Finding Heavy-Hitters with Optimal State Changes

WILLIAM SWARTWORTH, Carnegie Mellon University, United States
DAVID P. WOODRUFF, Carnegie Mellon University, United States

A recent work, [Jayaram, Woodruff, Zhou '24] studied the problem of designing streaming algorithms with few changes memory. In particular they showed that one can solve the ϵ-ℓ_k-heavy-hitters problem using $O(\frac{1}{\epsilon}n^{1-1/k}\text{polylog}(n/\epsilon))$ state changes and $O(\text{poly}(1/\epsilon)\log n)$ space, although the log factors here are too large for reasonable implementation.

Our first result is a new algorithm for solving the ℓ_k heavy hitters problem for $k \in [1, 2]$. Our algorithm requires only $O(\frac{1}{\epsilon}n^{1-1/k}\text{poly}(\log\log n)\log\frac{1}{\epsilon})$ state changes and $O(\frac{1}{\epsilon^k}\text{polylog}(n))$ space. We also extend our algorithm to $k \geq 2$, giving the same state change bound, but with an additional $n^{1-2/k}$ factor in the space. Finally, we complement this result by a lower bound showing that our state change complexity is optimal up to $\log\log n$ factors.

At the core of our improved algorithm is a new, and very simple algorithm for solving heavy-hitters in insertion only streams, which may be of independent interest.

CCS Concepts: • **Theory of computation → Sketching and sampling**.

Additional Key Words and Phrases: heavy hitters, streaming algorithms, state changes, morris counters

ACM Reference Format:
William Swartworth and David P. Woodruff. 2025. Finding Heavy-Hitters with Optimal State Changes. *Proc. ACM Manag. Data* 3, 5 (PODS), Article 278 (November 2025), 22 pages. https://doi.org/10.1145/3767714

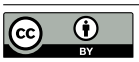
1 Introduction

Streaming algorithms have a long history of study, and have found numerous applications situations where there is a high throughput of data [2, 20]. In the streaming model the goal is to observe a stream of data one update at a time, and to "summarize" the data by producing a sketch of small size from which one can recover a useful statistic in the underlying data.

Perhaps the most classical example of a streaming algorithm is the so-called heavy-hitters problem, where the goal is to find the frequent items in a data stream. This is typically formalized with respect to an ℓ_k norm. If f is the frequency vector associated to the items in the data stream, then an item x is ϵ-heavy if $f_x \geq \epsilon \|f\|_k$. This problem of identifying heavy elements was famously studied by [9] who solved the ℓ_2-heavy-hitters problem, and was studied even earlier by [18] who solved what is now called the ℓ_1-heavy-hitters problem. Since then many other variants of the heavy-hitters problem have been considered, with various twists. For instance some algorithms achieve improved space for insertion only streams [5], while others focus on other metrics such as the update time required to process stream updates and update the sketch [15, 16].

Recently [14] introduced a new metric with which to measure the complexity of a streaming algorithm, namely the number of state changes made by the algorithm. After each update is received from the stream, an algorithm can choose whether or not to update its state. Most classical streaming

The authors were supported in part by a Simons Investigator Award and Office of Naval Research award number N000142112647.
Authors' Contact Information: William Swartworth, Carnegie Mellon University, United States, wswartwo@andrew.cmu.edu; David P. Woodruff, Carnegie Mellon University, United States, dwoodruf@cs.cmu.edu.



ACM 2836-6573/2025/11-ART278
https://doi.org/10.1145/3767714

algorithms do not give good state change bounds. For example CountSketch (and in fact all linear sketches) update state on all stream updates. Thus it is perhaps surprising that [14] was able to give an ℓ_2-heavy-hitters algorithm (among other algorithms) that achieved polylogarithmic space while only making $\sqrt{n}$ state changes, again up to polylogarithmic factors.

While state changes have not previously received much attention in the context of streaming algorithms, the model has natural motivations that make it worth studying. The most natural setting for this model is in situations where write operations are more expensive than read operations. For example, modern flash memory, which is widely used for long-term storage typically wears out after roughly ten thousand write operations [3], while read operations do not create nearly as much wear. So it can be useful to design algorithms that minimize write operations if an algorithm stores its data in long-term memory. This problem of designing algorithms to reduce the wear on memory has previously been studied outside the context of streaming algorithms [7, 8]. Similarly in situations where there is shared memory, read operations are often somewhat less expensive as they do not require synchronization. In a distributed setting where many threads are processing streaming data and maintaining a shared data structure, having many write operations may lead to more blocked threads on average, which could reduce processing time.

Outside of practical considerations, the few-state-change model may inspire new theoretical developments for streaming algorithms. For example, algorithms that rarely change state, tend to have very low update time on average. For certain streaming problems e.g., (ϵ, ℓ_2)-heavy-hitters, it is not known whether one can obtain truly $O(1)$ update time in the insertion-only model (independent of ϵ). It seems plausible that optimizing state changes could also lead to improved update-time bounds. We direct the reader to [14] for additional motivation for the few-state-change model.

Given the above considerations, the few-state-change model is worthy of more consideration. The previous work [14] gave algorithms that achieve optimal space bounds and state-change bounds up to polylogarithmic factors; however these factors are quite large, and the analysis is correspondingly complex. We ask:

Is it possible to obtain a nearly optimal state change bound for ℓ_k-heavy-hitters for $1 < k < \infty$ using space that is optimal up to polylogarithmic factors?

We show that this is indeed possible, and present a novel algorithm. Additionally, we show that one can strip away much of the complexity of the analysis of [14] for the heavy-hitter problem. In this direction, we give a simple, but new, algorithm with a short analysis that achieves optimal space and state changes up to log factors. This algorithm is easy to implement and competitive in practice. See the appendix for empirical results.

1.1 Our Models

Streaming model. We consider a stream of length m on a universe of $[n]$, with a series of updates $x_1, \ldots, x_m \in [n]$. Throughout we will always use m to denote the stream length and n to denote the universe size. For simplicity, we assume throughout that $m = \text{poly}(n)$. We use f to denote the frequency vector at the end of the stream, so f_i is the number of occurrences of item i in the stream. For a subset $S \subseteq [m]$ of stream updates, we use f_S to denote the frequency vector corresponding to the updates at times in S.

We are interested in minimizing the number of state changes of a streaming algorithm $\mathcal{A}$. If Π_i is the internal state of $\mathcal{A}$ after i stream updates, then the number of state changes is defined to be the number of i such that $\Pi_{i+1} \neq \Pi_i$. We allow the algorithm to receive independent random bits at every time step without this counting as additional state changes. This is a natural assumption in practice as one could for example use a random number generator seeded with the current clock time.

Heavy-hitters. For our purposes, we define the heavy-hitters problem as follows.

Problem 1. *We say that $i \in [n]$ is an (ϵ, ℓ_k)-heavy-hitter for f if $f_i \geq \epsilon \|f\|_k$. In the (ϵ, ℓ_k)-heavy-hitters problem, items $x_1, \ldots, x_m$ are presented in a stream. The goal is to output a set $\mathcal{H}$ such that $\mathcal{H}$ contains all (ϵ, ℓ_k) heavy-hitters and with $|\mathcal{H}| \leq O(\epsilon^{-k})$.*

1.2 Our Results

We give several results for solving the heavy-hitters problem with different trade-offs between state changes and space. Our first result shows that our core ℓ_2 heavy-hitters algorithm yields a good space and state change guarantee, provided that the sampling rate is chosen appropriately. The first result is showing the correctness of the following algorithm.

Algorithm 1.
Accept a sampling rate parameter p.

1. *Maintain a list of counters L, initially empty.*
2. *For each stream update, sample x_i at rate p. If the item is sampled start a counter for universe item x_i, and append that counter to L. (There may be multiple counters tracking a single universe item.)*
3. *Increment all counters in L for x_i*
4. *Let $L = [C_1, \ldots, C_k]$. For $i = 1 \ldots k-1$ if $C_{i+1} \geq \frac{1}{2} C_i$, then remove C_i, while preserving the ordering of the remaining counters.*
5. *Return a frequency vector $\hat{f}$ that contains the counts C_i as estimates of their respective values. Use 0 as estimates for all other values.*

Theorem 2. *Let $h \in [n]$ be a $(9/10, \ell_2)$-heavy-hitter. If $\frac{1}{2\|f\|_2} \leq p \leq \frac{2}{\|f\|_2}$. Let $\hat{f}$ be the count estimates returned by the algorithm. Then with at least constant nonzero probability, $\hat{f}_h \geq \frac{1}{5} \|f\|_2$. Moreover, $\hat{f}_x \leq f_x$ holds deterministically for all x. This algorithm uses $O(\log m(\log m + \log n))$ space and $O(\|f\|_2)$ state changes.*

Additionally, we remark that by running this algorithm for p at $O(\log m)$ different scales, we obtain the same guarantee with no assumption that $\|f\|_2$ is known to the algorithm, but with an additional $O(\log m)$ factor on both the space and state changes. It is often preferable to have a bound that scales with n, rather than with $\|f\|_2$, which might be larger. By subsampling to reduce $\|f\|_2$ to $O(\sqrt{n})$, we have the following result.

Theorem 3. *There is an algorithm that solves the $(0.9, \ell_2)$-heavy-hitters problem with $3/4$ probability using $O(\log^3 n)$ space and $O(\sqrt{n} \log^3 n)$ state changes.*

For the upper bounds, we ask whether it is possible to achieve the optimal state-change bound for ℓ_k heavy hitters, while retaining polylog(n) space. We show that we can get surprisingly close; for all $1 < k \leq 2$ our number of state changes is optimal to within $\log \log n$ factors.

Theorem 4. *For $k \in [1, 2]$, there is an algorithm that solves the (ϵ, ℓ_k)-heavy-hitters problem using $O(\frac{1}{\epsilon^2}(\log \frac{1}{\epsilon}) \log^6 n \log \log n)$ space and $O(\frac{1}{\epsilon}(\log \frac{1}{\epsilon}) n^{1-1/k} \operatorname{poly} \log \log n)$ state changes.*

In the appendix, we show that a small change to this argument solves heavy-hitters for $k \geq 2$.

