The Coin Problem with Applications to Data Streams

Mark Braverman[*] Princeton

Sumegha Garg[†] Harvard

David P. Woodruff[‡] CMU

Abstract

Consider the problem of computing the majority of a stream of n i.i.d. uniformly random bits. This problem, known as the *coin problem*, is central to a number of counting problems in different data stream models. We show that any streaming algorithm for solving this problem with large constant advantage must use $\Omega(\log n)$ bits of space. We extend our lower bound to proving tight lower bounds for solving multiple, randomly interleaved copies of the coin problem, as well as for solving the OR of multiple copies of a variant of the coin problem. Our proofs involve new measures of information complexity that are well-suited for data streams.

We use these lower bounds to obtain a number of new results for data streams. In each case there is an underlying d-dimensional vector x with additive updates to its coordinates given in a stream of length m. The input streams arising from our coin lower bound have nice distributional properties, and consequently for many problems for which we only had lower bounds in general turnstile streams, we now obtain the same lower bounds in more natural models, such as the bounded deletion model, in which $\|x\|_2$ never drops by a constant fraction of what it was earlier, or in the random order model, in which the updates are ordered randomly. In particular, in the bounded deletion model, we obtain nearly tight lower bounds for approximating $\|x\|_\infty$ up to additive error $\frac{1}{\sqrt{k}}\|x\|_2$, approximating $\|x\|_2$ up to a multiplicative $(1+\epsilon)$ factor (resolving a question of Jayaram and Woodruff in PODS 2018), and solving the Point Query and ℓ_2-Heavy Hitters Problems. In the random order model, we also obtain new lower bounds for the Point Query and ℓ_2-Heavy Hitters Problems. We also give new algorithms complementing our lower bounds and illustrating the tightness of the models we consider, including an algorithm for approximating $\|x\|_\infty$ up to additive error $\frac{1}{\sqrt{k}}\|x\|_2$ in turnstile streams (resolving a question of Cormode in a 2006 IITK Workshop), and an algorithm for finding ℓ_2-heavy hitters in randomly ordered insertion streams (which for random order streams, resolves a question of Nelson in a 2018 Warwick Workshop).

[*] Email: mbraverm@cs.princeton.edu. Research supported in part by the NSF Alan T. Waterman Award, Grant No. 1933331, a Packard Fellowship in Science and Engineering, and the Simons Collaboration on Algorithms and Geometry.

[†] Email: sumegha.garg@gmail.com. This work was completed when Sumegha was at Princeton

[‡] Email: dwoodruf@andrew.cmu.edu. Research supported in part by the National Science Foundation under Grant No. CCF-1815840.

1 Introduction

The data stream model is an important model for processing sequences of data that are too large to be stored in memory. Common examples of data streams include search logs, scientific data, sensor networks, and network traffic. In many of these applications, an algorithm is only allowed one pass over the stream and must store a short summary of what it has seen in order to answer a pre-defined query. We refer the reader to several surveys on data streams [17, 30, 31].

The arguably most fundamental problem in data streams is maintaining a counter C. Formally, suppose we are given a sequence of at most n updates of the form $C \leftarrow C + 1$ in an arbitrary order and we are allowed one pass over this sequence. While one can compute the exact count C using $\log_2 n$ bits of space, one can use significantly less space to compute a constant factor multiplicative approximation with constant probability; indeed, this is possible using only $O(\log \log n)$ bits of space using a probabilistic counter [28]. However, suppose now we see a stream of both positive and negative updates of the form $C \leftarrow C + 1$ or $C \leftarrow C - 1$. In this case $\Omega(\log n)$ bits of space are needed even to compute a constant factor approximation with constant probability [24].

But perhaps surprisingly, suppose now we see a uniformly random sequence of n i.i.d. updates of the form $C \leftarrow C + 1$ or $C \leftarrow C - 1$, meaning each update is independently $C \leftarrow C + 1$ with probability $1/2$ and $C \leftarrow C - 1$ with probability $1/2$. The following question is open:

> How much space is needed to compute a constant factor approximation to C w.h.p. for a uniformly random sequence of n i.i.d. updates of the form $C \leftarrow C + 1$ or $C \leftarrow C - 1$ (each with probability $1/2$)?

This problem is also known as the *coin problem* and has been studied in the context of branching program and circuit complexity [8, 33, 14, 25], though such results only give an $\Omega(\log \log n)$ space lower bound for our problem. This is far from the best known upper bound, which is $O(\log n)$ bits.

Despite the large body of work on data streams, and the simplicity of this question, why is it still open? To understand this, we look at common techniques for proving lower bounds in data streams. Arguably the most common is communication complexity, and in particular, 2-player communication complexity. We (typically) give half of the stream to Alice, and half of the stream to Bob, and Alice runs the streaming algorithm on her input. She then transmits the state of her algorithm to Bob, who continues the execution of the streaming algorithm on his input. If the output of the streaming algorithm can be used to solve the communication game, then the amount of memory of the streaming algorithm must be at least the 1-way communication of the game. Unfortunately, this approach fails for the above problem. Indeed, with large probability the count on Alice's stream will be $C\sqrt{n}$ in absolute value, for a constant $C > 0$, and it suffices for Alice to just send $O(1)$ bits to round the value of C to a smaller constant. Then Bob adds this to the count of his stream, which is also random, and obtains a constant factor approximation with large constant probability. The issue is the random order of the stream makes this problem too easy.

When 2-player communication complexity fails, one often resorts to multiplayer communication complexity to prove lower bounds. We again give each player a portion of the stream and perform the reduction above, passing the state of the streaming algorithm from one player to the next. However, when lower bounding the communication for the purposes of streaming, existing techniques have only looked at the blackboard or coordinator models of communication. In the blackboard model, the message sent from one player to the next is seen by all future players. For the question above, suppose we arbitrarily partition the stream into pieces and give a piece to each player. In this case, it is known that if each player randomly rounds their own input count so that the expectation is their actual count, then one can obtain a constant factor approximation to the overall count from these rounded counts with large probability [22]. This $O(\log \log n)$ bit upper bound is a simultaneous protocol (since each player's message does not depend on prior messages) and thus also holds in the so-called coordinator model. In fact, an $o(\log n)$ bit upper bound holds whenever the diameter of the underlying communication topology is sub-polynomial; see [22] for further discussion.

