+1}$. For each p_i, we set $p_i = 1 + g(\cos(i\pi/k))$, which can be efficiently computed. Thus, the set $\{p_0, \ldots, p_k\}$ in Lemma 3.7 can be efficiently computed as pre-processing, assuming that n and m are known a priori. Let $P(x)$ be the degree k polynomial interpolated at the points $p_0, \ldots, p_k$, so that $P(p_i) = F_{p_i}(f)$ for each $i \in [k]$, where $F_{p_i}(f)$ is the (p_i)-th moment of the frequency vector f. [53] then showed that a multiplicative $(1 + O(\varepsilon))$-approximation to $h(f) = 2^{H(f)}$ can then be computed from $2^{P(0)}$, and moreover a $(1 + O(\varepsilon))$-approximation to $2^{P(0)}$ can be computed from $(1 + O(\varepsilon))$-approximations to $F_{p_i}(f)$, for each $i \in [k]$.

Thus by Lemma 3.7 and Theorem 1.3, we have:

Theorem 3.8. *Given an accuracy parameter $\varepsilon \in (0,1)$, as well as the universe size n, and the stream length $m = \text{poly}(n)$, there exists a one-pass insertion-only streaming algorithm that has $\tilde{O}\left(\frac{1}{\varepsilon^{O(1)}}\sqrt{n}\right)$ internal state changes, uses $O\left(\frac{\log^{O(1)}(mn)}{\varepsilon^{O(1)}}\right)$ bits of space, and outputs $\widehat{H}$ such that $\mathbf{Pr}\left[\left|\widehat{H} - H\right| \leq \varepsilon\right] \geq \frac{2}{3}$, where H is the Shannon entropy of the stream.*

4 LOWER BOUND

In this section, we describe our lower bound showing that any streaming algorithm achieving a $(2-\Omega(1))$-approximation to F_p requires at least $\frac{1}{2}n^{1-1/p}$ state updates, regardless of the memory allocated to the algorithm. The proof of Theorem 1.2 is similar. The main idea is that we create two streams $\mathcal{S}_1$ and $\mathcal{S}_2$ of length $O(n)$ that look similar everywhere except for a random contiguous block B of $n^{1/p}$. In B, the first stream $\mathcal{S}_1$ has the same item repeated $n^{1/p}$ times, while the second stream $\mathcal{S}_2$ has $n^{1/p}$ distinct items each appear once. The remaining $n-n^{1/p}$ stream updates of $\mathcal{S}_1$ and $\mathcal{S}_2$ are additional distinct items that each appear once, so that $F_p(\mathcal{S}_1) \geq (2-o(1)) \cdot F_p(\mathcal{S}_2)$ and $F_p(\mathcal{S}_2) = \Omega(n)$. Any algorithm $\mathcal{A}$ that achieves a $(2-\Omega(1))$-approximation to F_p must distinguish between $\mathcal{S}_1$ and $\mathcal{S}_2$ and thus $\mathcal{A}$ must perform some action on B. However, B has size $n^{1/p}$ and has random location throughout the stream, so $\mathcal{A}$ must perform $\Omega(n^{1-1/p})$ state updates.

Theorem 1.4. *Let $\varepsilon \in (0,1)$ be a constant and $p \geq 1$. Any algorithm that achieves a $2-\varepsilon$ approximation to F_p with probability at least $\frac{2}{3}$ requires at least $\frac{1}{2}n^{1-1/p}$ state updates.*

Proof. Consider the two following possible input streams. For the stream $\mathcal{S}_1$ of length n on universe n, we choose a random contiguous block B of $n^{1/p}$ stream updates and set them all equal to the same random universe item $i \in [n]$. We randomly choose the remaining $n-n^{1/p}$ updates in the stream so that they are all distinct and none of them are equal to i. Note that by construction, we have $F_p(\mathcal{S}_1) = (n-n^{1/p}) + (n^{1/p})^p = 2n - n^{1/p}$. For the stream $\mathcal{S}_2$ of length n on universe n, we choose $\mathcal{S}_2$ to be a random permutation of $[n]$, so that $F_p(\mathcal{S}_2) = n$.

For fixed $\varepsilon \in (0,1)$, let $\mathcal{A}$ be an algorithm that achieves a $2-\varepsilon$ approximation to F_p with probability at least $\frac{2}{3}$, while using fewer than $\frac{1}{2}n^{1-1/p}$ state updates. We claim that with probability $\frac{1}{2}$, $\mathcal{A}$ must have the same internal state before and after B in the stream $\mathcal{S}_1$. Note that we can view each stream update as (i,j) where $i \in [n]$ is the index of the stream update and $j \in [n]$ is the identity of the universe item. Observe that for a random stream update $i \in [n]$, a random universe update $j \in [n]$ alters the state of $\mathcal{A}$ with probability at most $\frac{1}{2n^{1/p}}$, since otherwise for a random stream, the expected number of state changes would be larger than $\frac{n^{1-1/p}}{2}$, which would be a contradiction. Since both the choice of B and the element $j \in [n]$ that is repeated $n^{1/p}$ times are chosen uniformly at random, then the expected number of changes of the streaming algorithm on the block B is at most $\frac{n^{1/p}}{2n^{1/p}} = \frac{1}{2}$. Therefore, with probability at least $\frac{1}{2}$, the streaming algorithm's state is the same before and after the block B.

Moreover, the same argument applies to $\mathcal{S}_2$, and so therefore with probability at least $\frac{1}{2}$, the streaming algorithm cannot distinguish between $\mathcal{S}_1$ and $\mathcal{S}_2$ if its internal state only changes fewer than $\frac{n^{1-1/p}}{2}$ times. □

ACKNOWLEDGMENTS

We would like to thank Mark Braverman for asking questions related to the ones we studied in this work. D. W. would like to thank Google Research and the Simons Institute for the Theory of Computing, where part of this work was done, as well as a Simons Investigator Award. D.W. and S.Z. are supported in part by NSF CCF-2335411.

REFERENCES

[1] Kook Jin Ahn and Sudipto Guha. 2009. Graph Sparsification in the Semi-streaming Model. In *Automata, Languages and Programming, 36th Internatilonal Colloquium, ICALP Proceedings, Part II*. 328–338.

[2] Miklós Ajtai, Vladimir Braverman, T. S. Jayram, Sandeep Silwal, Alec Sun, David P. Woodruff, and Samson Zhou. 2022. The White-Box Adversarial Data Stream Model. In *Proceedings of the 41st ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS*. 15–27.

[3] Ameen Akel, Adrian M. Caulfield, Todor I. Mollov, Rajesh K. Gupta, and Steven Swanson. 2011. Onyx: A Prototype Phase Change Memory Storage Array. In *3rd USENIX Workshop on Hot Topics in Storage and File Systems, HotStorage1*.