Theorem 5. *For $k \geq 2$, there is an algorithm that solves the (ϵ, ℓ_k)-heavy-hitters problem using $\tilde{O}(\frac{1}{\epsilon^2} n^{1-2/k})$ space and $O(\frac{1}{\epsilon}(\log \frac{1}{\epsilon}) n^{1-1/k} \operatorname{poly} \log \log n)$ state changes.*

One might wonder if the additional $\log \frac{1}{\epsilon}$ factors are necessary in the above state change bounds. This factor shows up within the space bound of many heavy-hitter algorithms, including BPTree

[5], which is state-of-the-art for insertion-only streams. To the best of our knowledge, it is an open question whether the $\log \frac{1}{\epsilon}$ is necessary in the space bound for insertion-only streams. Interestingly, we show that it is necessary in the state-change bound:

THEOREM 6. *Let $k \in [1, \infty)$ Suppose that an algorithm solves the (ϵ, ℓ_k)-heavy-hitter problem with probability at least $3/4$. Then this algorithm must use at least $\Omega(n^{1-1/k} \frac{1}{\epsilon} \log \frac{1}{\epsilon})$ state changes.*

1.3 Existing Work

Our work is closely connected to [14] who introduced the state-change model in the context of streaming algorithms. While they obtain optimal bounds for ℓ_k-heavy-hitters up to $\text{polylog}(n)$ factors, these $\text{polylog}(n)$ factors are quite large – roughly $\log^{11+3k}$ if their results are applied directly, which is not implementable in practice. While some of these log factors could likely be improved by tightening their bounds, the extra log factors suggest that there may be a simpler algorithm and analysis. Indeed our algorithm for ℓ_2-heavy-hitters described below already yields much better polylog factors for both state changes and space. Moreover, our algorithm is extremely simple and the core analysis fits within a page. The algorithm of [14] creates counters for randomly sampled stream items, similar to our algorithm, but their counter maintenance is more complicated. They maintain geometric counters going backwards in time. Among counters that have existed for roughly the same time, they discard all but the smallest $\text{polylog}(n)$ counters. We do not even need to store the times at which the counters are created; we simply keep a counter alive until a new counter which was initialized at a later time overtakes its count (to within a factor of 2).

We also note that a counter-based algorithm was used to solved the heavy-hitters problem in [6], who used "pick-and-drop" sampling to find ℓ_k-heavy-hitters for $k > 3$. This algorithm again falls into the framework of "counter-based algorithms." This algorithm essentially creates new counters with some fixed probability, and holds the counter until some condition is met. The algorithm and its analysis heavily rely on the fact that $k > 2$, so is not applicable to our setting where $k \leq 2$. See [5] for a counterexample to "pick-and-drop" sampling when $k = 2$.

1.4 Technical Overview

The core ℓ_2 algorithm. Our main insight is a counter-based algorithm for solving ℓ_2-heavy hitters. While extremely simple and natural, this algorithm appears to be new. We describe our algorithm for solving the $(0.9, \ell_2)$-heavy-hitters problem.

The base algorithm takes as a parameter a sampling rate p, and maintains a list of counters for various universe items, initially empty. When a stream update for a universe item x is received, we choose to sample x with probability p. If x is sampled then we start a new counter Z for x, which maintains the count of x from that point in the stream onwards, and append it to the list. As stated so far, this procedure would create pm counters, which would use too much space. Our key insight is a simple procedure for pruning the list.

Suppose that $[Z_1, \ldots, Z_N]$ is the list of active counters at some fixed point in time. The rule is that if the count of Z_{i+1} ever becomes at least half of the count for Z_i, then we delete the counter Z_i. This means that we only ever store $O(\log m)$ counters since each count is at most the stream length m, and also each count is at least twice the count following it. Since each counter maintains both the identity of a universe item, as well as a count, the total space used is $O(\log m(\log n + \log m))$ bits, which is $O(\log^2 n)$ for $m = \text{poly}(n)$.

It turns out that the correct sampling rate to choose is $p = \frac{1}{\|f\|_2}$, just large enough to pick up a single heavy-hitter with good probability. The idea here is to consider what we call "competing pairs". If i is an update to the heavy-hitter h we say that j competes with i if j comes after i and the count for item x_j starting at time j eventually reaches half of the count to h starting at i. A

simple argument shows that the number of competing pairs is $O(\|f\|_2^2)$, so that each heavy hitter update is part of $O(\|f\|_2)$ competing pairs on average. This means that we are likely to sample a heavy hitter that is part of at most $O(\|f\|_2)$ competing pairs, and conditioned on sampling it, we are likely to not sample any of the updates that form a competing pair.

Note that we sample universe item i at most $\frac{f_i}{\|f\|_2}$ times in expectation. Each such counter that is started leads to at most f_i state changes from increment operations, and so the total number of state changes is at most $\sum_i \frac{f_i^2}{\|f\|_2} = \|f\|_2$ in expectation. If $\|f\|_2$ is known to the algorithm at runtime, then it turns out that we can subsample the original stream at a rate of $\frac{\sqrt{n}}{\|f\|_2}$. This preserves the heaviness of the heavy-hitter up to a constant factor and reduces the ℓ_2 norm of the stream to $\sqrt{n}$. So this yields an algorithm that makes $O(\sqrt{n})$ state changes.

This implies correctness of our algorithm, provided that our sampling rate p was chosen to be $\frac{1}{\|f\|_2}$ up to constant factors. An issue though is that $\|f\|_2$ typically is not known to an algorithm at runtime. In this situation, one option is to simply try all possible values of p up to a factor of 2. So we run the algorithm above independently for $p = 1, \frac{1}{2}, \frac{1}{4}, \ldots, \frac{1}{m}$. One of these p's is guaranteed to lie within a factor of 2 of $\frac{1}{\|f\|_2}$, and therefore achieves correctness, along with the state-change and space bound. This run of the algorithm provides an approximate count of the heavy-hitter accurate to within a constant factor with good probability. On all runs, the approximate counts returned are guaranteed to be underestimates, so we simply return the item with the highest count across all runs.

Optimizing state changes. The algorithm above yields $O(\sqrt{n}\log n)$ state changes, which is optimal to within a $\log n$ factor. We ask whether it is possible to do even better. It turns out that it is. We obtain an algorithm that makes $O(\sqrt{n}\operatorname{poly}\log\log n)$ state changes, while still only using $\operatorname{polylog}(n)$ space. The reason that the algorithm above gives a $\log n$ dependence for the number of state changes is that we have to run an instance of the heavy-hitters algorithm for each "guess" at the scale of $\|f\|_2$. To do better, we track ℓ_2 on prefixes of the stream. Ideally, we would like to track ℓ_2 to within a constant factor, but it is not clear how to do this without incurring additional $\log n$ factors in the state change bound. Instead we will settle for maintaining a $\operatorname{polylog} n$ multiplicative approximation.

Here we use a standard approach for approximating F_k moments, introduced by [1]. The idea is to rescale each coordinate i of f by an inverse exponential $1/u_i^{1/k}$, where u is exponentially distributed with density e^{-x}. The maximum coordinate after the rescaling is distributed as $\|f\|_k / u^{1/k}$, and so this maximum gives a constant factor approximation to ℓ_k with good probability. Moreover, the maximum is a $(\frac{1}{\log n}, \ell_k)$ heavy-hitter, so when $k = 2$, the algorithm that we discussed above finds it, along with an approximate count with good probability. We note that one could also apply other ℓ_k tracking algorithms, such as those based on level sets (for example [10, 13, 17]) as done by [14].

This yields an algorithm with $O(\operatorname{polylog}(n)\sqrt{n})$ state changes for tracking ℓ_2 at all times to within, say, a $\log n$ factor. To reduce the state changes further, the observation is that we can first of all assume that our original stream has length $O(n)$ by sub-sampling at rate n/m (which preserve the heaviness of the heavy-hitter). Then we are able to subsample the resulting stream at a rate of $1/\operatorname{polylog}(n)$ which preserves ℓ_2 to within a $\operatorname{polylog}(n)$ factor. The universe size is now reduced to say $n/\log^r(n)$, so our effective universe size is $O(n/\log^r n)$. This means that we now only use $O(\operatorname{polylog}(n)\sqrt{n/\log^r n})$ state changes. By choosing r appropriately, we may make the total number of state changes smaller than $O(\sqrt{n})$ for tracking ℓ_2.

Now, given a $\operatorname{polylog}(n)$ approximation to ℓ_2 at all times we start a new instance of our algorithm each time our ℓ_2 approximation changes by a factor of 2. We maintain only the $O(\log\log n)$ most recent instances of the algorithm, which is sufficient to guarantee that one instance was

started before half of the ℓ_2 mass occurred in the stream. It is sufficient for this instance to be run for $O(\log\log n)$ choices of p. Each instance of the algorithm can therefore be made to have $O(\sqrt{n}\log\log n)$ state changes.

Finally, to extend from 0.9-heavy-hitters to ϵ-heavy-hitters, we apply the standard reduction of hashing our universe into $O(1/\epsilon^2)$ buckets. Each heavy-hitter becomes 0.9-heavy in its respective bucket with good probability. So by repeating this process $O(\log\frac{1}{\epsilon})$ times, we identify all heavy-hitters with good probability. Note that we settle for obtaining a set of size $O(1/\epsilon^2)$ containing the heavy-hitters. The reason that we do not require everything in our set to be $\Omega(\epsilon)$ heavy is that we do not have a way of tracking ℓ_k to within a constant factor using $O(\sqrt{n}\,\mathrm{poly}\log\log n)$ state changes.

ℓ_k-heavy-hitters. Most of the ideas discussed above extend fairly directly to ℓ_k heavy-hitters for $1 \le k \le 2$ with a few modifications that turn out to complicate the analysis. The main issue is in obtaining the optimal $O(n^{1-1/k})$ state change bound, without extra $\log n$ factors. If we were to sample directly at a $\frac{1}{\|f\|_k}$ rate as above, then this would be sufficient for correctness. However the number of state changes is now $\sum_i \frac{f_i^2}{\|f\|_k}$, which can be too large. So instead of maintaining exact counts we use Morris counters, which can maintain an approximate count for i with $O(\log f_i)$ state changes. This reduces the total number of state changes to $\sum_i \frac{f_i \log f_i}{\|f\|_k}$, which we can bound by $O(n^{1-1/k})$ for $\|f\|_k \le n^{1/k}$ (As for ℓ_2 we reduce to this case by tracking $\|f\|_k$ and then sub-sampling the stream at a rate of roughly $n^{1/k}/\|f\|_k$.) A few complications arise. For example, the Morris counters can occasionally overestimate counts, so we need to be careful when simply taking the maximum across runs. While the details are technical, this turns out to not cause serious problems.