This helps explain why the above question is still open. Existing techniques simply do not capture the intuition for the problem above that each player needs to forward many bits of information about the inputs of previous players; in both communication models described above, the players only need to concern themselves with their own input, and let the blackboard or coordinator figure out the rest. While there are some topology-dependent communication lower bounds [12, 13], the problems studied in those works are not useful here and logarithmic factors for those problems are not accounted for, but here this is the main goal.

1.1 Our Results

We resolve the complexity of the coin problem. Namely, we show the following:

Theorem 1. *(Informal Coin Problem Lower Bound) Let $X_1, \ldots, X_n$ be a stream of uniform i.i.d. $\{0,1\}$ bits. Let A be any 1-pass streaming algorithm which reads $X_1, \ldots, X_n$ in order and outputs the majority bit with probability at least 0.999. Then A uses $\Omega(\log n)$ bits of memory.*

Here, the probability is over the input and the randomness used. Our proof of Theorem 1 uses techniques from information complexity, elaborated upon below. We note that the right definition of the information cost of a streaming algorithm is crucial for proving Theorem 1. We give two different definitions: (1) $\sum_{i=1}^{n} I(M_i; X_{\leq i})$ and (2) $\sum_{i=1}^{n} \sum_{j<i} I(M_i; X_j | M_{j-1})$, where M_i is the memory state of the algorithm after processing the i-th bit X_i. Here $I(A;B)$ denotes the mutual information between random variables A and B. We show that, for any deterministic algorithm computing the majority with 0.999 probability, both of these quantities are $\Omega(n \log n)$ (Theorem 13 and Claim 6). On the other hand, if the streaming algorithm is allowed to use private randomness (the streaming algorithm uses fresh randomness at every time step), then there exists a majority-computing algorithm such that the information cost as defined in (1) is $O(n)$, but for (2) it is still $\Omega(n \log n)$ for any streaming algorithm computing majority that uses private randomness (Corollary 14). Nevertheless, since we want a memory lower bound for the streaming algorithm for the uniform distribution on $X_1, \ldots, X_n$, we can always assume w.l.o.g. that the algorithm is deterministic by first fixing its private randomness. For deterministic algorithms with memory r, (1) evaluates to at most $n \cdot r$, thus implying a memory lower bound of $\Omega(\log n)$ bits. We also prove that for any streaming algorithm (possibly using private randomness) with memory r, (2) evaluates to at most $n \cdot r$, thus also implying a memory lower bound of $\Omega(\log n)$ bits. We then have a separation between information and space complexity for streaming algorithms (using private randomness) with respect to (1).

We next consider the problem of solving multiple copies of the coin problem simultaneously. Through a non-standard use of the data processing inequality and fundamental properties of mutual information, we are able to prove the following direct sum like theorem.

Theorem 2. *(Informal Direct Sum Theorem) Suppose we are given a sequence of kn i.i.d. stream updates, the j-th of which has the form $Y_j = (X_j, s_j)$, where s_j is chosen uniformly in $\{1, 2, \ldots, k\}$ and X_j is chosen uniformly in $\{0,1\}$. We interpret this as k independent instances of the coin problem, where the ℓ-th instance of the coin problem consists of the sequence of bits $X_{j_1}, \ldots, X_{j_r}$, where $s_{j_t} = \ell$ for each $t = 1, \ldots, r$. Suppose there is a streaming algorithm which, given an $\ell \in [k]$ at the end of the stream, outputs the majority bit of the ℓ-th instance of the coin problem with probability at least $1 - \frac{1}{4000}$. Then the algorithm uses $\Omega(k \log n)$ bits of memory.*

Here, the probability is over the input stream, private randomness used, and ℓ, which is chosen uniformly at random from $[k]$. We refer to the problem in Theorem 2 as the k-Coins Problem for a random order.

We also prove a variant of Theorem 2 (Corollary 17) in which we see exactly one update to each of the k instances, followed by exactly one update to each of the k instances again, and so on, repeating for n steps. That is, $s_j = (j-1) \mod k + 1$ for all $j \in [nk]$. We call this the Simultaneous k-Coins problem.

Next, using the notion of information cost for streaming algorithms, we show the hardness of calculating sums in the coin problem even up to large gaps. Consider the following problem. For a sequence of ± 1 integers, let S denote their sum. We say that the instance is a 0-instance if $|S| \leq 4\sqrt{n\alpha}$ and a 1-instance if $|S| \geq 4\sqrt{n\beta}$, which we refer to as the GapCoin(α, β) problem. We show the following:

Theorem 3. *Let M be a streaming algorithm (that might use private randomness) which outputs, with probability at least 2/3, the answer 0 on every input with $|\sum_i x_i| \leq 4\sqrt{n\alpha}$ and 1 on every input with*

$|\sum_i x_i| > 4\sqrt{n}\beta$. *Then (for $\alpha \geq 1$ and $c_2 \leq \beta \leq \mathrm{poly}(\log n)$, where $c_2 > 0$ is a large enough constant)*

$$\sum_{i=1}^{n}\sum_{j=1}^{i} I(M_i; X_j | M_{j-1}) > \Omega(n \log n/(\beta \log^2 \beta)^{11}),$$

where X is drawn from U_n.

We then look at the problem of solving the OR of k copies of the GapCoin(α, β) problem for $\log k \leq \alpha \leq \beta \leq \log^2 k$. We refer to this as the k-OR Problem. Using Theorem 3 and a direct sum on the information cost, under the uniform distribution over all the k instances, we show that solving the OR of multiple copies requires $\Omega\left(\frac{k \log n}{\mathrm{polylog}(k)}\right)$ bits of memory (Theorem 20). If the ratio β/α is sufficiently close to 1, we show how to remove the polylog(k) factor, obtaining an optimal $\Omega(k \log n)$ bit lower bound for this problem (Theorem 21), which is useful for our applications.

We next give a number of new lower bounds and upper bounds in the data stream model.

1.2 Data Stream Applications

Our lower bound for the coin problem gives new lower bounds for a number of problems in different data stream models. We now introduce these models and problems.

The most general data stream model is the *turnstile* model, for which there is an underlying d-dimensional vector x, initialized to 0^d, which undergoes a long sequence of additive updates of the form $x_{i_j} \leftarrow x_{i_j} + \Delta_j$, where $i_j \in \{1, \ldots, d\}$, $\Delta_j \in \{-1, 1\}$ and (i_j, Δ_j) is the j-th stream update.

A less general, though often more realistic model, is the *bounded deletions model* for which one still has that the updates Δ_j can be positive or negative, but one is promised that the norm $\|x\|_2 = (\sum_{i=1}^d x_i^2)^{1/2}$ never drops by more than an α-fraction of what it was at any earlier point in the stream, for a parameter α. We will focus on constant α in this work.