[4] Noga Alon, Phillip B. Gibbons, Yossi Matias, and Mario Szegedy. 2002. Tracking Join and Self-Join Sizes in Limited Storage. *J. Comput. Syst. Sci.* 64, 3 (2002), 719–747.

[5] Noga Alon, Yossi Matias, and Mario Szegedy. 1999. The Space Complexity of Approximating the Frequency Moments. *J. Comput. Syst. Sci.* 58, 1 (1999), 137–147.

[6] Alexandr Andoni, Robert Krauthgamer, and Krzysztof Onak. 2011. Streaming Algorithms via Precision Sampling. In *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS*. 363–372.

[7] Apple. 2023. https://developer.apple.com/documentation/xcode/reducing-disk-writes

[8] Manos Athanassoulis, Bishwaranjan Bhattacharjee, Mustafa Canim, and Kenneth A. Ross. 2012. Path processing using Solid State Storage. In *International Workshop on Accelerating Data Management Systems Using Modern Processor and Storage Architectures - ADMS*. 23–32.

[9] Khanh Do Ba, Piotr Indyk, Eric Price, and David P. Woodruff. 2010. Lower Bounds for Sparse Recovery. In *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*. 1190–1197.

[10] Brian Babcock, Shivnath Babu, Mayur Datar, Rajeev Motwani, and Jennifer Widom. 2002. Models and issues in data stream systems. In *Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*. 1–16.

[11] Mary Baker, John H. Hartman, Michael D. Kupfer, Ken Shirriff, and John K. Ousterhout. 1991. Measurements of a Distributed File System. In *Proceedings of the Thirteenth ACM Symposium on Operating System Principles, SOSP*. 198–212.

[12] Ziv Bar-Yossef, T. S. Jayram, Ravi Kumar, and D. Sivakumar. 2004. An information statistics approach to data stream and communication complexity. *J. Comput. Syst. Sci.* 68, 4 (2004), 702–732.

[13] Avraham Ben-Aroya and Sivan Toledo. 2011. Competitive analysis of flash memory algorithms. *ACM Trans. Algorithms* 7, 2 (2011), 23:1–23:37.

[14] Ran Ben-Basat, Gil Einziger, Roy Friedman, and Yaron Kassner. 2017. Optimal elephant flow detection. In *2017 IEEE Conference on Computer Communications, INFOCOM*. 1–9.

[15] Omri Ben-Eliezer, Rajesh Jayaram, David P. Woodruff, and Eylon Yogev. 2022. A Framework for Adversarially Robust Streaming Algorithms. *J. ACM* 69, 2 (2022), 17:1–17:33.

[16] Jaroslaw Blasiok, Vladimir Braverman, Stephen R. Chestnut, Robert Krauthgamer, and Lin F. Yang. 2017. Streaming symmetric norms via measure concentration. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC*. 716–729.

[17] Jaroslaw Blasiok, Jian Ding, and Jelani Nelson. 2017. Continuous Monitoring of l_p Norms in Data Streams. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM*. 32:1–32:13.

[18] Guy E. Blelloch, Jeremy T. Fineman, Phillip B. Gibbons, Yan Gu, and Julian Shun. 2015. Sorting with Asymmetric Read and Write Costs. In *Proceedings of the 27th ACM on Symposium on Parallelism in Algorithms and Architectures, SPAA*. 1–12.

[19] Jeremiah Blocki, Seunghoon Lee, Tamalika Mukherjee, and Samson Zhou. 2023. Differentially Private L_2-Heavy Hitters Model. In *The Eleventh International Conference on Learning Representations, ICLR*.

[20] Robert S. Boyer and J. Strother Moore. 1991. MJRTY: A Fast Majority Vote Algorithm. In *Automated Reasoning: Essays in Honor of Woody Bledsoe (Automated Reasoning Series)*. 105–118.

[21] Vladimir Braverman, Stephen R. Chestnut, Nikita Ivkin, Jelani Nelson, Zhengyu Wang, and David P. Woodruff. 2017. BPTree: An ℓ_2 Heavy Hitters Algorithm Using Constant Memory. In *Proceedings of the 36th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS*. 361–376.

[22] Vladimir Braverman, Stephen R. Chestnut, Nikita Ivkin, and David P. Woodruff. 2016. Beating CountSketch for heavy hitters in insertion streams. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC*. 740–753.

[23] Vladimir Braverman, Petros Drineas, Cameron Musco, Christopher Musco, Jalaj Upadhyay, David P. Woodruff, and Samson Zhou. 2020. Near Optimal Linear Algebra in the Online and Sliding Window Models. In *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS*. 517–528.

[24] Vladimir Braverman, Dan Feldman, Harry Lang, and Daniela Rus. 2019. Streaming Coreset Constructions for M-Estimators. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM*. 62:1–62:15.

[25] Vladimir Braverman, Dan Feldman, Harry Lang, Daniela Rus, and Adiel Statman. 2023. Least-Mean-Squares Coresets for Infinite Streams. *IEEE Trans. Knowl. Data Eng.* 35, 9 (2023), 8699–8712.

[26] Vladimir Braverman, Dan Feldman, Harry Lang, Adiel Statman, and Samson Zhou. 2021. Efficient Coreset Constructions via Sensitivity Sampling. In *Asian Conference on Machine Learning, ACML*. 948–963.

[27] Vladimir Braverman, Elena Grigorescu, Harry Lang, David P. Woodruff, and Samson Zhou. 2018. Nearly Optimal Distinct Elements and Heavy Hitters on Sliding Windows. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM*. 7:1–7:22.

[28] Vladimir Braverman, Avinatan Hassidim, Yossi Matias, Mariano Schain, Sandeep Silwal, and Samson Zhou. 2021. Adversarial Robustness of Streaming Algorithms through Importance Sampling. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems, NeurIPSl*. 3544–3557.

[29] Vladimir Braverman, Jonathan Katzman, Charles Seidell, and Gregory Vorsanger. 2014. An Optimal Algorithm for Large Frequency Moments Using $O(n^{(1-2/k)})$ Bits. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM*. 531–544.

[30] Vladimir Braverman, Joel Manning, Zhiwei Steven Wu, and Samson Zhou. 2023. Private Data Stream Analysis for Universal Symmetric Norm Estimation. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM*. 45:1–45:24.

[31] Vladimir Braverman and Rafail Ostrovsky. 2013. Approximating Large Frequency Moments with Pick-and-Drop Sampling. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques - 16th International Workshop, APPROX 2013, and 17th International Workshop, RANDOM. Proceedings*. 42–57.