Finally, in the appendix we address ℓ_k for $k > 2$. The main point is to observe that an (ϵ, ℓ_k)-heavy-hitter is also an $(\epsilon n^{1/k-1/2}\epsilon, \ell_2)$-heavy-hitter. Then we can apply our ℓ_2 analysis direction. The issue is that as in the $k < 2$ case, this may give too many candidate heavy hitters. To avoid this, we give a slightly stronger bound on the Morris counters than for the $k < 2$ case to show that they rarely overestimate dramatically.

Lower Bound. Finally we show that one cannot hope to do better than $O(\frac{1}{\epsilon}\log\frac{1}{\epsilon}n^{1-1/k})$ state changes for solving (ϵ, ℓ_k)-heavy-hitters in general, even with no restrictions on space. Our lower bound is inspired by the corresponding lower bound of [14], and is perhaps even slightly simpler.

We first consider the problem of solving 1/2-heavy-hitters with failure probability δ. We consider two streams. For the first stream all universe items occur in succession: $1, 2, 3, \ldots, n$. The second stream is the same, except that one item at random is chosen to repeat $n^{1/k}$ times when it occurs, making it heavy. So for example our stream might be: $1, 2, 3, 3, 3, 3, 3, 4, 5, \ldots, n$. Roughly, in order to succeed the algorithm must change state on one of the repeated elements, otherwise it cannot distinguish the two streams. But if the algorithm has probability q of changing state on the first repeated element, then it has probability $(1-q)^{n^{1/k}}$ of never changing state on any of the repeated elements (since the probability q cannot change without the algorithm changing state). This means that q typically needs to be on the order $\log\frac{1}{\delta}n^{-1/k}$, which implies that a correct algorithm must make $\Omega(n^{1-1/k}\log\frac{1}{\delta})$ state changes in expectation.

Now for ϵ-heavy hitters, we divide the universe into ϵ^{-k} pieces and construct an instance similar to the above on each piece. We then must effectively solve the 1/2-heavy-hitters problem on a universe of size $\epsilon^k n$ with failure probability at most $1/\mathrm{poly}(\epsilon)$. Summing over all instances, then gives the desired bound.

2 The Counter Algorithm

In this section, we present the analysis for our counter algorithm, assuming that we have an accurate estimate of the F_2 frequency moment. We will later show how to track F_2 using counters, which will allow us to develop an algorithm that does not rely on knowing F_2.

2.1 The core heavy-hitter algorithm.

Theorem 7. *Let $h \in [n]$ be a $(9/10, \ell_2)$-heavy-hitter. If $\frac{1}{2\|f\|_2} \le p \le \frac{2}{\|f\|_2}$. Let $\hat{f}$ be the count estimates returned by the algorithm. Then with at least constant nonzero probability, $\hat{f}_h \ge \frac{1}{5}\|f\|_2$. Moreover, $\hat{f}_x \le f_x$ holds deterministically for all x. This algorithm uses $O(\log m(\log m + \log n))$ space and $O(\|f\|_2)$ state changes.*

Proof. For stream indices $i < j \in [m]$, we say that (i, j) is a *competing pair* for h if $x_i = h$ and there exists $T \ge j$ such that $f_{[j,T]}(x_j) \ge \frac{1}{2} f_{[i,T]}(h)$.[1]

We first count the competing pairs for h. Let $f = [f_1, \ldots, f_n]$ be the frequency vector. Note that

$$\#\{i : \exists j, x_j = k \text{ and } (i, j) \text{ is a competing pair}\} \le 2f_k. \tag{1}$$

To form a competing pair (i, j) with $x_j = k$, there are at most f_k ways to choose j, and by the above fact there are at most $2f_k$ ways to choose i. That means that are at most $2\sum_k f_k^2 = 2\|f\|_2^2$ competing pairs.

Therefore the average update to h is part of at most $2\|f\|_2^2 / f_h$ competing pairs. Say that an update to h is bad, if it is part of at least $2M\|f\|_2^2 / f_h$ competing pairs, where we set $M = \frac{2f_h}{\|f\|_2}$. By Markov's bound, there are at most $\frac{1}{M} f_h = \frac{1}{2}\|f\|_2$ bad updates.

Since h is a $9/10$ heavy-hitter, there are at least $\frac{2}{5}\|f\|_2$ good updates. At our sampling rate, we sample one of the first $\frac{1}{5}\|f\|_2$ good updates with at least constant nonzero probability. By the definition of the algorithm, this item will be kept forever, and the associated count will be at least $\frac{1}{5}\|f\|_2$, provided that we do not sample any of the at most $6\|f\|_2$ values which form a competing pair. By our sampling rate, we sample one of these items with probability at most $9/10$ by a union bound. So our algorithm succeeds with constant nonzero probability.

Next we bound the number of state changes. The universe item i has frequency f_i and will therefore be sampled with probability at most $\frac{2f_i}{\|f\|_2}$. If item i is sampled, then the counter for item i will be updated at most f_i times. Therefore the expected number of state changes is at most

$$\sum_i \frac{2f_i}{\|f\|_2} f_i \le 2\|f\|_2.$$

By Markov's inequality, the number of state changes is bounded by $O(\|f\|_2)$ with constant probability. (On runs where the number of state changes reaches $c\|f\|_2$, for a chosen constant c, we may simply abort.)

For the space bound, observe that we maintain at most $\log m$ counters, since each counter is always at least twice the next, and the value of each counter is bounded by m. Each counter stores both a universe identity and a count, which requires $O(\log n + \log m)$ space.

□

Corollary 8. *Suppose that $\|f\|_2$ is promised to lie in $[a, b]$. Then there is an algorithm using $O(\log(b/a)\log m(\log n + \log m))$ space and $O(\log(b/a)\|f\|_2)$ state changes that achieves the guarantee of Theorem 7 with $3/4$ probability.*

[1] $f_{[i,T]}(h)$ means the frequency count of h on the stream interval $[i, T]$.

Proof. Recall that Theorem 7 only provides underestimates of the true frequency values. We may therefore run this algorithm on power-of-two increments with $p = a, 2a, 4a, \dots b$. Since one of these values of p is in the range of Theorem 7, we may simply take the maximum frequency count entrywise over runs, so that the guarantee holds with constant nonzero probability. By repeating the entire procedure a constant number of times (and again taking maxes) we may boost the overall success probability to 3/4. □

Theorem 9. *There is an algorithm that solves the* $(0.9, \ell_2)$*-heavy-hitters problem with* 3/4 *probability using* $O(\log^3 n)$ *space and* $O(\sqrt{n}\log^3 n)$ *state changes.*

Proof. We run essentially the same algorithm as in Corollary 8 above but with a few small changes. First suppose that $\|f\|_2$ is known to lie in $[M, 2M]$. Then by applying Lemma 13 we may subsample at rate $\Theta(\sqrt{n}/M)$ to reduce to the case where $\|f\|_2 = O(n)$. After subsampling, the heavy-hitter remains $\Theta(1)$-heavy with good probability.

Now we recall the Morris+ counter of [21]. This yields a counter which is guaranteed to be accurate to within $1 \pm \frac{1}{10}$ multiplicative error, and which is correct with probability at least $1 - \delta$, and uses $O(\log\log N + \log\log\frac{1}{\delta})$ space and $O(\log N \log\frac{1}{\delta})$ state changes after counting N items.

We run our algorithm from above on the subsampled stream. Whenever a new counter is initialized by our algorithm, we additionally create a Morris+ counter for $\delta = 1/\text{poly}(m)$. For this counter we keep track the number of instances of that item in the original (not subsampled) stream. Let f' denote the subsampled stream. Our analysis above in the proof of Theorem 7 shows that at the $1/M$ sampling rate, with good probability, a counter for the heavy element h survives which was initialized among the first $0.7\,\|f'\|_2$ good updates. Also with high probability, this update is among the first $0.9\,\|f\|_2$ updates, by a Chernoff bound. So with at least constant probability, the Morris counter initialized in parallel gives a constant factor approximation to $\|f\|_2$. Since we can guarantee that all Morris counters are correct with high probability, we may take the maximum frequency count estimates over a serious of runs setting $M = 1, 2, 4, 8 \dots, m$. The resulting count for the heavy-hitter is at least $\frac{1}{10}\,\|f\|_2$ with constant probability. Moreover, by correctness of the Morris counters, there can only be at most $O(1)$ elements that achieve a frequency count estimate this large (since there are at most $\frac{1}{20}$-heavy items in the initial stream).

For the space bound, we run $\log m$ instances of an algorithm, each of which maintain $\log m$ counters. Each counter stores a universe identity which requires $O(\log n)$ bits. So the total space used is $O(\log^2 m \log n)$.

For the state changes, note that a Morris+ counter for f_i makes $O(\log f_i \log m)$ state changes. So the total number of state changes for one choice of p is at most $\sum_i \frac{f_i}{\|f\|_2}\log(f_i)\log m \le O(\sqrt{n}\log^2 m)$ in expectation when $p = \Theta(1/\sqrt{n})$. For choices of p, where more state changes occur, we simply truncate the algorithm early.

□

2.2 Improvements with Morris Counters

Next we observe that the state changes for this algorithm can be optimized using Morris Counters. We recall the following result:

Theorem 10. *There is an algorithm that gives a constant-factor approximation for the count of an item that occurs at most* N *times with* 9/10 *probability, using* $O(\log\log N)$ *space and* $O(\log N)$ *state changes in expectation. Moreover after* k *updates, the resulting count has expectation* k*, and variance at most* $(k/10)^2$*.*

Proof. This follows from the standard analysis of Morris counters. See for example [11, 12, 19, 21]. Note that the parameters can be changed to make the variance ck^2 for any choice of constant c. □

Lemma 11. *There is a counter algorithm that maintains an estimate of the number of instances of x under the promise that x occurs at most m times. Let Z_k be the estimate after k instances of x have occurred. The counter Z_k satisfies*

- *Simultaneously for all k $Z_k \le Rk$, with probability at least $1 - \frac{1}{5R^2}$ for $R \ge 2$.*
- *Simultaneously for all k, $Z_k \ge k/2$ with probability at least $9/10$*

Moreover this counter uses $O((\log\log m)^2)$ space and at most $O(\log m \log\log m)$ state changes.