We let x_j denote the number of occurrences, or *frequency* of item j. While it is impossible to store accurate approximations to all frequencies with a small amount of memory, there are many useful summaries that suffice for applications.

1.2.1 "Lifting" Lower Bounds to the Bounded Deletions Model

There are a large number of lower bounds known in the turnstile model but they involve very sudden drops to the norm of the underlying vector and consequently do not hold in the bounded deletions model. Using our lower bounds for the coin problem, for a number of applications we are able to obtain improved lower bounds in the bounded deletions model, matching those that previously were only known to hold in the turnstile model.

Estimating the maximum frequency. This is denoted by $\|x\|_\infty = \max_{i \in \{1,\ldots,d\}} |x_i|$.

By a reduction from the so-called Index problem in 2-player communication complexity, this problem requires $\Omega(d)$ bits of memory to approximate up to a multiplicative factor of 2, even in random-order streams [10]. A common goal is to instead output an approximation to $\|x\|_\infty$ with additive error $\frac{1}{\sqrt{k}}\|x\|_2$ [27, 19, 26].

One can prove an $\Omega(k \log m)$ lower bound for turnstile streams using the standard Augmented-Indexing communication problem, see [24] for similar reductions. However, this lower bound inherently requires the norm of the underlying vector x to grow to poly(m) and then drop to 2^i for a random $i \in \{1, 2, \ldots, \log m\}$. These $\log m$ scales are precisely the source of the $\log m$ factor in the lower bound, while the k factor comes from having to solve a problem requiring k bits of information to solve at each scale. In the bounded deletions model one cannot reduce the norm of x by more than a constant factor and thus, only an $\Omega(k)$ lower bound is known and standard[1].

[1] One can prove this via a reduction from the Index problem. In this problem, Alice has a vector $a \in \{0,1\}^{k/9}$, and Bob has an index $j \in \{1, 2, \ldots, k/9\}$. In the stream, Alice presents the updates $x_i \leftarrow x_i + 1$ for each i for which $a_i = 1$. She sends the

We resolve this question in the bounded deletions model, up to poly$(\log k)$ factors. Our lower bound applies even to outputting an estimate Z with both additive and multiplicative error:

$$\|x\|_\infty - \frac{\|x\|_2}{\sqrt{k}} \leq Z \leq \gamma \|x\|_\infty + \frac{\|x\|_2}{\sqrt{k}} \tag{1}$$

Theorem 4. *($\|x\|_\infty$-Approximation) Any streaming algorithm achieving (1) in the bounded deletions model with probability at least 2/3, over its private randomness, requires $\Omega(k \log m)$ bits of memory if $\gamma = 1$, and $\Omega\left(\frac{k \log m}{\text{polylog}(k)}\right)$ bits of space for any $1 < \gamma \leq$ polylog(k).*

Estimating the variance. Another problem is to estimate the variance of the frequencies, or equivalently $\|x\|_2$. Here the goal is to output an estimate within $(1 \pm \epsilon)\|x\|_2$. It is known [24] that this requires $\Omega(\epsilon^{-2} \log m)$ bits of memory in the turnstile model. The proof again requires the underlying vector x to grow to poly(m) and then drop to 2^i for a random $i \in \{1, 2, \ldots, \log m\}$. These $\log m$ scales are the source of the $\log m$ factor in the lower bound, while the ϵ^{-2} factor comes from a single-scale lower bound.

In [21] it was asked if one could obtain a better upper bound for this problem in the bounded deletions model. The only known lower bound is $\Omega((\log(1/\alpha))/\epsilon^2)$ [21], given the promise that at each time in the stream, $\|x\|_2$ is never an α-fraction below of what it was at an earlier time in the stream.

We show the input stream we generate for the variance estimation problem, using the SIMULTANEOUS k-COINS PROBLEM with $k = \Theta(\epsilon^{-2})$, satisfies the bounded deletions property (with constant α) and we resolve the question in [21] in the negative.

Theorem 5. *(Euclidean Norm Estimation) Any streaming algorithm in the bounded deletions model (with the α above being an absolute small enough constant) which outputs a number within $(1 \pm \epsilon)\|x\|_2$, with probability at least 2/3 over its private randomness, requires $\Omega((\log m)/\epsilon^2)$ bits of memory.*

For this theorem, and throughout, we assume $\epsilon^{-2} \leq \min\{m, d\}^{0.9}$, which is the most common setting.

The Point Query Problem. Another related problem is the ℓ_2-Point Query Problem, in which one is given a single index j at the end of the stream and one would like to estimate x_j up to additive error $\epsilon\|x\|_2$ with constant probability over the private randomness, see, e.g., [18, 15].

Again this can be shown to require $\Omega(\epsilon^{-2} \log m)$ bits of memory in the turnstile model, using standard techniques as in [24]. However, again the $\log m$ factor occurs in the lower bound because of the need to have $\log m$ geometric scales and drastically shrink the norm of x at the end of the stream. This is also optimal given an $O(\epsilon^{-2} \log m + \log d)$ bit upper bound using the COUNTSKETCH data structure [11] (here we only need a constant number of rows in the data structure of [11], since we only need to be correct on a fixed index j. We can also first hash $\{1, 2, \ldots, d\}$ to a universe of size m^2 using a pairwise independent hash function specified with $O(\log(dm))$ bits.).

No lower bound better than $\Omega(\epsilon^{-2} + \log d)$ was known in the bounded deletions model. We are able to show an $\Omega(\epsilon^{-2} \log m)$ bit lower bound in the bounded deletions model.

Theorem 6. *(Point Query Problem) Any streaming algorithm which, in the bounded deletions model, solves the ℓ_2-Point Query Problem, requires $\Omega(\epsilon^{-2} \log m)$ bits of memory.*

The ℓ_2-Heavy Hitters Problem. Let $F_2 = \|x\|_2^2 = \sum_{i=1}^d x_i^2$ be the second moment of the data stream. We consider the ℓ_2-heavy hitters problem.