[32] Vladimir Braverman, Emanuele Viola, David P. Woodruff, and Lin F. Yang. 2018. Revisiting Frequency Moment Estimation in Random Order Streams. In *45th International Colloquium on Automata, Languages, and Programming, ICALP*. 25:1–25:14.

[33] Vladimir Braverman, Viska Wei, and Samson Zhou. 2021. Symmetric Norm Estimation and Regression on Sliding Windows. In *Computing and Combinatorics - 27th International Conference, COCOON, Proceedings*. 528–539.

[34] Li-Pin Chang. 2007. On efficient wear leveling for large-scale flash-memory storage systems. In *Proceedings of the 2007 ACM Symposium on Applied Computing (SAC)*. 1126–1130.

[35] Yuan-Hao Chang, Jen-Wei Hsieh, and Tei-Wei Kuo. 2007. Endurance Enhancement of Flash-Memory Storage, Systems: An Efficient Static Wear Leveling Design. In *Proceedings of the 44th Design Automation Conference, DAC*. 212–217.

[36] Moses Charikar, Kevin C. Chen, and Martin Farach-Colton. 2004. Finding frequent items in data streams. *Theor. Comput. Sci.* 312, 1 (2004), 3–15.

[37] Shimin Chen, Phillip B. Gibbons, and Suman Nath. 2011. Rethinking Database Algorithms for Phase Change Memory. In *Fifth Biennial Conference on Innovative Data Systems Research, CIDR*. 21–31.

[38] Tseng-Yi Chen, Tsung Tai Yeh, Hsin-Wen Wei, Yu-Xun Fang, Wei-Kuan Shih, and Tsan-sheng Hsu. 2012. CacheRAID: An Efficient Adaptive Write Cache Policy to Conserve RAID Disk Array Energy. In *IEEE Fifth International Conference on Utility and Cloud Computing, UCC*. 117–124.

[39] Sangyeun Cho and Hyunjin Lee. 2009. Flip-N-Write: a simple deterministic technique to improve PRAM write performance, energy and endurance. In *42st Annual IEEE/ACM International Symposium on Microarchitecture (MICRO-42)*. 347–357.

[40] Michael B. Cohen, Cameron Musco, and Jakub Pachocki. 2020. Online Row Sampling. *Theory Comput.* 16 (2020), 1–25.

[41] Vincent Cohen-Addad, David P. Woodruff, and Samson Zhou. 2023. Streaming Euclidean k-median and k-means with $o(\log n)$ Space. In *64th IEEE Annual Symposium on Foundations of Computer Science, FOCS*.

[42] Graham Cormode, Piotr Indyk, Nick Koudas, and S. Muthukrishnan. 2002. Fast Mining of Massive Tabular Data via Approximate Distance Computations. In *Proceedings of the 18th International Conference on Data Engineering*. 605–614.

[43] Graham Cormode and S. Muthukrishnan. 2005. An improved data stream summary: the count-min sketch and its applications. *J. Algorithms* 55, 1 (2005), 58–75.

[44] Xiangyu Dong, Norman P. Jouppi, and Yuan Xie. 2009. PCRAMsim: System-level performance, energy, and area modeling for Phase-Change RAM. In *2009 International Conference on Computer-Aided Design, ICCAD*. 269–275.

[45] Xiangyu Dong, Xiaoxia Wu, Guangyu Sun, Yuan Xie, Hai Li, and Yiran Chen. 2008. Circuit and microarchitecture evaluation of 3D stacking magnetic RAM (MRAM) as a universal memory replacement. In *Proceedings of the 45th Design Automation Conference, DAC*. 554–559.

[46] David Eppstein, Michael T. Goodrich, Michael Mitzenmacher, and Pawel Pszona. 2014. Wear Minimization for Cuckoo Hashing: How Not to Throw a Lot of Eggs into One Basket. In *Experimental Algorithms - 13th International Symposium,*

SEA. Proceedings. 162–173.

[47] Cristian Estan and George Varghese. 2002. New directions in traffic measurement and accounting. In *Proceedings of the ACM SIGCOMM 2002 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*. 323–336.

[48] Min Fang, Narayanan Shivakumar, Hector Garcia-Molina, Rajeev Motwani, and Jeffrey D. Ullman. 1998. Computing Iceberg Queries Efficiently. In *VLDB'98, Proceedings of 24rd International Conference on Very Large Data Bases*. 299–310.

[49] Joan Feigenbaum, Sampath Kannan, Martin Strauss, and Mahesh Viswanathan. 2002. An Approximate L1-Difference Algorithm for Massive Data Streams. *SIAM J. Comput.* 32, 1 (2002), 131–151.

[50] Sumit Ganguly and David P. Woodruff. 2018. High Probability Frequency Moment Sketches. In *45th International Colloquium on Automata, Languages, and Programming, ICALP*. 58:1–58:15.

[51] Mina Ghashami, Edo Liberty, Jeff M. Phillips, and David P. Woodruff. 2016. Frequent Directions: Simple and Deterministic Matrix Sketching. *SIAM J. Comput.* 45, 5 (2016), 1762–1792.

[52] André Gronemeier. 2009. Asymptotically Optimal Lower Bounds on the NIH-Multi-Party Information Complexity of the AND-Function and Disjointness. In *26th International Symposium on Theoretical Aspects of Computer Science, STACS Proceedings*. 505–516.

[53] Nicholas J. A. Harvey, Jelani Nelson, and Krzysztof Onak. 2008. Sketching and Streaming Entropy via Approximation Theory. In *49th Annual IEEE Symposium on Foundations of Computer Science, FOCS*. 489–498.

[54] Piotr Indyk. 2004. Algorithms for dynamic geometric problems over data streams. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing*. 373–380.

[55] Piotr Indyk. 2006. Stable distributions, pseudorandom generators, embeddings, and data stream computation. *J. ACM* 53, 3 (2006), 307–323.

[56] Piotr Indyk. 2013. Sketching via hashing: from heavy hitters to compressed sensing to sparse fourier transform. In *Proceedings of the 32nd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS*. 87–90.

[57] Piotr Indyk, Shyam Narayanan, and David P. Woodruff. 2022. Frequency Estimation with One-Sided Error. In *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms, SODA*. 695–707.

[58] Piotr Indyk and David P. Woodruff. 2005. Optimal approximations of the frequency moments of data streams. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing*. 202–208.

[59] Sandy Irani, Moni Naor, and Ronitt Rubinfeld. 1992. On the Time and Space Complexity of Computation Using Write-Once Memory Or Is Pen Really Much Worse Than Pencil? *Math. Syst. Theory* 25, 2 (1992), 141–159.

[60] Rajesh Jayaram and David P. Woodruff. 2018. Data Streams with Bounded Deletions. In *Proceedings of the 37th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*. 341–354.