Proof. See appendix. □

We will next analyze the following version of our counter algorithm. This is essentially the same algorithm as Algorithm 1, but the addition of line 5 makes the constants a bit more manageable. (We could also use Algorithm 1 here.)

Algorithm 2.

Accept a sampling-rate parameter p.

(1) *Maintain a list of counters L, initially empty.*
(2) *For each stream update, sample x_i at rate p. If the item is sampled start a counter for universe item x_i, and append that counter to L.*
(3) *Increment all counters in L for x_i*
(4) *Let $L = [C_1, \ldots, C_k]$. For $i = 1 \ldots k-1$ if $C_{i+1} \ge \frac{1}{2}C_i$, then remove C_i.*
(5) *If C_{i+1} and C_i are counters for the same item, then delete C_{i+1}.*
(6) *Return a frequency vector $\hat{f}$ that contains the counts C_i as estimates of their respective values. Use 0 as estimates for all other values.*

Lemma 12. *Suppose that h is a $(0.99, \ell_k)$-heavy-hitter. There is an algorithm that makes $\|f\|_k^{k-1}$ state changes in expectation, uses $O(\log m(\log n + \log m \log\log m))$ space, and outputs a frequency vector estimate $\hat{f}$, such that*

(1) *$\hat{f}_h \ge \frac{1}{10}\|f\|_k$ with probability at least 0.05*
(2) *For any fixed $x \in [n]$, and $R \ge 2$, we have $\hat{f}_x \le Rf_x$ with at least $1 - \frac{1}{25R^2}$ probability.*

Proof. We run the Algorithm 2, and use counters of Lemma 11. In addition, each time we start a counter, we create a second independent counter in parallel, only to be used at the end of the algorithm to produce approximate counts.

Set the sampling rate p in $[\frac{1}{2\|f\|_k}, \frac{1}{\|f\|_k}]$. Since $k \le 2$, note in particular that $p \le \frac{1}{\|f\|_2}$. For stream indices $i < j \in [m]$, we say that (i, j) is an *R-competing pair* for h if $x_i = h$, $x_j \ne h$, and there exists $T \ge j$ such that $f_{[j,T]}(x_j) \ge \frac{1}{R}f_{[i,T]}(h)$.

We first count the competing pairs. Note that

$$\#\{i : \exists j, x_j = k \text{ and } (i,j) \text{ is an } R\text{-competing pair}\} \le Rf_k. \tag{2}$$

To form an R-competing pair (i, j) with $x_j = z$, there are at most f_z ways to choose j, and by the above fact there are at most Rf_z ways to choose i. Summing over $z \ne h$, this implies that the number of R-competing pairs is at most

$$R\sum_{z\ne h} f_z^2 = R(\|f\|_2^2 - f_h^2) \le R(\|f\|_2^2 - \frac{99}{100}\|f\|_k^2) \le R(\|f\|_2^2 - \frac{99}{100}\|f\|_2^2) \le R(\frac{1}{100}\|f\|_2^2).$$

Now, for each heavy-hitter update i, define the sets $S_{i,r}$ to be the set of $j > i$ such that (i, j) is a 2^r-competing pair for i but not a 2^{r-1}-competing pair. Suppose that our algorithm initializes a counter at i. Recall that this counter is at least half the true count at all times simultaneously with probability at least $9/10$ by Lemma 11. For now, suppose that this event holds. If this counter is

discarded due to a counter started in $S_{i,\ell}$, then this latter counter must overestimate by a factor of at least $\frac{1}{4}2^{\ell-1} = \frac{1}{8}2^{\ell}$ since none of the items in $S_{i,\ell}$ are $2^{\ell-1}$-competing. By Lemma 11 this occurs with probability at most $\frac{8^2}{2\cdot 2^{2\ell}}$ if $\ell \geq 4$. The probability of sampling an element in $S_{i,\ell}$ is at most $p\left|S_{i,\ell}\right|$. So the probability that a counter initialized at i is discarded is at most

$$\sum_{\ell=1}^{3} p\left|S_{i,\ell}\right| + \sum_{\ell\geq 3} p\left|S_{i,\ell}\right| \frac{8^2}{5(2^\ell)^2},$$

conditioned on the counter maintaining a 2-approxiamation.

Note that $\sum_i \left|S_{i,\ell}\right|$ is bounded by the total number of 2^ℓ-competing pairs which we showed is at most $2^\ell \frac{1}{100}\|f\|_2^2$. Summing the above quantity over heavy-hitter updates i gives

$$\begin{aligned}\sum_i \left[\sum_{\ell=1}^{3} p\left|S_{i,\ell}\right| + \sum_{\ell\geq 4} p\left|S_{i,\ell}\right| \frac{8^2}{5(2^\ell)^2}\right] &= p\sum_{\ell=1}^{3}\sum_i \left|S_{i,\ell}\right| + p\sum_{\ell\geq 3}\frac{8^2}{5(2^\ell)^2}\sum_i \left|S_{i,\ell}\right| \\ &\leq p\sum_{\ell=1}^{3} 2^\ell \frac{1}{100}\|f\|_2^2 + p\sum_{\ell\geq 3}\frac{8^2}{5(2^\ell)^2} 2^\ell \frac{1}{100}\|f\|_2^2 \\ &\leq p\frac{1}{100}\|f\|_2^2 \cdot 17.\end{aligned}$$

Therefore the number of heavy-hitter updates i such that a counter initialized at i will be deleted with at least $1/2$ probability is at most $0.34p\|f\|_2^2 \leq 0.34\|f\|_2$, so there are at least $0.99\|f\|_k - 0.34\|f\|_2 \geq 0.99\|f\|_k - 0.34\|f\|_k = 0.64\|f\|_k$ such "good" updates.

Let $\mathcal{G}$ be the first $0.5\|f\|_k$ good updates. With probability at least 0.18, the first update to h that we sample is in $\mathcal{G}$. This means that the associated counter will be deleted with probability at most $\frac{1}{2} + \frac{1}{10}$ (the $1/10$ is the probability that the counter for h ever underestimates by a factor of 2). Also if this is the first counter initialized for h, then it cannot be deleted due to another counter for h as in line 5 of the algorithm. The theorem now follows by correctness of the counters, as given in Lemma 11 (recall that we use the independently started counters here to produce approximate counts).

Bounding state changes. Consider a universe item x with frequency f_x. The expected number of times that a counter is started for x is at most $\frac{f_x}{\|f\|_k}$. Moreover, the expected number of state changes to such a counter is $O(\log(f_x)\log\log(f_x)) \leq c_k f_x^{k-1}$ for a constant c_k depending only on k. So the expected number of state changes is bounded by

$$\sum_x \frac{f_x}{\|f\|_k}(c_k f_x^{k-1}) = c_k\|f\|_k^{k-1}.$$

Finally for the space, we maintain $O(\log m)$ counters each of which maintains an identity and an approximate count. This requires $O(\log m(\log n + \log m \log\log m))$ space overall.

□

2.3 Subsampling the stream

Currently our state change and space bounds depend on $\log m$ as well as on $\|f\|_k$. It turns out that we can replace these bounds with a bound depending only on n if we subsample the stream appropriately. To remove the dependence on m with will show that we can subsample the stream at an n/m rate. Then to remove the dependence on $\|f\|_k$ we will show that we can subsample at an appropriate rate to reduce the $\|f\|_k$ of the stream to $n^{1/k}$, provided that we have an approximation

of $\|f\|_k$. We will show how to obtain this approximation in the following section. We delegate the proofs in this section to the appendix.

Lemma 13. *Let f be the frequency count vector for a stream, and let g be the frequency count vector for the stream where each stream update is sampled independently with probability p.*

Suppose that p satisfies $p^{k-1} \geq \frac{\|f\|_1}{\|f\|_k^k}$ and also $p^{2k-1} \geq \frac{\|f\|_1}{\|f\|_{2k}^{2k}}$. Then with $3/4$ probability we have $\|g\|_k \leq Cp \|f\|_k$ for an absolute constant C. As a consequence, we have the following facts:

- *Suppose that $\|f\|_k = Tn^{1/k}$ and that $p \leq 1/T$. Then with $3/4$ probability, we have that $\|g_i\|_k^k \leq Cn$.*
- *Suppose that $p = n/m$. Then we $3/4$ probability, we have that $\|g\|_k \leq C\frac{n}{m} \|f\|_k$.*

We will only require a polylog approximation to F_k and this can accomplished with few state changes. The key idea is to subsample the stream, which roughly preserves the F_k of the stream up to log factors. We need only a very course bound relating the F_k of the subsampled stream to the F_k of the original stream. The next lemma shows that we can aggressively subsample a stream while maintaining its F_2 up to a polylog factor.

Lemma 14. *Let $k \in [1, 2]$. Let S be a stream of length m on a universe of n items and suppose that $m = O(n)$, let S' be a stream gotten by subsampling stream updates independently at a rate of $p = \frac{1}{\log^r n}$ where $r \geq 1$ is a constant, and let f' be the corresponding frequency count vector. Then with probability at least $1 - \delta$,*

$$\frac{1}{C \log^{2r} n} \log(1/\delta) \|f\|_k^k \leq \max(1, \|f'\|_k^k) \leq \|f\|_k^k.$$

3 Improved State Changes via. Tracking F_k.

In order to perform F_k tracking, we use the max stability property of inverse exponentials, as initially applied in [1]. The key fact that makes inverse exponentials useful for ℓ_k estimation is as follows.

Proposition 15. *Let $u_1, \ldots, u_n$ be i.i.d. from the exponential distribution e^{-x}. Then*

$$\max_i \frac{|f_i|}{u_i^{1/k}} \sim \frac{\|f\|_k}{u^{1/k}}$$

where u is exponential with parameter 1.

Proof. See [1]. □

Next we observe that for $k < 2$, after rescaling by inverse exponentials the resulting maximum is typically an $(\Omega(1), \ell_2)$-heavy-hitter. For $k = 2$, the maximum is an $(\frac{1}{\log(n)}, \ell_2)$ heavy hitter.