Definition 1. *In the ℓ_2-Heavy Hitters Problem with parameter ϵ, one should output a set S which (1) contains all indices $i \in [d]$ for which $x_i^2 \geq \epsilon^2 \cdot F_2$, and (2) does not contain any $i \in [d]$ for which $x_i^2 \leq \frac{\epsilon^2}{2} \cdot F_2$. Further, for all $i \in S$, one should output an estimate $\hat{x}_i$ with $|\hat{x_i} - x_i| \leq \epsilon\|x\|_2$.*

state of the streaming algorithm to Bob, who inserts $x_j \leftarrow x_j + 1$. If $a_j = 1$, then $\|x\|_\infty = 2$, otherwise $\|x\|_\infty = 1$. Note that $\frac{1}{\sqrt{k}}\|x\|_2 < \frac{1}{2}$, so from the approximate output Bob can deduce a_j and thus the space complexity of the streaming algorithm, which is Alice's message, must be at least $\Omega(k)$, the one-way communication complexity of the INDEX problem.

In the bounded deletions model, it was observed in [21] that an $O(\epsilon^{-2}(\log(1/\epsilon))\log(dm))$ bits of space algorithm [7, 6] applies (for any constant α in the definition of the bounded deletions model). There is a trivial $\Omega(\epsilon^{-2}\log d)$ lower bound just to write down the identities of the potentially ϵ^{-2} many ℓ_2-heavy hitters. However, in the bounded deletions model, it was unknown if there was also an $\Omega(\epsilon^{-2}\log m)$ lower bound, which may be much larger than $\Omega(\epsilon^{-2}\log d)$ if $d \ll m$.

We show an $\Omega(\epsilon^{-2}\log m)$ lower bound in the bounded deletions model, which together with the trivial $\Omega(\epsilon^{-2}\log d)$ lower bound, implies the algorithms of [7, 6] are optimal up to a $\log(1/\epsilon)$ factor.

Theorem 7. *(ℓ_2-Heavy Hitters Problem) Any streaming algorithm which, in the bounded deletions model, solves the ℓ_2-Heavy Hitters Problem with sufficiently large constant probability, over the private randomness, requires $\Omega(\epsilon^{-2}\log m)$ bits of memory.*

1.2.2 Random Order Streaming Lower Bounds

Another well-studied model is the random order model. Here, as in the turnstile model, we allow both positive and negative updates (see Section 1.2.3 below for further discussion on the necessity of this for the following problems), though the order of the stream is not allowed to be worst case, but rather the stream updates arrive in a uniformly random order. Even in this model we are able to prove strong lower bounds for both the Point Query Problem and ℓ_2-Heavy Hitters Problem:

Theorem 8. *(Point Query Problem) Any streaming algorithm which, in the random order model, solves the ℓ_2-Point Query Problem with sufficiently large constant probability, over the random order and private randomness, requires $\Omega(\epsilon^{-2}\log m)$ bits of memory.*

Theorem 9. *(ℓ_2-Heavy Hitters Problem) Any streaming algorithm which, in the random order model, solves the ℓ_2-Heavy Hitters Problem with sufficiently large constant probability, over the random order and private randomness, requires $\Omega(\epsilon^{-2}\log m)$ bits of memory.*

1.2.3 On the Tightness of the Models in Our Lower Bounds: New Upper Bounds

For the problems above, we consider if stronger lower bounds are possible in different models.

In the more general turnstile model, in the 2006 IITK Workshop on Data Streams, in Open Problem 3, Cormode[2] asks whether it is possible to estimate $\|x\|_\infty$ up to additive error $\frac{1}{\sqrt{k}}\|x\|_2$ using fewer than $O(k(\log d)\log m)$ bits of memory, which was the previous upper bound.

Given that we have shown a lower bound of $\Omega(k\log m)$ for this problem in the bounded deletions model, it is natural to ask whether our lower bound can be improved to $\Omega(k(\log d)\log m)$ for turnstile streaming algorithms. We show this is impossible, by giving an algorithm for solving this problem and using $O(k(\log k + \log\log m)\log(dm))$ bits of memory, which works in the turnstile model and thus also the bounded deletions model, showing our lower bound is tight up to a $\log k + \log\log m$ factor.

Theorem 10. *(ℓ_∞-Estimtion) There is a turnstile streaming algorithm which approximates $\|x\|_\infty$ up to additive error $\frac{1}{\sqrt{k}}\|x\|_2$ with probability at least $2/3$ (over the private randomness) and using $O(k(\log(dm))(\log k + \log\log m))$ bits of memory.*

Another natural question that Theorem 8 and Theorem 9 raise is whether our lower bounds hold even in the more restrictive insertion-only model. Recall that stream updates have the form $x_{i_j} \leftarrow x_{i_j} + \Delta_j$ in general, and the insertion-only streaming model requires $\Delta_j = 1$ for all j.

We show this is impossible in the insertion-only model, at least if the stream updates are randomly ordered. Recalling that our lower bounds Theorem 8 and Theorem 9 hold in the random order model, this shows that our lower bounds in Theorem 8 and Theorem 9 require deletions to be allowed as well.

To show this, we give a new algorithm which solves both the ℓ_2-Heavy Hitters problem and the ℓ_2-Point Query Problem in the insertion-only model, for randomly ordered streams.

[2] https://www.semanticscholar.org/paper/OPEN-PROBLEMS-IN-DATA-STREAMS-AND-RELATED-TOPICS-ON-Agarwal-Baswana/5394ab5bf4b66bfb52f111525d6141a3226ba883

Theorem 11. *(ℓ_2-Heavy Hitters and ℓ_2-Point Query Problems) There is a streaming algorithm in the insertion-only model, with randomly ordered updates, which, with probability at least $2/3$ (over the private randomness and the random order), solves the ℓ_2-Heavy Hitters problem and ℓ_2-Point Query Problem. For the ℓ_2-Heavy Hitters problem, the memory is $O(\epsilon^{-2}(\log d + \log^2(1/\epsilon) + \log^2 \log m) + \log m)$ bits. For the ℓ_2-Point Query Problem, the memory is $O(\epsilon^{-2}(\log^2(1/\epsilon) + \log^2 \log m) + \log m + (\log(1/\epsilon) \log d))$ bits.*

For the ℓ_2-Heavy Hitters problem, our estimates $\hat{x_i}$ for those i in our output set S satisfy the stronger guarantee that $\hat{x_i} = (1 \pm \epsilon)x_i$ (see Section 5.1 for details).

Notice that our algorithm for ℓ_2-heavy hitters bypasses the $\Omega(\epsilon^{-2} \log m)$ lower bound for bounded deletions for $d \ll m$. Theorem 9 also gives a separation between the turnstile model, with randomly ordered updates, and insertion-only models with randomly ordered updates. The above theorem is stated with error probability 1/3 for simplicity, though can be generalized to any constant failure probability.