[61] Rajesh Jayaram and David P. Woodruff. 2018. Perfect Lp Sampling in a Data Stream. In *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS*. 544–555.

[62] Rajesh Jayaram and David P. Woodruff. 2019. Towards Optimal Moment Estimation in Streaming and Distributed Models. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2019*. 29:1–29:21.

[63] Rajesh Jayaram, David P. Woodruff, and Samson Zhou. 2022. Truly Perfect Samplers for Data Streams and Sliding Windows. In *PODS '22: International Conference on Management of Data, Philadelphia*. 29–40.

[64] T. S. Jayram. 2009. Hellinger Strikes Back: A Note on the Multi-party Information Complexity of AND. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, 12th International Workshop, APPROX 2009, and 13th International Workshop, RANDOM. Proceedings*.

[65] T. S. Jayram and David P. Woodruff. 2009. The Data Stream Space Complexity of Cascaded Norms. In *50th Annual IEEE Symposium on Foundations of Computer Science, FOCS*. 765–774.

[66] Tanqiu Jiang, Yi Li, Honghao Lin, Yisong Ruan, and David P. Woodruff. 2020. Learning-Augmented Data Stream Algorithms. In *8th International Conference on Learning Representations, ICLR*.

[67] Hossein Jowhari, Mert Saglam, and Gábor Tardos. 2011. Tight bounds for Lp samplers, finding duplicates in streams, and related problems. In *Proceedings of the 30th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS*. 49–58.

[68] Daniel M. Kane, Jelani Nelson, and David P. Woodruff. 2010. On the Exact Space Complexity of Sketching and Streaming Small Norms. In *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*. 1161–1178.

[69] Hyojun Kim, Sangeetha Seshadri, Clement L. Dickey, and Lawrence Chiu. 2014. Evaluating Phase Change Memory for Enterprise Storage Systems: A Study of Caching and Tiering Approaches. *ACM Trans. Storage* 10, 4 (2014), 15:1–15:21.

[70] Kasper Green Larsen, Jelani Nelson, Huy L. Nguyen, and Mikkel Thorup. 2016. Heavy Hitters via Cluster-Preserving Clustering. In *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS*. 61–70.

[71] Christian Janos Lebeda and Jakub Tetek. 2023. Better Differentially Private Approximate Histograms and Heavy Hitters using the Misra-Gries Sketch. In *Proceedings of the 42nd ACM SIGMOD-SIGACT-SIGAI Symposium on Principles*

of Database Systems, PODS. 79–88.

[72] Benjamin C. Lee, Engin Ipek, Onur Mutlu, and Doug Burger. 2009. Architecting phase change memory as a scalable dram alternative. In *36th International Symposium on Computer Architecture (ISCA),*. 2–13.

[73] Roie Levin, Anish Prasad Sevekari, and David P. Woodruff. 2018. Robust Subspace Approximation in a Stream. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems, NeurIPS*.

[74] Ping Li. 2008. Estimators and tail bounds for dimension reduction in l_α $(0 < \alpha \leq 2)$ using stable random projections. In *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, Shang-Hua Teng (Ed.). 10–19.

[75] Yi Li, Vasileios Nakos, and David P. Woodruff. 2018. On Low-Risk Heavy Hitters and Sparse Recovery Schemes. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM*. 19:1–19:13.

[76] Yi Li and David P. Woodruff. 2013. A Tight Lower Bound for High Frequency Moment Estimation with Small Error. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques - 16th International Workshop, APPROX 2013, and 17th International Workshop, RANDOM. Proceedings*. 623–638.

[77] Sepideh Mahabadi, David P. Woodruff, and Samson Zhou. 2022. Adaptive Sketches for Robust Regression with Importance Sampling. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM)*, Amit Chakrabarti and Chaitanya Swamy (Eds.). 31:1–31:21.

[78] Gurmeet Singh Manku and Rajeev Motwani. 2012. Approximate Frequency Counts over Data Streams. *Proc. VLDB Endow.* 5, 12 (2012), 1699.

[79] Jagan Singh Meena, Simon Min Sze, Umesh Chand, and Tseung-Yuen Tseng. 2014. Overview of emerging nonvolatile memory technologies. *Nanoscale research letters* 9 (2014), 1–33.

[80] Ahmed Metwally, Divyakant Agrawal, and Amr El Abbadi. 2005. Efficient Computation of Frequent and Top-k Elements in Data Streams. In *Database Theory - ICDT 2005, 10th International Conference, Proceedings*. 398–412.

[81] Jayadev Misra and David Gries. 1982. Finding Repeated Elements. *Sci. Comput. Program.* 2, 2 (1982), 143–152.

[82] Robert H. Morris Sr. 1978. Counting Large Numbers of Events in Small Registers. *Commun. ACM* 21, 10 (1978), 840–842.

[83] Shanmugavelayutham Muthukrishnan et al. 2005. Data streams: Algorithms and applications. *Foundations and Trends® in Theoretical Computer Science* 1, 2 (2005), 117–236.

[84] Dushyanth Narayanan, Austin Donnelly, and Antony I. T. Rowstron. 2008. Write off-loading: Practical power management for enterprise storage. *ACM Trans. Storage* 4, 3 (2008), 10:1–10:23.

[85] Jelani Nelson and Huacheng Yu. 2022. Optimal Bounds for Approximate Counting. In *PODS '22: International Conference on Management of Data*. ACM, 119–127.

[86] John Nolan. 2003. *Stable distributions: models for heavy-tailed data*. Birkhauser New York.

[87] Moinuddin K. Qureshi, Sudhanva Gurumurthi, and Bipin Rajendran. 2011. *Phase Change Memory: From Devices to Systems*. Morgan & Claypool Publishers.

[88] Eno Thereska, Austin Donnelly, and Dushyanth Narayanan. 2011. Sierra: practical power-proportionality for data center storage. In *European Conference on Computer Systems, Proceedings of the Sixth European conference on Computer systems, EuroSys*. 169–182.

[89] Mikkel Thorup and Yin Zhang. 2004. Tabulation based 4-universal hashing with applications to second moment estimation. In *Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*. 615–624.

[90] Stratis Viglas. 2012. Adapting the B + -tree for Asymmetric I/O. In *Advances in Databases and Information Systems - 16th East European Conference, ADBIS. Proceedings*. 399–412.

[91] Stratis Viglas. 2014. Write-limited sorts and joins for persistent memory. *Proc. VLDB Endow.* 7, 5 (2014), 413–424.

[92] David P. Woodruff. 2004. Optimal space lower bounds for all frequency moments. In *Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*. 167–175.

[93] David P. Woodruff and Qin Zhang. 2012. Tight bounds for distributed functional monitoring. In *Proceedings of the 44th Symposium on Theory of Computing Conference, STOC*. 941–960.