Lemma 16. *Let $f \in \mathbb{R}^n$ be fixed and let u_i be i.i.d. exponentials. Let $g_i = \frac{f_i}{u_i^{1/k}}$, where $k \in [1, 2]$. Then with at least $3/4$ probability,*

$$\max_i |g_i| \geq \frac{1}{\log^2 n} \|y\|_k,$$

where c_k is a constant depending on k.

Proof. See appendix. □

Lemma 17. *Suppose that f has a (α, ℓ_k)-heavy-hitter where c is a constant. There is an algorithm that gives an $O(\log m)$-multiplicative approximation to $\|f\|_\infty$ with $9/10$ probability using $O(\frac{1}{\alpha^k} \log^3 m(\log n + \log m \log\log m))$ space and $O(\frac{1}{\alpha^k}(\log^3 m)n^{1-1/k})$ state changes.*

Proof. First by hashing the universe into $1/\alpha^k$ buckets, we may reduce to the case where $\alpha = \Theta(1)$.

Suppose that we were to subsample the stream at rate $q = \Theta(n^{1/k}/\|f\|_k)$. Then by Lemma 13 the resulting stream g has $\|g\|_k \leq \Theta(n^{1/k})$. Also since the heavy-hitter occurs $\Theta(\|f\|_k)$ it occurs $\Theta(n^{1/k})$ times in g with high probability, and hence remains $\Theta(1)$-heavy. Now consider applying the algorithm of Lemma 12 to the the resulting stream g. We make one small modification: in addition to the counters already used by the algorithm, whenever a new item is sampled create an additional Morris counter that tracks the count in the original stream f rather than the subsampled stream g. Note that this does not change the overall space, and increases the state changes by a factor of at most $O(\log m)$.

If we apply a sampling rate $p \in [\frac{1}{2\|g\|_k}, \frac{1}{\|g\|_k}]$, then the algorithm of 12 will use $O(\log m(\log n + \log m \log\log m))$ space and $O(\|f\|_k^{k-1}) = O(n^{1-1/k})$ state changes. Since we do not know the correct values to choose for the sampling rates p and q, we try all $\log^2 m$ possible pairs of rates, each of the form $1/2^i$ for $i = 1, \ldots, \log m$. One of these combinations will yield a 2-approximation to $\|f\|_\infty$ with good probability using $O(\log mn^{1-1/k})$ state changes.

Now on each of the $O(\log^2 m)$ runs, we simply terminate the run if it uses more than the $\Theta((\log m)n^{1-1/k})$ state changes. Then for the runs that were not terminated, we take the maximum of the largest surviving counters for each run. The probability that any given counter for an item x, exceeds $\log^2 mf_x$ is at most $1/\log^4 n$, and only a total of $\log^3 m$ counters survive the $\log^2 m$ runs. So with failure probability at most $1/\log n$ the maximum counter is not an overestimate of $\|f\|_\infty$ by more than a $\log^2 m$ factor.

Finally we may repeat this entire procedure a constant number of times, and take the maximum over runs to boost the success probability to any desired constant. □

Lemma 18. *Suppose that our stream is of length $O(n)$. There is an algorithm that uses $O(\log^6 n \log\log n)$ space and only $O(n^{1-1/k})$ state changes, and maintains a* polylog(n) *multiplicative approximation to $\left\|f_{[0,t]}\right\|_k$ simultaneously at all times t.*

Proof. First consider subsampling the stream at a rate $\Theta(1/(\log^{10}(n)\log(m)))$ to produce a new frequency vector g. By Lemma 14 this provides a polylog(n) multiplicative approximation to $\|f\|_k$ at any point in the stream with failure probability at most $1/m^2$ say. So by taking a union bound over the stream, we obtain a polylog(n) approximation at all points in the stream simultaneously with high probability. Since we assume that the initial stream had length $O(n)$, with high probability, g has at most $O(n/\log^{10} n)$ items that occur with nonzero frequency.

It now suffices to obtain a polylog(n) approximation to $\|g\|_k$ at all times. To do this, consider rescaling g by exponentials as in Proposition 15 to produce $\tilde{g}$. By Lemma 16 the top element of $\tilde{g}$ is $1/\log n$-heavy with $3/4$ probability, and given this, we obtain a $\log n$ multiplicative approximation with $9/10$ probability by Lemma 17. We repeat this process $\Theta(\log n)$ times for different rescalings, and take the median over our estimates for the maximum entry. With failure probability at most $1/\text{poly}(n)$, the resulting median will lie between the 0.1 and 0.9 quantiles of $\frac{\|f\|_k}{u^{1/k}}$, up to a $\log n$ factor, by the max-stability property, where u is exponential. But this implies that the median is within a polylog(n) factor of $\|f\|_k$ with high probability.

Finally note that since our effective universe size is $O(n/\log^{10} n)$, we may obtain $O(n^{1-1/k})$ state changes in our application of Lemma 17. □

3.1 (ϵ, ℓ_k)-heavy-hitters

In this section we combine the various pieces to give our general heavy-hitters algorithm.

Theorem 19. *Suppose that h is $(0.999, \ell_k)$ heavy-hitter, on the universe $[n]$ for a stream of length m. There is an algorithm that makes $O(n^{1-1/k}(\log\log n)^2 \log\log\log n)$ state changes in expectation, uses $O(\log^6 n \log\log n)$ space, and outputs a frequency vector estimate $\hat{f}$, such that*

(1) $\hat{f}_h \geq \frac{1}{10} \|f\|_k$ *with probability at least* 0.05
(2) *For any fixed $x \in [n]$, and $R \geq 2$, we have $\hat{f}_x \leq R f_x$ with at least $1 - \frac{1}{25R^2}$ probability.*

Proof. First we may subsample our stream to have length $O(n)$ using Lemma 13. Let f' be the frequency count vector for the resulting stream.

This sub-sampling maintains $\Theta(1)$-heaviness with 9/10 probability. Now on the resulting stream, we apply Lemma 18 to track $\|f'\|_k$ at all times accurate to within a polylog(n) factor. Let $t_1, t_2, \ldots, t_N$ be the first times at which our estimate for $\left\|f_{[0,t_i]}\right\|_k$ is at least 2^i. Then for some constant r we know that $\left\|f_{[0,t_i]}\right\|_k$ lies in $[\frac{2^i}{\log^r n}, \log^r n 2^i]$.

We maintain the list of t_i's as follows. If L is the current estimate of the ℓ_k norm of the stream up to the current time, then we will delete all t_i such that $2 \cdot 2^{i+1} \log^r n \leq \frac{L}{\log^r n}$ and maintain all other t_i's. Note that this deletion rule, along with correctness of the estimators, implies that we only ever maintain $O(\log\log n)$ of the t_i's. Also note that we never delete t_i until we can certify that $\left\|f_{[0,t_{i+1}]}\right\|_k$ is at most half $\|f\|_k$. In particular, the smallest t_i that is maintained always satisfies $\left\|f_{[0,t_i]}\right\|_k \leq \frac{1}{2}\|f\|_k$. Let t_s be the smallest t_i at the end of the stream. We have $\left\|f_{[0,t_s]}\right\|_k \leq \frac{1}{2}\|f\|_k$. We also have that $2^s \geq \frac{\|f\|_k}{4\log^{2r}(n)}$. These two facts imply that $\|f\|_k \in [\frac{2\cdot 2^s}{\log^r n}, 4 \cdot 2^s \log^{2r} n]$.

At each time t_i we start $\log\log n$ new instances of the heavy-hitter algorithm from Lemma 12 using sampling rates p spaced geometrically in powers of 2 in this interval $[\frac{2\cdot 2^s}{\log^r n}, 4 \cdot 2^s \log^{2r} n]$ When a time t_i is deleted, we also delete the corresponding data from these runs of the algorithm. For technical reasons to be described below, we actually run $\Theta(\log\log\log n)$ instances of this algorithm and then take the 0.02 quantile of the resulting frequency estimates. Note that one of these values of p is guaranteed to be a 2-approximation to $\|f\|_k$ at the end of the algorithm, by construction. This also only require $O(\log\log n)$ instances of the algorithm per active t_i.

Note that a 0.999-heavy-hitter of f' will be a 0.99 heavy-hitter of the stream beginning at t_s, since $\left\|f_{[0,t_s]}\right\|_k \leq \frac{1}{2}\|f\|_k$. So one of the algorithm instances started at t_s will yield a frequency estimate for h that is at least $\frac{1}{10}\|f\|_k$ with good probability.

To get our estimate for the frequency counts we take the 0.02 quantile of each frequency estimate over the $\Theta(\log\log\log n)$ runs of the algorithm. Then we take the maximum over the runs corresponding to the $O(\log\log n)$ choices of p. The probability that any give run overestimates by a factor of R is then at most $\frac{1}{25R^2} \cdot \frac{1}{\log\log n}$, so for fixed x, we have the desired upper bound by a union bound. The lower bound on the estimate of the frequency of h follows from the correctness of Lemma 12.

□

Theorem 20. *For $k \in [1, 2]$, there is an algorithm that solves the (ϵ, ℓ_k)-heavy-hitters problem using*
$O(\frac{1}{\epsilon^2}(\log\frac{1}{\epsilon})\log^6 n \log\log n)$ space and $O(\frac{1}{\epsilon}(\log\frac{1}{\epsilon})n^{1-1/k}\operatorname{poly}\log\log n)$ state changes.

Proof. Using a standard idea as in [5] for example, we reduce to the $O(1)$-heavy-hitters problem. We use a pairwise independent hash function to sample a subset of the universe $[n]$ into $\Theta(\frac{1}{\epsilon^k})$ subsets of size $\Theta(\epsilon^k n)$. If h is an (ϵ, ℓ_k)-heavy-hitter and the constant are chosen appropriately, then with probability at least ϵ^{-k}, h is a $(0.99, \ell_k)$ heavy-hitter of its set with at least 1/2 probability.

We then run our $\Omega(1)$-heavy-hitter algorithm of Theorem 19 on each of the subsets of $[n]$. Since each subset has size $\Theta(\epsilon^k n)$, this requires $O((\epsilon^k n)^{1-1/k})$ state changes per subset, and therefore $O(\frac{1}{\epsilon} n^{1-1/k})$ state changes overall.