Also, note that the previous best algorithm for heavy hitters in insertion streams, even if one assumes a random order, was $O(\epsilon^{-2}(\log 1/\epsilon) \log(dm))$ bits of memory. In addition to a worse memory bound, the previous algorithm was not able to obtain the stronger guarantee that $\hat{x_i} = (1 \pm \epsilon)x_i$ for all i in the output set.

In the Warwick Workshop on Data Summarization, Jelani Nelson explicitly poses the question for insertion-only streams[3], if one can achieve $O(\epsilon^{-2} \log(dm))$ bits of memory in insertion-only streams, namely, if the $O(\log 1/\epsilon)$ factor can be removed from the upper bound. Our Theorem 11 shows that this is indeed possible in insertion-only streams, at least when the stream updates are randomly ordered. Indeed, under the standard setting of parameters when $\log d = \Omega(\log^2(1/\epsilon) + \log^2 \log m)$, our algorithm gives $O(\epsilon^{-2} \log d + \log m)$ bits of memory, significantly improving the earlier $O(\epsilon^{-2}(\log(1/\epsilon) \log(dm)))$ bit algorithm.

1.3 Our Techniques

We start by describing the techniques involved in proving our main lower bound for the coin problem.

1.3.1 Technical Overview of our Lower Bound for the Coin Problem

The starting point for our lower bounds is Theorem 1, which states that a streaming algorithm that computes the majority of n bits with a constant probability (≥ 0.999) requires $\Omega(\log n)$ bits of memory. Note that simple counting gives a streaming algorithm that uses exactly $\log n$ bits of memory.

The intuition for the lower bound is that a successful streaming algorithm will need to "remember" $\Omega(1)$ bits of memory about each "scale" of its past bits. At step i the memory state M_i will remember $\Omega(1)$ bits about each of the blocks $\{X_i\}$, $\{X_{i-2..i-1}\}$, $\{X_{i-6..i-3}\}$, $\{X_{i-14..i-7}\}$, etc. Here the notation $X_{a..b}$ is used to represent the input bits $X_a, X_{a+1}, \ldots, X_b$. There are two main challenges in realizing this approach: (1) coming up with the correct information-theoretic formalism (the definition of "remember"), which is sufficiently strong for memory lower bounds and downstream applications; (2) proving the actual lower bound on the information remembered.

Sidestepping the first challenge for a moment, let us sketch the proof of the lower bound on the information remembered. Let us focus at scale 2^j. Let $t = n/2^j$, and let $S_1, \ldots, S_t$ be the sums of each of the t blocks of X_i's (so that $S_m = \sum_{\ell=(m-1)\cdot 2^j+1}^{m \cdot 2^j} X_\ell$). $M_{m \cdot 2^j}$ represents the memory state of the the streaming algorithm after the m-th block.

If $M_{m \cdot 2^j}$ contains $\delta \ll 1$ information about S_m (conditioned on $M_{(m-1)\cdot 2^j}$), then conditioned on $M_{m \cdot 2^j}$ the sum S_m has almost full variance (i.e., $\mathrm{Var}(S_m | M_{m\cdot 2^j}, M_{(m-1)\cdot 2^j}) > 2^j \cdot (1 - \varepsilon)$). Conditioned on the memory states $M_{m \cdot 2^j}$, $m \in [t]$, the sums S_m become independent. Therefore, if $M_{m \cdot 2^j}$ contains $\delta \ll 1$ information about S_m (conditioned on $M_{(m-1)\cdot 2^j}$) for the vast majority of m's, then conditioned on the $M_{m \cdot 2^j}$s, the variance of the sum $\sum S_m = \sum X_i$ remains close to the full variance n, which means that the algorithm gains no advantage in estimating the majority of the X_i's (Claim 5).

One complication of the above outline is that it shows that M_i must have $\Omega(1)$ information on average about the block $X_{i-2^j+1..i}$. However, this range actually contains j different scales. For the argument to

[3] See Slide 86 here: `https://warwick.ac.uk/fac/sci/dcs/research/focs/conf2017/abstracts/jn-slides.pdf`

work we need M_i to contain $\Omega(1)$ information on average about $X_{i-2^j+1..i-\alpha\cdot 2^j}$ for some constant $\alpha > 0$. We prove this — essentially by showing that M_i cannot have $\Omega(1)$ information about the sum of 2^j bits (low variance in the sum) if it only has information about the last $\alpha \cdot 2^j$ of them (Claim 4). Putting those together, we get that M_i has $\Omega(1)$ bits of information about $\Omega(\log n)$ disjoint blocks of X's (the different scales).

It is important to choose the correct conditioning, while defining the "information" M_i knows about $X_{\leq i}$. Consider the following simple example: the streaming algorithm M samples $n/10^4$ $\{\pm 1\}$ bits and starts the counter with their sum, that is, $M_1 \sim \mathrm{Bin}(n/10^4, 1/2)$ (this is the binomial distribution over $n/10^4$ uniformly random ± 1 bits). Then M keeps adding the input stream to the counter, that is $M_i = M_{i-1} + X_i$. M outputs $\mathrm{sign}(M_n)$. It is not hard to see that this algorithm computes the majority of the X_i's correctly with a high constant probability. However, M_i contains almost no information about X_i (only $O(1/n)$). In fact, the information between M_i and $X_{\leq i}$ ($I(M_i; X_{\leq i})$) is $O(1)$. And hence, $\sum_i I(M_i; X_{\leq i})$ is $O(n)$. However, we show that for deterministic algorithms, even this information quantity works to show an $\Omega(\log n)$ lower bound.

For randomized algorithms that use fresh randomness at every time step, to bypass the above example, when measuring the amount of information M_i has about X_i, we should condition on M_{i-1} — this captures the fact that when creating M_i out of M_{i-1} we must store a bit capturing the value of X_i. More generally, when measuring the mutual information between M_i and the block $X_{i-2^j+1..i-\alpha\cdot 2^j}$, we condition on M_{i-2^j}. It appears that the most natural way to formalize this conditioning is by defining the total information cost of a streaming algorithm as

$$IC(M, U_n) := \sum_{i=1}^{n} \sum_{j=1}^{i} I(M_i; X_j | M_{j-1}). \tag{2}$$

Here, U_n represents the uniform distribution over $\{+1, -1\}^n$ and X is drawn from U_n. It will be interesting to see whether this quantity finds other streaming applications. For computing the majority, the discussion above implies that, on average

$$I(M_i; X_j | M_{j-1}) = \Omega\left(\frac{1}{i-j+1}\right).$$

Coming back to the first challenge, the quantity $IC(M, U_n)$ from (2) is bounded from above by $\sum_{i=1}^{n} H(M_i)$, and can be used in direct sum contexts (with private randomness).