[94] David P. Woodruff and Samson Zhou. 2021. Separations for Estimating Large Frequency Moments on Data Streams. In *48th International Colloquium on Automata, Languages, and Programming, ICALP*. 112:1–112:21.

[95] David P. Woodruff and Samson Zhou. 2021. Tight Bounds for Adversarially Robust Streams and Sliding Windows via Difference Estimators. In *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS*. 1183–1196.

[96] Cong Xu, Xiangyu Dong, Norman P. Jouppi, and Yuan Xie. 2011. Design implications of memristor-based RRAM cross-point structures. In *Design, Automation and Test in Europe, DATE*. 734–739.

[97] Byung-Do Yang, Jae-Eun Lee, Jang-Su Kim, Junghyun Cho, Seung-Yun Lee, and Byoung-Gon Yu. 2007. A Low Power Phase-Change Random Access Memory using a Data-Comparison Write Scheme. In *International Symposium on Circuits and Systems (ISCAS 2007)*. 3014–3017.

[98] Ping Zhou, Bo Zhao, Jun Yang, and Youtao Zhang. 2009. A durable and energy efficient main memory using phase change memory technology. In *36th International Symposium on Computer Architecture (ISCA 2009)*. 14–23.
[99] Qingbo Zhu, Zhifeng Chen, Lin Tan, Yuanyuan Zhou, Kimberly Keeton, and John Wilkes. 2005. Hibernator: helping disk arrays sleep through the winter. In *Proceedings of the 20th ACM Symposium on Operating Systems Principles, SOSP*. 177–190.
[100] Vladimir M. Zolotarev. 1989. One-dimensional stable distributions. *Bull. Amer. Math. Soc* 20 (1989), 270–277.

A MISSING PROOFS FROM SECTION 3

We note the following corollary of Lemma 2.5.

Lemma 3.4. *Let $r \in [R]$ be fixed. Suppose $j \in I_\ell^{(r)}$ and $(f_j)^p \geq \frac{\varepsilon^2 \cdot F_p(I_\ell^{(r)})}{2^7 \gamma \log^2(nm)}$. Then with probability at least $\frac{9}{10}$, $H_\ell^{(r)}$ outputs $\widehat{f_j}$ with*

$$\left(1 - \frac{\varepsilon}{8\log(nm)}\right) \cdot (f_j)^p \leq (\widetilde{f_j})^p \leq \left(1 + \frac{\varepsilon}{8\log(nm)}\right) \cdot (f_j)^p.$$

Proof. The proof follows from Lemma 2.5 and the fact that FullSampleAndHold is only run on the substream induced by $I_\ell^{(r)}$. □

Lemma 3.5. *Let $\varepsilon \in (0,1)$, Γ_i be a fixed level set and let $\ell := \max\left(1, i - \left\lfloor \log \frac{\gamma \log(nm)}{\varepsilon^2} \right\rfloor\right)$. For a fixed $r \in [R]$, let $\mathcal{E}_1$ be the event that $|I_\ell^{(r)}| \leq \frac{32n}{2^\ell}$ and let $\mathcal{E}_2$ be the event that $F_p(I_\ell^{(r)}) \leq \frac{32F_p}{2^\ell}$. Then conditioned on $\mathcal{E}_1$ and $\mathcal{E}_2$, for each $j \in \Gamma_i \cap I_\ell^{(r)}$, there exists $(j, \widetilde{f_j})$ in $S_i^{(r)}$ such that with probability at least $\frac{9}{10}$,*

$$\left(1 - \frac{\varepsilon}{8\log(nm)}\right) \cdot (f_j)^p \leq (\widetilde{f_j})^p \leq \left(1 + \frac{\varepsilon}{8\log(nm)}\right) \cdot (f_j)^p.$$

Proof. We consider casework on whether $i - \left\lfloor \log \frac{\gamma \log^2(nm)}{\varepsilon^2} \right\rfloor \leq 1$ or $i - \left\lfloor \log \frac{\gamma \log^2(nm)}{\varepsilon^2} \right\rfloor > 1$. This corresponds to whether the frequencies $\left(\widehat{f_j}\right)^p$ in a significant level set are large or not large, informally speaking. If the frequencies are large, then it suffices to estimate them using our sampling-based algorithm. However, if the frequencies are not large, then subsampling must first be performed before we can estimate the frequencies using our sampling-based algorithm.

Suppose $i - \left\lfloor \log \frac{\gamma \log^2(nm)}{\varepsilon^2} \right\rfloor \leq 1$, so that $\frac{1}{2^i} \geq \frac{\varepsilon^2}{\gamma \log^2(nm)}$. Since $j \in \Gamma_i$, we have $(f_j)^p \in \left[\frac{\lambda \cdot \widetilde{F_p}}{2^i}, \frac{2\lambda \widetilde{F_p}}{2^i}\right]$ and thus

$$(f_j)^p \geq \frac{\varepsilon^2 \cdot \widetilde{F_p}}{\gamma \log^2(nm)}, \qquad f_j \geq \frac{\varepsilon^{2/p} \cdot \left(\widetilde{F_p}\right)^{2/p}}{\gamma^{1/p} \log^{1/p}(nm)}$$

Moreover, for $\ell = \max\left(1, i - \left\lfloor \log \frac{\gamma \log^2(nm)}{\varepsilon^2} \right\rfloor\right)$, we have $\ell = 1$, so we consider the outputs by the Morris counters $H_1^{(r)}$. By Lemma 3.4, we have that with probability at least $\frac{9}{10}$, $H_\ell^{(r)}$ outputs $\widehat{f_j}$ such that

$$\left(1 - \frac{\varepsilon}{8\log(nm)}\right) \cdot (f_j)^p \leq (\widetilde{f_j})^p \leq \left(1 + \frac{\varepsilon}{8\log(nm)}\right) \cdot (f_j)^p,$$

as desired.