We repeat this entire procedure of hashing and running $O(1)$-heavy-hitter $O(\log \frac{1}{\epsilon})$ times. With probability at least $1/100$ each heavy-hitter h is given a count estimate of at least $\frac{\epsilon}{10} \|f\|_k$ by Theorem 19. Therefore by a Chernoff bound, if we run the procedure $N = \Theta(\log \frac{1}{\epsilon})$ times, h will receive a count estimate of at least $\frac{1}{10} \|f\|_k$ on at least $\frac{1}{200} N$ runs with probability at least $1 - \frac{1}{10}\epsilon^k$.

Now we argue that not many items can achieve such a large count estimate on a constant fraction of runs. Suppose that x is a universe item that occurs f_x times. By Theorem 19, the probability that the count estimate for $\hat{f}_x$ for x is at least $\frac{\epsilon}{10} \|f\|_k$ is at most $\frac{100 f_x^2}{\epsilon^2 \|f\|_k^2} := p_x$. Then the probability that this x occurs with at least this count on at least $\frac{1}{200} N$ runs is at most p_x for if $p_x < 1/400$ for N a large enough constant.

So the expected number of items that achieve a count of at least $\frac{\epsilon}{10} \|f\|_k$ on $\frac{1}{200} N$ runs is at most

$$\left(\sum_{x : p_x \geq \frac{1}{400}} 1 + \sum_{x : p_x < \frac{1}{400}} p_x \right) \leq \#\{x : p_x \geq \frac{1}{400}\} + \sum_{x : f_x \leq c\epsilon \|f\|_k} \frac{100 f_x^2}{\epsilon^2 \|f\|_k^2}$$

The first term is at most $O(1/\epsilon^k)$ since the set consists of $\Omega(\epsilon)$-heavy-hitters. For the second term, we have the quantities f_x, each bounded by $c\epsilon \|f\|_k$ and whose kth powers sum to at most $\|f\|_k^k$. To bound the sum we wish to maximize $\sum f_x^2$ subject to these constraints. Using the interpolation inequality from the proof of Proposition 25 we have that in general $\|v\|_2^2 \leq \|v\|_k^k \|v\|_\infty^{2-k}$. So in our situation, $\sum f_x^2 \leq \|f\|_k^k (c\epsilon \|f\|_k)^{2-k}$, which implies that the second sum above is bounded by $O(\epsilon^{-k})$. This means that it suffices for us to return the largest $O(\epsilon^{-k})$ frequency count estimates. □

4 Lower Bound

In this section we present a simple lower bound for the ϵ-heavy-hitters problem.

Theorem 21. *Let $k \in [1, \infty)$ Suppose that an algorithm solves the (ϵ, ℓ_k)-heavy-hitter problem with probability at least $3/4$. Then this algorithm must use at least $\Omega(n^{1-1/k} \frac{1}{\epsilon} \log \frac{1}{\epsilon})$ state changes.*

Proof. We proceed in a series of steps.

High probability lower bound for O(1)-heavy-hitters. We take our stream to have as updates $1, 2, 3, \ldots, n$ in that order. For a single item h chosen uniformly at random we repeat that item $n^{1/k}$ times when it occurs. By construction, h is then an $O(1)$-heavy-hitter. We call this stream $\mathcal{S}$.

Now fix an algorithm $\mathcal{A}$ that makes at most T state changes. Consider running $\mathcal{A}$ on the stream $\mathcal{S}'$ which is $1, 2, \ldots, n$ with no item repeated. At the item i is seen, the internal state of $\mathcal{A}$ will dictate a probability p_i that $\mathcal{A}$ changes state on item i. (Note that p_i is itself a random variable.) Note that expected number of state changes is $\mathrm{E}(p_1) + \ldots + \mathrm{E}(p_n) \leq T$. This means that $\mathrm{E}(p_i) \leq 10T/n$ for at least $(9/10)n$ values i. So with probability $9/10$ we have that $\mathrm{E}(p_h) \leq 10T/n$, which implies that $p_h \leq 40T/n$ with probability at least $1/2$. This means that with probability at least

$$\frac{1}{2}(1 - \frac{40T}{n})^{n^{1/k}},$$

$\mathcal{A}$ does not change state on h. So if $\mathcal{A}$ distinguishes $\mathcal{S}$ from $\mathcal{S}'$ with probability $1 - \delta$ then we must have

$$(1 - \frac{40T}{n})^{n^{1/k}} \leq 2\delta,$$

or equivalently $n^{1/k}\log(1-\frac{40T}{n}) \leq \log(2\delta)$, which after bounding $\log(1-x)$ by a first order approximation implies that $T \geq \Omega(n^{1-1/k}\log\frac{1}{\delta})$ for large enough n.

Lower bound for identifying a heavy-hitter. With the same setup as above, suppose that $\mathcal{A}$ is an algorithm that runs on $\mathcal{S}$ and outputs a set H of size at most R, which contains the heavy-hitter with probability at least $1-\delta$. Then let $\mathcal{S}_{h_1}$ and $\mathcal{S}_{h_2}$ be the streams with h_1 and h_2 respectively repeated as above. Such a $\mathcal{A}$ could distinguish between $\mathcal{S}_{h_1}$ and $\mathcal{S}_{h_2}$ with probability at least $1-R/n-\delta$. This means that $\mathcal{A}$ must distinguish at least one of the two streams from $\mathcal{S}'$ with failure probability at most $\frac{1}{2}(R/n+\delta)$. So $\mathcal{A}$ must make $\Omega(n^{1-1/k}\log(\frac{R}{n}+\delta)^{-1})$ state changes.

Extending to ϵ-heavy-hitters. Now divide the universe into $R := 1/\epsilon^k$ equally sized pieces, each of size $\epsilon^k n$. Construct streams $\mathcal{P}_1,\ldots,\mathcal{P}_r$ where in each stream $\mathcal{P}_i$, one item i is chosen to repeat $(\epsilon^k n)^{1/k} = \epsilon n^{1/k}$ times precisely as above and concatenate them. Let $h_1,\ldots,h_r$ be the items chosen to be heavy on each stream $\mathcal{P}_i$. Note that the final output to the heavy-hitters problem must have size at most $c\epsilon^{-k}$. In particular the intersection of the output with each sub-universe must have size at most $O(c\epsilon^{-k})$. So we relax the problem and allow our algorithm to output $O(c\epsilon^{-k})$ items in each sub-universe. To be correct it must output the heavy item in each $\mathcal{P}_i$. It is clear that a correct heavy-hitter algorithm could solve this problem. For $n \geq 1/\epsilon^4$ say, the previous paragraph shows that we must make $\Omega((\epsilon^k n)^{1-1/k}\log(1/\epsilon))$ state changes to succeed on stream $\mathcal{P}_i$ with failure probability at most $\epsilon/10$. But a correct algorithm cannot fail with probability $\epsilon/10$ on more than half of the stream, so on all streams together our algorithm requires $\Omega(\epsilon^{-k}(\epsilon^k n)^{1-1/k}\log(1/\epsilon))$ state changes as claimed.

□

References

[1] Alexandr Andoni. 2017. High frequency moments via max-stability. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 6364–6368.

[2] 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.

[3] Avraham Ben-Aroya and Sivan Toledo. 2006. Competitive analysis of flash-memory algorithms. In *European Symposium on Algorithms*. Springer, 100–111.

[4] S Boucheron, G Lugosi, and P Massart. 2013. *Inequalities. a nonasymptotic theory of independence.* Oxford University Press.

[5] Vladimir Braverman, Stephen R Chestnut, Nikita Ivkin, Jelani Nelson, Zhengyu Wang, and David P Woodruff. 2017. Bptree: An 2 heavy hitters algorithm using constant memory. In *Proceedings of the 36th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*. 361–376.

[6] Vladimir Braverman and Rafail Ostrovsky. 2013. Approximating large frequency moments with pick-and-drop sampling. In *International Workshop on Approximation Algorithms for Combinatorial Optimization*. Springer, 42–57.

[7] 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*. 1126–1130.

[8] 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 annual Design Automation Conference*. 212–217.

[9] Moses Charikar, Kevin Chen, and Martin Farach-Colton. 2002. Finding frequent items in data streams. In *International Colloquium on Automata, Languages, and Programming*. Springer, 693–703.

[10] Kenneth L Clarkson and David P Woodruff. 2014. Sketching for M-estimators: A unified approach to robust regression. In *Proceedings of the twenty-sixth annual ACM-SIAM symposium on Discrete algorithms*. SIAM, 921–939.

[11] Philippe Flajolet. 1985. Approximate counting: a detailed analysis. *BIT Numerical Mathematics* 25, 1 (1985), 113–134.

[12] André Gronemeier and Martin Sauerhoff. 2009. Applying approximate counting for computing the frequency moments of long data streams. *Theory of Computing Systems* 44 (2009), 332–348.

[13] Piotr Indyk and David Woodruff. 2005. Optimal approximations of the frequency moments of data streams. In *Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*. 202–208.

[14] Rajesh Jayaram, David P Woodruff, and Samson Zhou. 2024. Streaming Algorithms with Few State Changes. *Proceedings of the ACM on Management of Data* 2, 2 (2024), 1–28.

[15] Daniel M Kane, Jelani Nelson, Ely Porat, and David P Woodruff. 2011. Fast moment estimation in data streams in optimal space. In *Proceedings of the forty-third annual ACM symposium on Theory of computing*. 745–754.

[16] Kasper Green Larsen, Jelani Nelson, Huy L Nguyn, and Mikkel Thorup. 2019. Heavy hitters via cluster-preserving clustering. *Commun. ACM* 62, 8 (2019), 95–100.

[17] Yi Li, David Woodruff, and Taisuke Yasuda. 2021. Exponentially improved dimensionality reduction for l1: Subspace embeddings and independence testing. In *Conference on Learning Theory*. PMLR, 3111–3195.

[18] Jayadev Misra and David Gries. 1982. Finding repeated elements. *Science of computer programming* 2, 2 (1982), 143–152.

[19] Robert Morris. 1978. Counting large numbers of events in small registers. *Commun. ACM* 21, 10 (1978), 840–842.