1.3.2 A Direct Sum Theorem and the OR of Many Copies

In this section, we give a brief overview of our lower bounds for two generalizations of the coin problem. First, we consider *a direct sum theorem* for solving the coin problem for k copies. Before we delve into the lower bound for the problem defined in Theorem 2, let us consider the following problem P_1: M sees k instances of the coin problem $(Y^1, Y^2, \ldots, Y^k)$ in a stream as follows: at the i-th step, M sees the i-th input bit for each instance, that is, $Y_i^j, \forall j \in [k]$. At the end, M is given a random index $\ell \in [k]$ and is asked to output the majority of the ℓ-th instance. A slight variant of the Problem P_1 turns out to be a very important special case for our applications, and we refer to this as the Simultaneous k-Coins Problem.

The notion of information cost as defined in (2) suffices to prove an $\Omega(n \log n)$ information lower bound for streaming algorithms that use private randomness. Therefore, a direct sum theorem can be readily used to prove that any streaming algorithm M that solves P_1 with probability at least 0.9999 requires $\Omega(k \cdot n \log n)$ information cost. A direct sum theorem just uses the fact that

$$\sum_{i=1}^{n} \sum_{j=1}^{i} \sum_{l=1}^{k} I(M_i; Y_j^l | M_{j-1} Y_j^{<l}) \geq \sum_{l=1}^{k} \left(\sum_{i=1}^{n} \sum_{j=1}^{i} I(M_i; Y_j^l | M_{j-1}) \right) \tag{3}$$

(because Y_j^l is independent of $Y_j^{<l}$) given M_{j-1}). Therefore, for at least half fraction of ls,

$$\sum_{i=1}^{n} \sum_{j=1}^{i} I(M_i; Y_j^l | M_{j-1}) \leq \frac{2}{k} \sum_{i=1}^{n} \sum_{j=1}^{i} \sum_{l=1}^{k} I(M_i; Y_j^l | M_{j-1} Y_j^{<l}).$$

And for at least $9/10$ fraction of ls, the success probability is at least 0.999. Let l' be an l in the intersection, Using M, one can obtain a streaming algorithm for solving a single instance of the coin problem, which uses private randomness to generate the other instances (the fact that $\forall j, Y_1^j, Y_2^j, \ldots Y_n^j$ are independently drawn is crucial for this step), has information cost equivalent to $\sum_{i=1}^{n}\sum_{j=1}^{i} I(M_i; Y_j^l | M_{j-1})$ and success probability of at least 0.999.

We stress that the direct sum theorem here is subtle. For example, we cannot give the streaming algorithm all updates to instances ℓ' for $\ell' > \ell$ at the end of the stream, and still prove a direct sum theorem, which a priori one might suspect since it is reminiscent of techniques in 2-player communication complexity used to prove lower bounds for the AUGMENTED-INDEXING problem [24]. Indeed, one can show that if $k = \log n$ and one is given all updates to all coins $\ell' > \ell$ at the end of the stream, then $O(\log n)$ bits suffice to solve this problem, rather than the $\Omega(\log^2 n)$ bits one would expect from a direct sum of $\log n$ copies. Indeed, we leave it to the reader to check that the algorithm which replaces 0 updates to the j-th coin by -10^{4j} and 1 updates to the j-th coin by 10^{4j} in the stream, and maintains a single counter of the sum of all $O(\log n)$ scaled updates across all k coins, and then subtracts all (scaled) updates to all coins $\ell' > \ell$ at the end of the stream, can be used to determine the bias of the ℓ-th coin with only $O(\log n)$ bits of memory.

Things get trickier for the problem defined in Theorem 2 (P_2). In P_2, M is not given all the k instances in parallel, but randomly interleaved in a stream of kn independent updates, of the form $Y_j = (X_j, S_j)$, where S_j is chosen uniformly in $\{1, 2, \ldots, k\}$ and X_j is chosen uniformly in $\{0, 1\}$. Proving a lower bound for this version of the k-Coins Problem is crucial for the streaming applications downstream.

The proof of Theorem 2 is more challenging than the direct sum theorem mentioned above. One of the difficulties faced is that the number of updates in P_2 is kn instead of n, and hence to prove an $\Omega(k \log n)$ memory lower bound, we need an $\Omega(k^2 \cdot n \log n)$ information lower bound on M. To prove the theorem, we first divide the information cost of the streaming algorithm into information cost for the individual instances as follows.

$$\sum_{i=1}^{nk}\sum_{j=1}^{i} I(M_i; X_j | M_{j-1}) = \sum_{s\in[k]}\sum_{i=1}^{nk}\sum_{j\in[i] \text{ and } S_j=s} I(M_i; X_j | M_{j-1})$$

Then we show, through a non-standard use of the data processing inequality, that a streaming algorithm that solves the k-COINS PROBLEM even when S_js are fixed as $S_j = (j-1) \mod k+1$, requires $\Omega(k \cdot n \log n)$ information cost each for a large fraction of individual instances, and a direct sum can be applied henceforth. In fact, we can show the above information cost lower bound for a typical sequence $\{S_j\}_{j\in[nk]}$ when the S_j are *i.i.d.* uniform in $[k]$, hence, proving a $\Omega(k \log n)$ memory lower bound and Theorem 2.

We next consider the second generalization, that is, a memory lower bound for solving the OR of k coin problems, that is, output 1 if the majority bit of any of the k instances is 1 and 0 otherwise. In communication complexity, the most famous information lower bound for a communication problem that solves the OR of n instances, is that for two-party set-disjointness. [4] gave a simple lower bound of $\Omega(n)$ on the communication required to solve set-disjointness, using a direct sum theorem. There are three steps to using a direct sum theorem to prove such a lower bound: (1) define the information cost for a communication protocol that decomposes into a sum of information costs on single instances; (2) show that an algorithm for solving the OR can be used for solving a single instance; and (3) show that the information cost for solving a single instance is large. Set-disjointness is equivalent to solving the OR of n AND instances. The authors of [4] complete these three steps by lower bounding the information cost of any communication protocol that solves a single instance (AND of two bits), on an input distribution which is supported on $\{00, 01, 10\}$. However, our techniques can only be used to lower bound (2) when the input $X = (X_1, X_2, \ldots, X_n)$ to the coin problem is drawn from a product distribution as M is only allowed to use private randomness.