For the other case, suppose $i - \left\lfloor \log \frac{\gamma \log^2(nm)}{\varepsilon^2} \right\rfloor > 1$, so that $\ell = i - \left\lfloor \log \frac{\gamma \log^2(nm)}{\varepsilon^2} \right\rfloor$ and $p_\ell = 2^{1-\ell}$. Therefore,

$$\frac{1}{2^\ell} = \frac{\gamma \log^2(nm)}{\varepsilon^2} \frac{1}{2^i}.$$

Since $j \in \Gamma_i$, we have again $(f_j)^p \in \left[\frac{\lambda \cdot \widetilde{F_p}}{2^i}, \frac{2\lambda \widetilde{F_p}}{2^i}\right]$ and therefore,

$$(f_j)^p \geq \frac{F_p}{4 \cdot 2^i} \geq \frac{\varepsilon^2}{4\gamma \log^2(nm)} \frac{F_p}{2^\ell}.$$

Conditioning on the event $\mathcal{E}_2$, we have $F_p(I_\ell^{(r)}) \le \frac{32F_p}{2^\ell}$ and thus

$$(f_j)^p \ge \frac{F_p}{4\cdot 2^i} \ge \frac{\varepsilon^2}{128\gamma\log^2 n}\cdot F_p(I_\ell^{(r)}).$$

Therefore by Lemma 3.4, we have that with probability at least $\frac{9}{10}$, $H_\ell^{(r)}$ outputs $\widehat{f_j}$ such that

$$\left(1-\frac{\varepsilon}{8\log(nm)}\right)\cdot(f_j)^p \le (\widetilde{f_j})^p \le \left(1+\frac{\varepsilon}{8\log(nm)}\right)\cdot(f_j)^p,$$

as desired. □

We now justify the approximation guarantees of our algorithm.

Lemma 3.6. $\mathbf{Pr}\left[\left|\widehat{F_p}-F_p\right| \le \varepsilon\cdot F_p\right] \ge \frac{2}{3}$.

Proof. We would like to show that for each level set i, we accurately estimate its contribution C_i, i.e., we would like to show $|\widehat{C_i}-C_i| \le \frac{\varepsilon}{8\log(nm)}\cdot F_p$ for all i. For a fixed i, recall that $C_i = \sum_{j\in\Gamma i}(f_j)^p$, where $j\in\Gamma_i$ if $(f_j)^p \in \left[\frac{\lambda\cdot\widetilde{F_p}}{2^i}, \frac{2\lambda\widetilde{F_p}}{2^i}\right)$. On the other hand, $\widehat{C_i}$ is a scaled sum of items j whose estimated frequency satisfies $\left(\widehat{f_j}\right)^p \in \left[\frac{\lambda\cdot\widetilde{M}}{2^i}, \frac{2\lambda\cdot\widetilde{M}}{2^i}\right]$. Then j could be classified into contributing to $\widehat{C_i}$ even if $j\notin\Gamma_i$. Thus we first consider an idealized process where j is correctly classified across all level sets and show that in this idealized process, we achieve a $(1+O(\varepsilon))$-approximation to F_p. We then argue that because we choose λ uniformly at random, then only a small number of coordinates will be misclassified and so our approximation guarantee will only slightly degrade, but remain a $(1+\varepsilon)$-approximation to F_p.

Idealized process. We first show that in a setting where $(\widehat{f_j})^p$ is correctly classified for all j, then for a fixed level set i, we have $|\widehat{C_i}-C_i| \le \frac{\varepsilon}{8\log(nm)}\cdot F_p$ with probability $1-\frac{1}{\mathrm{poly}(nm)}$.

For a fixed $r\in[R]$, let $\mathcal{E}_1$ be the event that $|I_i^{(r)}| \le \frac{32n}{2^i}$ and let $\mathcal{E}_2$ be the event that $F_p(I_i^{(r)}) \le \frac{32F_p}{2^i}$. Let $\mathcal{E}_3$ be the event that

$$\left(1-\frac{\varepsilon}{8\log(nm)}\right)\cdot(f_j)^p \le (\widetilde{f_j})^p \le \left(1+\frac{\varepsilon}{8\log(nm)}\right)\cdot(f_j)^p.$$

Conditioned on $\mathcal{E}_1$, $\mathcal{E}_2$, and $j\in I_i^{(r)}$, then we have that $\mathbf{Pr}[\mathcal{E}_3] \ge \frac{9}{10}$, by Lemma 3.5.

We define $\widehat{C_i^{(r)}} := \frac{1}{p_\ell}\sum_{(j,\widehat{f_j})\in S_i^{(r)}}\left(\widehat{f_j}\right)^p$. Therefore, we have that $\widehat{C_i} = \mathrm{median}_{r\in[R]}\,\widehat{C_i^{(r)}}$, recalling $\ell = \max\left(1, i-\left\lfloor\log\frac{\gamma\log(nm)}{\varepsilon^2}\right\rfloor\right)$.

Conditioned on $\mathcal{E}_3$, we have

$$\left(1-\frac{\varepsilon}{8\log(nm)}\right)D_i^{(r)} \le \widehat{C_i^{(r)}} \le \left(1+\frac{\varepsilon}{8\log(nm)}\right)D_i^{(r)},$$

where we define

$$D_i^{(r)} = \frac{1}{p_\ell}\sum_{j\in S_i^{(r)}\cap\Gamma_i}(f_j)^p.$$

Note that

$$\mathbb{E}\left[D_i^{(r)}\right] = \frac{1}{p_\ell}\sum_{j\in\Gamma_i}p_\ell\cdot(f_j)^p = C_i,$$

where we recall that C_i denotes the contribution of level set Γ_i. We also have

$$\mathbb{V}\left[D_i^{(r)}\right] \leq \frac{1}{(p_\ell)^2}\sum_{j\in\Gamma_i} p_\ell\cdot(f_j)^{2p} \leq \frac{\varepsilon^2}{2^{i-1}\cdot(\gamma^2\log(nm))}\sum_{j\in C_i}(f_j)^{2p}.$$

For $j\in\Gamma_i$, we have $(f_j)^{2p}\leq\frac{4\lambda^2(\widetilde{F_p})^2}{2^{2i}}\leq\frac{16(F_p)^2}{2^{2i}}$, since $\lambda\leq 1$ and $\widetilde{F_p}\leq 2F_p$. Therefore for sufficiently large γ,

$$\mathbb{V}\left[D_i^{(r)}\right] \leq \frac{\varepsilon^2|\Gamma_i|}{100p^2\cdot 2^i\cdot\log^2(nm)}(F_p)^2.$$

Since $\frac{|\Gamma_i|}{2^i}\leq\phi_i\leq 1$, then

$$\mathbb{V}\left[D_i^{(r)}\right] \leq \frac{\varepsilon^2}{10000p\log(nm)}\cdot(F_p)^2.$$

Thus by Chebyshev's inequality, we have

$$\mathbf{Pr}\left[\left|D_i^{(r)}-C_i\right|\geq\frac{\varepsilon}{10p\log(nm)}\cdot F_p\right]\leq\frac{1}{10}.$$

Therefore,

$$\mathbf{Pr}\left[\left|\widehat{C_i^{(r)}}-C_i\right|\leq\frac{\varepsilon}{10p\log(nm)}\cdot F_p \;\middle|\; \mathcal{E}_1\wedge\mathcal{E}_2\right]\geq\frac{4}{5}.$$