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

[21] Jelani Nelson and Huacheng Yu. 2022. Optimal bounds for approximate counting. In *Proceedings of the 41st ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*. 119–127.

A Extension to $k > 2$.

To extend our heavy-hitters result to $k > 2$, we can apply exactly the argument for $k < 2$, but need better concentration for the Morris counters. In particular the techiniques of [21] effectively show that Morris counters are extremely unlikely to overestimate by a large factor R.

Lemma 22. *Let X denote the value of a Morris counter after N updates, and let $\widehat{N} = 2^X - 1$ be its estimate of N. Then, for any $R \geq 2$, there exists an absolute constant $c > 0$ such that*

$$\mathbb{P}[\widehat{N} \geq RN] \leq \exp(-cR).$$

Proof. To reach level $k = \log_2(RN + 1)$, the Morris counter must process a random number of updates given by the sum

$$T_k = \sum_{i=0}^{k-1} G_i,$$

where $G_i \sim \text{Geom}(p_i)$ with $p_i = 2^{-i}$. Each G_i is independent and satisfies $\mathbb{E}[G_i] = 2^i$, so the expected number of updates to reach level k is

$$\mathbb{E}[T_k] = \sum_{i=0}^{k-1} 2^i = 2^k - 1 = RN.$$

We seek to bound $\mathbb{P}[T_k \leq N] = \mathbb{P}[T_k - \mathbb{E}[T_k] \leq -(R-1)N]$. Note that each G_i is sub-gamma[2] with variance proxy $v_i = \text{Var}(G_i) \leq 2^{2i}$ and scale parameter $b_i \leq 2^i$. The sum T_k is then sub-gamma with parameters

$$v = \sum_{i=0}^{k-1} v_i \leq \sum_{i=0}^{k-1} 2^{2i} \leq \frac{2^{2k}}{3} = O(R^2N^2), \quad b = \max_i b_i \leq 2^{k-1} = O(RN).$$

Applying the sub-gamma lower-tail bound, for $t = (R-1)N \leq v/b$, we have:

$$\mathbb{P}[T_k \leq \mathbb{E}[T_k] - t] \leq \exp\left(-\frac{t^2}{2v}\right) \leq \exp\left(-\frac{(R-1)^2N^2}{2v}\right) \leq \exp(-cR)$$

for some constant $c > 0$. Hence,

$$\mathbb{P}[\widehat{N} \geq RN] = \mathbb{P}[X \geq \log_2(RN+1)] = \mathbb{P}[T_k \leq N] \leq \exp(-cR),$$

as claimed □

This gives the following version of Theorem 19. The proof is identical, but we use the stronger Morris counter guarantee for (2).

Theorem 23. *Suppose that h is $(0.999, \ell_k)$ heavy-hitter, on the universe $[n]$ for a stream of length m. There is an algorithm that makes $O(n^{1-1/k}(\log\log n)^2 \log\log\log n)$ state changes in expectation, uses $O(\log^6 n \log\log n)$ space, and outputs a frequency vector estimate $\hat{f}$, such that*

(1) *$\hat{f}_h \geq \frac{1}{10}\|f\|_k$ with probability at least 0.05*
(2) *For any fixed $x \in [n]$, and $R \geq 2$, we have $\hat{f}_x \leq Rf_x$ with at least $1 - \exp(-cR)$ probability.*

Next we give our ℓ_k heavy-hitter result for $k \geq 2$.

Theorem 24. *For $k \geq 2$, there is an algorithm that solves the (ϵ, ℓ_k)-heavy-hitters problem using $\tilde{O}(\frac{1}{\epsilon^2}n^{1-2/k})$ space and $O(\frac{1}{\epsilon}\log\frac{1}{\epsilon}n^{1-1/k}\,\text{poly}\log\log n)$ state changes.*

[2]See [4] for background on subgamma random variables.

Proof. First observe that for $k \geq 2$, an (ϵ, ℓ_k)-heavy-hitter is always an $(\epsilon n^{1/k-1/2}, \ell_2)$-heavy-hitter. Let $\epsilon' = \epsilon n^{1/k-1/2}$. Now we run the argument of Theorem 20 to identify the (ϵ', ℓ_2)-heavy-hitters. This set may be too large, but we now have better count estimates by the above result. In particular, we wish to bound the number of items with count estimates at least $\frac{\epsilon}{10} \|f\|_k$.

We follow the last paragraph in the proof of Theorem 20. say that i is a small index if $f_i \leq \frac{\epsilon}{20} \|f\|_k$. observe that the probability that the frequency of a small index f_i is estimated to be at least $\frac{\epsilon}{10} \|f\|_k$ is at most $\exp(-c\frac{\epsilon \|f\|_k}{f_i}) \leq O(\frac{f_i^k}{\epsilon^k \|f\|_k^k})$. So the expected number of small indices with a count estimate this large is at most $O(\frac{1}{\epsilon^k})$. There are at most $O(\frac{1}{\epsilon^k})$ heavy hitters, and so $O(\frac{1}{\epsilon^k})$ count estimates larger than $\frac{\epsilon}{10} \|f\|_k$.

The rest of the proof of Theorem 20 applies to show that all the heavy-hitters are found and receive a count estimate of at least $\Omega(\epsilon' \|f\|_2) \geq \Omega(\epsilon \|f\|_k)$. It therefore suffices to select the largest $O(1/\epsilon^k)$ count estimates from among the (ϵ', ℓ_2)-heavy-hitters. □

B Proofs of subsampling lemmas

We need a simple bound for the fractional moments of a binomial random variable.

Proposition 25. *Let X be Binomial(n, p) and let $r \in [1, 4]$ Then*

$$\mathrm{E}(X^r) \leq C(np + (pn)^r).$$

Recall the following interpolation inequality for norms for $1 \leq q_1 \leq r \leq q_2$:

$$\|f\|_r \leq \|f\|_{q_1}^{1-\theta} \|f\|_{q_2}^{\theta}$$

where $\frac{1}{r} = \frac{\theta}{q_2} + \frac{1-\theta}{q_1}$.

By the moments of a binomial random variable, we have $\mathrm{E}(X) = Cpn$ and $\mathrm{E}(X^4) = O(np + (np)^4)$. Now consider two cases.

If $np \leq 1$ then $\mathrm{E}(X) = O(np)$ and $\mathrm{E}(X^4)^{1/4}$ is $O((np)^{1/4})$ and so by the interpolation inequality with $q_1 = 1$ and $q_2 = 4$, we have

$$\mathrm{E}(X^r)^{1/r} \leq O((np)^{1-\theta} + (np)^{\theta/4}) = O((np)^{1-\frac{3}{4}\theta}) = O((np)^{1/r}).$$

Otherwise if $np \geq 1$, then $\mathrm{E}(X)$ and $\mathrm{E}(X^4)^{1/4}$ are both $O(pn)$ and so $\mathrm{E}(X^r)^{1/r}$ is $O(np)$ as well by the interpolation inequality.

Proof of Lemma 13. By applying Proposition 25, we have the bound

$$\begin{aligned}
\mathrm{E}(\|g\|_k^{2k}) &= \mathrm{Var}((g_1^k + \ldots + g_n^k)) + \mathrm{E}(g_1^k + \ldots + g_n^k)^2 \\
&= \mathrm{Var}(g_1^k) + \ldots + \mathrm{Var}(g_n^k) + \left(\mathrm{E}(g_1^k) + \ldots + \mathrm{E}(g_n^k)\right)^2 \\
&\leq \mathrm{E}(g_1^{2k}) + \ldots + \mathrm{E}(g_n^{2k}) + \left(\mathrm{E}(g_1^k) + \ldots + \mathrm{E}(g_n^k)\right)^2 \\
&\leq \sum_i C(pf_i + (pf_i)^{2k}) + \left(\sum_i C(pf_i + (pf_i)^k)\right)^2 \\
&= C(p\|f\|_1 + p^{2k}\|f\|_{2k}^{2k}) + C^2(p\|f\|_1 + p^k\|f\|_k^k)^2.
\end{aligned}$$

The latter term within each pair of parentheses is larger by the hypothesis on p, and so the first part of the claim follows from Markov.

For the first bullet point, we verify the two inequalities for $p = 1/T$ (note that choosing p smaller only decreases $\|g\|_k$). Then we have $\|f\|_1 \leq n^{1-1/k}\|f\|_k = Tn$, which rearranges to $\frac{\|f\|_1}{T^k n} \leq \frac{1}{T^{k-1}}$.

Similarly, we have

$$\|f\|_1 \le n^{1-1/k} \|f\|_k \le n^{2-1/k} \|f\|_k = n^{2-1/k} \frac{\|f\|_k^{2k}}{\|f\|_k^{2k-1}} \le n^{2-1/k} \frac{\|f\|_{2k}^{2k}}{\|f\|_k^{2k-1}} = p^{2k-1} \|f\|_{2k}^{2k}.$$

For the second bullet point, suppose that $p = \frac{n}{m} = \frac{n}{\|f\|_1}$. We verify the two inequality. First recall that $\|f\|_1 \le n^{1-1/k} \|f\|_k$ which rearranges to $\frac{\|f\|_1}{\|f\|_k^k} \le \frac{n^{k-1}}{\|f\|_1^{k-1}} = p^{k-1}$. Similarly, $\|f\|_1 \le n^{1-1/2k} \|f\|_{2k}$, which rearranges to $\frac{\|f\|_1}{\|f\|_{2k}^{2k}} \le \frac{n^{2k-1}}{\|f\|_1^{2k-1}} = p^{2k-1}$.

We use the following version of the Chernoff bound.

Proposition 26. *Let $X_1, \ldots, X_n$ be i.i.d Bernoulli(p) random variables. Let $X = X_1 + \ldots + X_n$. Then for $\delta \in (0,1)$,*

$$\Pr(X \le (1-\delta)pn) \le \exp(-\delta^2 pn/2).$$

Proof of Lemma 14. First divide the universe $[n]$ into dyadic level sets $S_0, S_1, \ldots$ according to frequency counts, where S_i contains the universe items with frequency count in $[2^i, 2^{i+1})$.

Note that

$$\sum_i |S_i| \, 2^{ki} \ge \|f\|_k^k.$$

Since all frequency counts are bounded by m, for some i, we have $|S_i| \, 2^{ki} \ge \frac{1}{C \log m} \|f\|_k^k$.