We sidestep the above issue by changing the underlying coin problem to the following (P_3). Consider the following problem: given a sequence of n $\{-1, +1\}$ bits X_1, X_2,..., X_n, output 0 if $|\sum_i X_i| \leq 4\sqrt{n\alpha}$ and 1 if $|\sum_i X_i| \geq 4\sqrt{n\beta}$. We refer to this as the GapCoin(α, β) problem. We will always set $\alpha = \log k$ and prove our bounds for $4\alpha \leq \beta = O(\log^2 k)$. To describe our results, we focus on the case $\alpha = \log k$ and $\beta = 4\alpha$; the general proof is similar. For a sequence of n ±1 integers, let S denote their sum. We say that the instance is a 0-instance if $|S| \leq 4\sqrt{n \log k}$ and a 1-instance if $|S| \geq 8\sqrt{n \log k}$. The crucial fact is that: when S is the sum of n i.i.d. uniform ± 1 integers, then with high probability, $|S| \leq 4\sqrt{n \log k}$. Thus, the probability of a 0 instance is high under the uniform distribution on $\{-1,+1\}^n$. Hence, an algorithm A for solving the OR of k P_3 problems, can be used to develop an algorithm A' for a single instance of P_3, where A' privately generates the other instances, to be given to A, one step at a time. Also, whenever A outputs a 0, the answer to P_3 is definitely 0 and whenever A outputs 1, with high probability, the other instances all evaluate to 0, and hence, the answer to P_3 is also 1. This completes Step (2). Step (1) is easy to show and follows similarly to the discussion given for our direct sum theorem for P_1. Step (3) is what requires significant technical work. We prove that any streaming algorithm that solves P_3 requires $\Omega\left(\frac{\log n}{(\log k \log\log k)^{22}}\right)$ bits of memory. To give a glimpse of our proof, we partition our stream into $t = (\log k \log\log k)^2$ blocks and use a corollary of Theorem 1 on n/t sized blocks (the following theorem), which might be of independent interest.

Theorem 12 (Informal Version of Claim 3)**.** *Let B be a streaming algorithm (that might use private randomness) of length m. Then, for $\epsilon > 0$, one of two things holds:*

1. *Either $IC(B, U_m) = \Omega(m \log m \cdot \epsilon^{10})$.*
2. *Or there exists a distribution μ on m bits, such that*
   - $\forall x \in \{-1,+1\}^m, \mu(x) \leq 2U_m(x)$,
   - $E_{x\sim\mu}[\sum_i x_i] \geq \Omega\left(\sqrt{\frac{m}{\log(1/\epsilon)}}\right)$*, and*
   - $\|B(U_m) - B(\mu)\|_1 < \epsilon$.

Recall that U_m represents the uniform distribution over $\{-1,+1\}^m$. Thus, for a streaming algorithm that has low information cost, there exists a distribution μ, which has a significantly higher number of ones, on average, compared to the uniform distribution, but B cannot distinguish between μ and the uniform distribution. Informally, to construct μ, we strategically shift some weight from $-x$ to x, and show that if B could distinguish between μ and uniform, it would have "solved majority" with low information cost. Next, we show that if there is a low information cost streaming algorithm that solves P_3, then it should be able to distinguish between μ^t and U_m^t, which would contradict Theorem 12.

When the ratio of $\frac{\beta}{\alpha}$ is sufficiently close to 1 in the GapCoin(α,β) problem, which is important for our applications, we give a more direct proof of the k-OR problem, which we call the k-OR small gap problem. The proof is via a direct reduction from the Simultaneous k-Coins Problem. Given $\ell \in [k]$ at the end of the stream in an instance of the Simultaneous k-Coins Problem, we can add $4\sqrt{n \log k}$ updates to the ℓ-th coin. This makes the sum of the ℓ-th coin way more than the average, and by making the ratio $\frac{\beta}{\alpha}$ subconstant (though still large enough to be meaningful in the applications), determining the solution to the GapCoin(α,β) problem decides the majority bit of the ℓ-th coin.

1.3.3 Technical Overview of our Algorithms

ℓ_∞-Estimation. Our algorithm is inspired from a universe reduction technique of [23], which shows for $0 < p < 2$, there is a randomized oblivious linear map which takes a vector x and produces a vector y so that with good probability, if i is a so-called ℓ_p-heavy hitter of x, then $y_{h(i)}$ is an ℓ_p-heavy hitter of y, where h corresponds to a hash function defining y. The oblivious linear map is nothing other than a CountSketch map, with stronger randomness guarantees than what is usually needed. Here we need to argue this holds for $p = 2$, which in a certain sense simplifies the analysis of [23], but on the other hand, we also need to argue that there are no entries of y that are heavy but do not correspond to any heavy hitter in x. We need this because we need $\|y\|_\infty \approx \|x\|_\infty$, which was not necessary in [23].

After applying the universe reduction, we then feed y into a CountSketch data structure itself to find all of the ℓ_2-heavy hitters of y together with good approximations to their frequencies. Taking the maximum frequency found gives us a good approximation to $\|y\|_\infty$, and in turn to $\|x\|_\infty$. This can all be done in one pass because the universe reduction and CountSketch maps are both linear and oblivious; hence, so is their composition. The CountSketch map uses $O(k(\log d)\log m)$ bits of space and finds the ℓ_2-heavy hitters, but since we are applying it to the vector y, which has dimension $\mathrm{poly}(k \log d)$, we obtain overall $O(k(\log k + \log \log d) \log m)$ bits of space.

ℓ_2-Heavy Hitters and ℓ_2-Point Query. Our algorithm for ℓ_2-Point Query is a slight modification to our algorithm for the ℓ_2-Heavy Hitters problem, so we only describe the latter. Our algorithms for both problems do not use the Gaussian process and chaining-based arguments in the previous $O(\epsilon^{-2}(\log(1/\epsilon)) \log(dm))$ space algorithms [7, 6]. Rather, the arguments are elementary, which could be of independent interest.

Our key subroutine Main1 assumes we know F_2 and m in advance (m is the number of elements in the stream). Our actual algorithm Main runs a constant-factor F_2-estimate on the side and invokes Main1 with this F_2-estimate. It creates a new instance of Main1 each time the stream length doubles, and only maintains two instances of Main1 at a time, the current one starting from when m last doubled, and the previous one which completed the last time m doubled. This enables us to reuse space. Since Main1 ends up being run on a stream of length 2^i for some known value of i with $2^i = \Theta(m)$, and since our F_2-estimate is a good enough constant factor approximation, by using the random order property of the stream we can argue that the heavy hitters found by invoking Main1 on this substream contain the actual heavy hitters for the entire stream, and no items that are sufficiently far from being heavy in the actual stream.