To analyze the probability of the events $\mathcal{E}_1$ and $\mathcal{E}_2$ occurring, note that in level ℓ, each item is sampled with probability $2^{-\ell+1}$. Hence,

$$\mathbb{E}\left[|I_\ell^{(r)}|\right]\leq\frac{n}{2^{\ell-1}},\qquad \mathbb{E}\left[F_p(I_\ell^{(r)})\right]\leq\frac{F_p}{2^\ell-1}.$$

Since $\mathcal{E}_1$ is the event that $|I_\ell^{(r)}|\leq\frac{32n}{2^\ell}$ and $\mathcal{E}_2$ is the event that $F_p(I_\ell^{(r)})\leq\frac{32F_p}{2^\ell}$, then by Markov's inequality, we have

$$\mathbf{Pr}\left[E_1\right]\geq\frac{15}{16},\qquad \mathbf{Pr}\left[E_2\right]\geq\frac{15}{16}.$$

By a union bound,

$$\mathbf{Pr}\left[\left|\widehat{C_i^{(r)}}-C_i\right|\leq\frac{\varepsilon}{10p\log(nm)}\cdot F_p\right]\geq 0.676.$$

Since $\widehat{C_i}=\text{median}_{r\in[R]}\,\widehat{C_i^{(r)}}$ over $R=O(\log\log n)$ independent instances, then we have

$$\mathbf{Pr}\left[\left|\widehat{C_i}-C_i\right|\leq\frac{\varepsilon}{10p\log(nm)}\cdot F_p\right]\geq 1-\frac{1}{\text{polylog}(n)}.$$

Hence by a union bound over the $p\log(nm)$ level sets,

$$\left|\widehat{F_p}-F_p\right| = \left|\sum_{i=1}^{p\log(nm)}\widehat{C_i}-\sum_{i=1}^{p\log(nm)}C_i\right| \leq \sum_{i=1}^{p\log(nm)}\left|\widehat{C_i}-C_i\right|$$
$$\leq \sum_{i=1}^{p\log(nm)}\frac{\varepsilon}{10p\log(nm)}\cdot F_p\leq\frac{\varepsilon}{10}\cdot F_p.$$

Randomized boundaries. Given a fixed $r\in[R]$, we say that an item $j\in[n]$ is misclassified if there exists a level set Γ_i such that

$$(f_j)^p\in\left[\frac{\lambda\cdot\widetilde{F_p}}{2^i},\frac{2\lambda\cdot\widetilde{F_p}}{2^i}\right],$$

but for the estimate $\widetilde{f_j}$, we have either

$$(\widetilde{f_j})^p < \frac{\lambda \cdot \widetilde{F_p}}{2^i} \qquad \text{or} \qquad (\widetilde{f_j})^p \geq \frac{2\lambda \cdot \widetilde{F_p}}{2^i}.$$

By Lemma 3.4, we have that conditioned on $\mathcal{E}_3$,

$$\left(1 - \frac{\varepsilon}{8\log(nm)}\right) \cdot (f_j)^p \leq (\widetilde{f_j})^p \leq \left(1 + \frac{\varepsilon}{8\log(nm)}\right) \cdot (f_j)^p,$$

independently of the choice of λ. Since $\lambda \in \left[\frac{1}{2}, 1\right]$ is chosen uniformly at random, then the probability that $j \in [n]$ is misclassified is at most $\frac{\varepsilon}{2\log(nm)}$.

Furthermore, if $j \in [n]$ is misclassified, then it can only be classified into either level set Γ_{i+1} or level set Γ_{i-1}, because $\left(\widehat{f_j}^p\right)$ is a $\left(1 + \frac{\varepsilon}{8\log(nm)}\right)$-approximation to $(f_j)^p$. Thus, a misclassified index induces at most $2(f_j)^p$ additive error to the contribution of level set Γ_i. In expectation across all $j \in [n]$, the total additive error due to misclassification is at most $F_p \cdot \frac{\varepsilon}{2\log(nm)}$. Therefore by Markov's inequality for sufficiently large n and m, the total additive error due to misclassification is at most $\frac{\varepsilon}{2} \cdot F_p$ with probability at least 0.999. Hence in total,

$$\mathbf{Pr}\left[\left|\widehat{F_p} - F_p\right| \leq \varepsilon \cdot F_p\right] \geq \frac{2}{3}.$$

□

Putting things together, we give the full guarantees of our F_p estimation algorithm.

Theorem 1.3. *Given a constant $p \geq 1$, there exists a one-pass insertion-only streaming algorithm that has $\tilde{O}\left(n^{1-1/p}\right)$ internal state changes, and outputs $\widehat{F_p}$ such that*

$$\mathbf{Pr}\left[\left|\widehat{F_p} - F_p\right| \leq \varepsilon \cdot F_p\right] \geq \frac{2}{3}.$$

For $p \in [1, 2)$, the algorithm uses $O\left(\frac{\log^{9+3p}(mn)}{\varepsilon^{4+4p}}\right)$ bits of space, while for $p > 2$, the algorithm uses $\tilde{O}\left(\frac{1}{\varepsilon^{4+2p}} n^{1-2/p}\right)$ space.

Proof. The space bound follows from fact that for $p \in [1, 2)$, only $O\left(\frac{\log^{8+3p}(mn)}{\varepsilon^{4+4p}}\right)$ counters are stored, while for $p > 2$, only $\tilde{O}\left(\frac{1}{\varepsilon^{4+2p}} n^{1-2/p}\right)$ counters are stored. The internal state can change each time an item is sampled. Since each item of the stream is sampled with probability $\varrho = \frac{\gamma^2 n^{1-1/p} \log^4(nm)}{\varepsilon^2 m}$, then with high probability, the total number of internal state changes is $\frac{\gamma^2 n^{1-1/p} \log^4(nm)}{\varepsilon^2}$.

Finally for correctness, we have by Lemma 3.6, that

$$\mathbf{Pr}\left[\left|\widehat{F_p} - F_p\right| \leq \varepsilon \cdot F_p\right] \geq \frac{2}{3}.$$

□

B MISSING PROOFS FROM SECTION 4

Theorem 1.2. *Let $\varepsilon \in (0, 1)$ be a constant and $p \geq 1$. Any algorithm that solves the L_p-heavy hitters problem with threshold ε with probability at least $\frac{2}{3}$ requires at least $\frac{1}{2\varepsilon} n^{1-1/p}$ state updates.*

Proof. Consider the two input streams $\mathcal{S}_1$ and $\mathcal{S}_2$ defined as follows. We define the stream $\mathcal{S}_1$ to have length n on a universe of size n. Similar to before, we choose a random contiguous block B of $\varepsilon \cdot n^{1/p}$ stream updates and set them all equal to the same random universe item $i \in [n]$. We randomly choose the remaining $n - n^{1/p}$ updates in the stream so that they are all distinct and none of them are equal to i. Note that by construction, we have

$$F_p(\mathcal{S}_1) = (n - n^{1/p}) + (n^{1/p})^p = 2n - n^{1/p}.$$

Therefore, the item replicated $\varepsilon \cdot n^{1/p}$ times in block B is an $\frac{\varepsilon}{2}$-heavy hitter with respect to L_p.