Next we show that that a reasonable fraction of the mass of this level set is preserved with high probability. Let j be a universe item in S_i. Then j occurs with frequency at least 2^i. Sampling at a rate p, we have by Proposition 26 that

$$\Pr(f_j' \le (p/2) f_j) \le \exp(-p f_j/8) \le \exp(-p 2^i/8) := 1 - q.$$

Now we consider two cases. First suppose that $|S_i| \le M$, where we set $M = 8 \log^r(n) \log(1/\delta)$. Then $2^i \ge \frac{1}{(CM \log n)^{1/k}} \|f\|_k$.

So

$$\begin{aligned}\Pr(f_j' \le (p/2) f_j) &\le \exp(-p \frac{1}{(CM \log n)^{1/k}} \|f\|_k / 8) \\ &\le \exp(-\frac{1}{8(CM)^{1/k} \log^{1/k+r}(n)} \|f\|_k)\end{aligned}$$

As long as $\|f\|_k \ge C \log^{2r}(n) \log(1/\delta)$ this probability is bounded by $\frac{\delta}{n}$. Then taking a union bound over the elements in S_i, we have that $f_j' \ge (p/2) f_j$ for all $j \in S_i$ with failure probability at most δ. This implies that

$$\|f'\|_k^k \ge \sum_{j \in S_i} (p/2)^2 f_j^2 \ge |S_i| \, (p/2)^k 2^{ki} \ge \frac{1}{4C \log^{r+1} n} \|f\|_k^k.$$

For the second case, suppose that $|S_i| \ge M$. Say that an item j in S_i is good if $f_j' \ge (p/2) f_j$. Note that j is good with probability at least p.

Again by a Chernoff bound, the number of good items is at least $(p/2) \, |S_i|$ with failure probability at most

$$\exp(-p \, |S_i| / 8) \le \exp(-pM/8) \le \exp(-\frac{1}{\log^r n} M/8) \le \delta,$$

by our choice of M. Conditioned on the number of good items being at least $(p/2) \, |S_i|$, we have that

$$\|f'\|_k^k \ge (p/2) \, |S_i| \, 2^{ik} \ge (p/2) \frac{1}{C \log n} \|f\|_k^k = \frac{1}{C \log^{r+1} n} \|f\|_k^k.$$

In either case, we showed that

$$\|f'\|_k^k \geq \frac{1}{4C\log^{r+1}(n)}\|f\|_k^k,$$

with failure probability at most δ, provided that $\|f\|_k^k \geq C\log^{2r}(n)\log(1/\delta)$. Finally if $\|f\|_k^k$ is smaller than this value, then we obtain the desired bound immediately.

C Proof of Lemma 16

We first bound $\|g\|_k^k$. Note that

$$\Pr(|g_i|^k \geq \frac{\|f\|_k^k}{\ell^k}) = 1 - \exp(-|f_i|^k\,\ell^k/\|f\|_k^k) \leq |f_i|^k\,\ell^k/\|f\|_k^k.$$

Therefore

$$\mathrm{E}\left[\#\{i : |g_i| \geq \frac{\|f\|_k}{\ell}\}\right] \leq \ell^k.$$

let $\mathcal{E}_r$ be the event that there are most $(10\log n)2^{kr}$ entries in g larger than $2^{-r}\|f\|_k$. By Markov, $\mathcal{E}_r$ fails to hold with probability at most $\frac{1}{10\log n}$. The probability that at least one of the events $\mathcal{E}_r$ does not hold is at most $1/10$ by a union bound over all r.

So with probability at least $9/10$, all of the events $\mathcal{E}_r$ hold. Suppose that all events $\mathcal{E}_r$ hold. Let $M = \|g\|_\infty$. Then we have the bound

$$\begin{aligned}
\|g\|_k^k &\leq M^k \cdot \#\{i : g_i \geq 2^{-1}\|f\|_k\} + \sum_{r>1}\#\{i : g_i \geq 2^{-r}\|f\|_k\}(2^{-r+1}\|f\|_k)^k \\
&\leq M^k(10\log n 2^k) + \sum_{r>1}(10\log n 2^{kr})(2^{-r+1}\|f\|_k)^k \\
&\leq M^k(10\log n 2^k) + 10\cdot 2^k(\log^2 n)\|f\|_k^k,
\end{aligned}$$

after summing over $\log n$ terms. Recall that $M^k/\|f\|_k^k \sim 1/u$ where u is exponentially distributed. So $u \leq 3$ with probability at least 0.95, and hence $M \geq (1/3)^{1/k}\|f\|_k \geq (1/3)\|f\|_k$ with at least 0.95 probability. This implies that $\|g\|_2^2 \leq c_k\|f\|_k^2$ as desired.

D Morris Counter Results

Proof of Lemma 11. Let Y_k be the value of a Morris counter from Theorem 10 after k increments. Recall that $\mathrm{E}(Y_k) = k$ and $\mathrm{Var}(Y_k) \leq (k/10)^2$. By Chebyshev's inequality,

$$\Pr(Y_k \geq Rk) \leq \Pr(|Y_k - k| \geq (R-1)k) \leq \frac{1}{100(R-1)^2}.$$

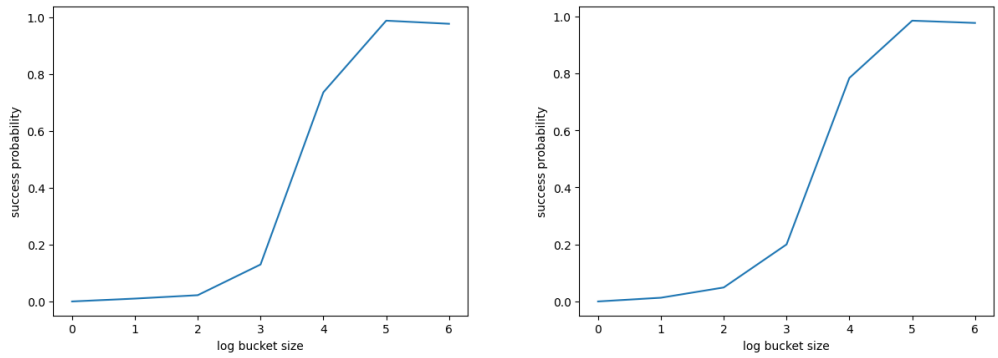
We run $O(\log\log m)$ instances of the Morris Counter in parallel and use the median count as our estimate, which we call Z_k. Then for fixed k and $R \geq 2$ we have,

$$\Pr(Z_k \geq Rk) \leq \frac{1}{100\log_2 m}\frac{1}{(R-1)^2},$$

after choosing the hidden constant appropriately. By a union bound, this implies that $Z_k < Rk$ holds at $k = \lceil m/2^i\rceil$ for all i with failure probability at most $\frac{1}{100(R-1)^2}$. Since Z_k is non-decreasing, this means that

$$\Pr(\exists k, Z_k \geq 2Rk) \leq \frac{1}{100}\frac{1}{(R-1)^2} \leq \frac{1}{25R^2},$$

for $R \geq 2$.



(a) (b)

Fig. 1. Behavior of CountSketch for varying buckets and $\lfloor \log_2 n \rfloor$ repetitions.

Also by Chebyshev, $\Pr(Y_k \leq (3/4)k) \leq \frac{1}{5}$. So we have that $\Pr(Z_k \leq (3/4)k) \leq \frac{1}{10 \log_4 m}$. Again union bounding over a geometric set of updates starting at 1, we obtain $Z_k \geq k/2$ for all k with probability at least $9/10$.

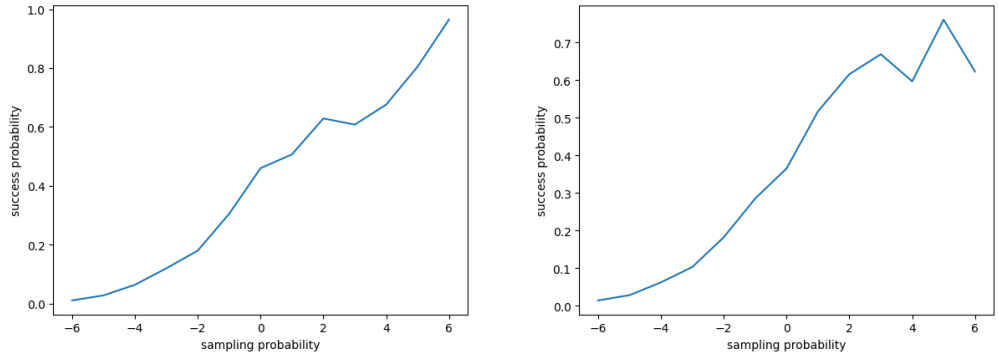
E Experiments

We conduct preliminary experiments which suggest that our algorithm may yield reasonable constants in practice.

In each experiment we take a stream and universe of length 10000. We consider two distributions on stream, (a) and (b). For distribution (a) each item of the stream is the heavy item with $\frac{1}{\sqrt{n}}$, and is otherwise a uniformly random item. For distribution (b) we have $\sqrt{n}$ heavy items, and then 3 other items that occur $\sqrt{n}/2$ times $\sqrt{n}/4$ times and $\sqrt{n}/8$ times. The remaining items all occur once. An experiment is successful the algorithm correctly identifies the heavy hitter. Each datapoint in our graphs is the fraction of successes over 1000 independent trials.

In Figure 1 we show the success probability of CountSketch for varying numbers of buckets, and a fixed number of repetitions. In Figure 2 we show the performance of our counter-based Algorithm 1 for various sampling rate parameters p. Since CountSketch is being repeated 13 times in each run, it uses comparable space to our datastructure on these datasets. Indeed for obtaining probabilities near $1/2$, our data structure appears to do slightly better. We note that we could also boost our data structure by repeating it to achieve higher failure probability. While these input distributions are quite simple, they suggest that our algorithm may in practice be comparable to CountSketch, at least in certain regimes. We leave a more detailed empirical study to future work.

Received June 2025; accepted August 2025



(a) (b)

Fig. 2. Behavior of our counter-based algorithm for various sampling rates. A sampling rate of $q/\sqrt{n}$ is recorded as $\log q$ on the x-axis.