We also define the stream $\mathcal{S}_2$ to have length n on universe n. As before, we choose $\mathcal{S}_2$ to be a random permutation of $[n]$, so that $F_p(\mathcal{S}_2) = n$.

For fixed $\varepsilon \in (0, 1)$, let $\mathcal{A}$ be an algorithm that solves the ε-heavy hitter problem with respect to L_p, with probability at least $\frac{2}{3}$, while using fewer than $\frac{1}{2\varepsilon} n^{1-1/p}$ state updates. We claim that with probability $\frac{1}{2}$, $\mathcal{A}$ must have the same internal state before and after B in the stream $\mathcal{S}_1$.

Observe that each stream update can be viewed as (i, j) where $i \in [n]$ is the index of the stream update and $j \in [n]$ is the identity of the universe item. For a random universe item $i \in [n]$, the probability that a random stream update $j \in [n]$ alters the state of $\mathcal{A}$ is at most $\frac{1}{2\varepsilon \cdot n^{1/p}}$, since otherwise for a random stream, the expected number of state changes would be larger than $\frac{n^{1-1/p}}{2\varepsilon}$, which would be a contradiction. Because both the choice of B is uniformly and the element $j \in [n]$ that is repeated $\varepsilon \cdot n^{1/p}$ times are chosen uniformly at random, then the expected number of changes of the streaming algorithm on the block B is at most $\frac{\varepsilon \cdot n^{1/p}}{2\varepsilon \cdot n^{1/p}} = \frac{1}{2}$. Hence, the streaming algorithm's state is the same before and after the block B, with probability at least $\frac{1}{2}$.

However, the same argument applies to $\mathcal{S}_2$. Thus with probability at least $\frac{1}{2}$, the streaming algorithm cannot distinguish between $\mathcal{S}_1$ and $\mathcal{S}_2$ if its internal state only changes fewer than $\frac{n^{1-1/p}}{2\varepsilon}$ times. Therefore, any algorithm that solves the L_p-heavy hitter problem with threshold ε with probability at least $\frac{2}{3}$ requires at least $\frac{1}{2\varepsilon} n^{1-1/p}$ state updates. □

C COMPARISON WITH PREVIOUS ALGORITHMS

There are a number of differences between our algorithm and the sample-and-hold approach of [47]. Firstly, once [47] samples an item, a counter will be initialized and maintained indefinitely for that item. By comparison, our algorithm will sample more items than the total space allocated to the algorithm, so we must carefully delete a number of sampled items. In particular, it is NOT correct to delete the sampled items with the largest counter. Secondly, [47] updates a counter each time a subsequent instance of the sample arrives. Because our paper is focused on a small number of internal state changes, our algorithm cannot afford such a large number of updates. Instead, we maintain approximate counters that sacrifice accuracy in exchange for a smaller number of internal state changes. We show that the loss in accuracy can be tolerated in choosing which samples to delete.

Another possible point of comparison is the precision sampling technique of [6], which is a linear sketch, so although it has an advantage of being able to handle insertion-deletion streams, unfortunately it must also be updated for each stream element arrival, resulting in a linear number of internal state changes. Similarly, a number of popular heavy-hitter algorithms such as Misra-Gries [81], CountMin [43], CountSketch [36], and SpaceSaving [80] can only achieve a linear number of internal state changes. By comparison, our sample-and-hold approach results in a sublinear number of internal state changes.

Finally, several previous algorithms are also based on sampling a number of items throughout the stream, temporarily maintaining counters for those items, and then only keeping the items that are globally heavy, e.g., [29, 31]. It is known that these algorithms suffer a bottleneck at $p = 3$,

i.e., they cannot identify the L_p heavy-hitters for $p < 3$. The following counterexample shows why these algorithms cannot identify the L_2 heavy-hitters and illustrates a fundamental difference between our algorithms.

Suppose the stream consists of $\sqrt{n}$ blocks of $\sqrt{n}$ updates. Among these updates, there are $\sqrt{n}$ items with frequency $n^{1/4}$, which we call pseudo-heavy. There is a single item with frequency $\sqrt{n}$, which is the heavy-hitter. Then the remaining items each have frequency 1 and are called light. Note that the second moment of the stream is $\Theta(n)$, so that only the item with frequency $\sqrt{n}$ is the heavy-hitter, for constant $\varepsilon < 1$.

Let $S = \{1, 2, \ldots, n^{1/4}\}$ and suppose for each $w \in S$, block w is a special block that consists of $n^{1/4}$ different items, each with frequency $n^{1/4}$. Let $T = x + S$, for $x = \{1, 2, \ldots, n^{1/8}\}$, so that T consists of the $n^{1/8}$ blocks after each special block. Each block in T consists of $n^{1/8}$ instances of the heavy-hitter, along with $\sqrt{n} - n^{1/8}$ light items. The remaining blocks all consist of light items.

Observe that without dynamic maintenance of counters for different scales, in each special block, we will sample polylog(n) pseudo-heavy items whose counters each reach about $\tilde{O}\left(n^{1/4}\right)$. But then each time a heavy-hitter is sampled, its count will not exceed the pseudo-heavy item before the number of counters before it is deleted, because it only has $n^{1/8}$ instances in its block. Thus with high probability, the heavy-hitter will never be found, and this is an issue with previously existing sampling-based algorithms, e.g., [29, 31].

Our algorithm overcomes this challenge by only performing maintenance on counters that have been initialized for a similar amount of time. Thus in the previous example, the counters for the heavy-hitters will not be deleted because they are not compared to the counters for the pseudo-heavy items until the heavy-hitters have sufficiently high frequency. By comparison, existing algorithms will retain counters for the pseudo-heavy items, because they locally look "larger", at the expense of the true heavy-hitter.

Reference	State Changes	Setting
[81]	$O(m)$	L_1-Heavy Hitters Only
[43]	$O(m)$	L_1-Heavy Hitters Only
[80]	$O(m)$	L_1-Heavy Hitters Only
[36]	$O(m)$	L_2-Heavy Hitters
Our Work	$\tilde{O}\left(n^{1-2/p}\right)$	L_2-Heavy Hitters

Table 1. Summary of our results compared to existing results. We emphasize that reporting L_2 heavy-hitters includes the L_1 heavy-hitters. All algorithms use near-optimal space.

Received June 2023; revised August 2023; accepted September 2023