The idea behind our Main1 algorithm is to partition the stream into consecutive and contiguous equal-sized blocks $B_1, \ldots, B_t$, where $t \approx \epsilon\sqrt{F_2}$. This means that each B_i has size about $\epsilon^{-1}m/\sqrt{F_2}$. An important idea is that for each item $j \in [d]$, we randomly assign $O(\log(1/\epsilon))$ numbers $h^1(j), \ldots, h^{O(\ln(1/\epsilon))}(j) \in \{1, 2, \ldots, t\}$ to it. A key observation is that if $f_j^2 \geq \epsilon^2 F_2$, or equivalently $f_j \geq \epsilon\sqrt{F_2}$, then because the stream is randomly ordered, j has a constant probability of appearing in each block B_i. Moreover, it has probability $1 - O(\epsilon^2)$ of appearing in a B_i for which i is one of the $O(\log(1/\epsilon))$ numbers which is assigned to j. We say:

Definition 2. *An item $j \in [d]$ is* **excited** *in B_i if j occurs in block B_i and if i is one of the $O(\log(1/\epsilon))$ numbers which is assigned to j.*

Since there are only $O(\epsilon^{-2})$ total heavy hitters, by a union bound each heavy hitter is excited in some B_i. Moreover, one can also show that the set S^i of items j for which both $j \in B_i$ and $h^\ell(j) = i$ for some ℓ, $1 \leq \ell \leq O(\log(1/\epsilon))$, has size $O(\epsilon^{-2}\log(1/\epsilon))$ with good probability; in particular by a union bound with constant probability, simultaneously every heavy hitter is in such a set.

We next reduce S^i to at most a single element by looking at the next block B_{i+1}. We cannot afford to store S^i because it has size $O(\epsilon^{-2}\log(1/\epsilon))$ and each item identifier requires $O(\log d)$ bits. Instead, we choose another hash function g which maps the elements of S^i to a range of size $\mathrm{poly}(\epsilon^{-1}\log m)$, and we only store the hash, under g, of the items in S^i. For each of these hashed identities $g(j)$, we check if there is a unique item j' in B_{i+1} for which $g(j') = g(j)$ and for which $h^q(j') = i$ for some $q \in [10\log(1/\epsilon)]$. It is crucial that we also filter by ensuring that $h^q(j') = i$ for some $q \in [10\log(1/\epsilon)]$, as otherwise since the blocks are polynomially (in m) large there will be elements j' for which $g(j') = g(j)$, since the range of g is only $\mathrm{poly}(\epsilon^{-1}\log m)$. However, almost all of the items j in these next blocks are not excited in B_i.

A crucial property here is that for an actual heavy hitter j, when it is excited in a block B_i, it will also have good probability of occurring in the next block. Since there are $O(\log(1/\epsilon))$ values of q, we can make sure this happens with probability $1 - \mathrm{poly}(\epsilon)$ for one of the blocks that j is excited in. Further, we can ensure with good probability j is the unique item with this property. The intuition is that items with very low frequency are very unlikely to be excited in a block B_i, and then occur in the very next block. On the other hand, there are only a small number of items with large frequency, and these items will all be excited in blocks far away from each other since they are each only excited in a few random blocks.

An issue is that since there are $m^{\Omega(1)}$ blocks for which we start tracking the S^i, there will still be blocks not containing a heavy hitter which pass the above test, meaning that after inspecting B_{i+1}, there will be exactly one hashed identity with count equal to 1. Indeed, this happens with $\mathrm{poly}(\epsilon)$ probability. To counter this, we run a subroutine Identify to take the single hashed identifier which passed the above test for a given block and track the actual (unhashed) identifier over the next $O(\epsilon^{-2}\log m)$ blocks, counting the number of times the item occurs. The first issue is we do not know the actual identifier at this point, since we could not afford to store it when creating the set S^i corresponding to the block B_i where this identifier occurred. We first look at the next $100\log(1/\epsilon)$ blocks in our Identify procedure to recover the actual identifier with probability $1-\text{poly}(\epsilon)$. This probability is large enough to ensure for all blocks which actually passed the test and contained a single heavy hitter, that we recover the corresponding heavy hitter. However, this probability is not enough to rule out false positives. However, now that we have an actual identifier, we count its number of occurrences over the next $4000\epsilon^{-2}\log m = O(\epsilon^{-2}\log m)$ blocks. At this point, the probability a non-heavy hitter occurs in a significantly large fraction of these blocks is $1/\text{poly}(m)$, which is small enough that we can ensure there are no false positives.

One issue which arises with tracking an actual $O(\log d)$-bit identifier over the next $O(\epsilon^{-2}\log m)$ blocks, is that we may be tracking up to $O(\epsilon^{-2}\log m)$ actual identifiers at any given time, and this requires $\Theta(\epsilon^{-2}(\log d)\log m)$ bits of space, which is too large. We fix this by enforcing that if we are ever already tracking an actual identifier, then we do not begin tracking a new one. This may affect correctness now, since we decide to not track some actual identifiers, and therefore may not find one of the heavy hitters. Fortunately, because of the random order property, the locations of the heavy hitters are spread out in the stream, and one can show under our condition that $m \geq \text{poly}(\epsilon^{-1}\log(d))$ (as otherwise there is a simpler algorithm we describe), that we never run into this problem.

One last problem is that we may start tracking a non-heavy hitter in the $4000\epsilon^{-2}\log m$ blocks preceding where we start tracking a heavy hitter j, and therefore since we have already invoked Identify, we may decide not to start tracking item j. Note that if the probability of tracking the actual identifier of a non-heavy hitter were only $\text{poly}(\epsilon)$, and say, ϵ were not too small, then we almost certainly would start tracking a non-heavy hitter among the $4000\epsilon^{-2}\log m$ blocks preceding that of a heavy hitter, and therefore decide not to start tracking the heavy hitter. Fortunately though, the probability of tracking the actual identifier of a non-heavy hitter can be shown to be at most $\text{poly}(\epsilon/\log(dm))$, for a large enough polynomial, and this probability is small enough to argue that for every heavy hitter, for any of the $4000\epsilon^{-2}\log m$ blocks preceding it, we do not start tracking an actual identifier of some other item. Thus, when we process the block containing the heavy hitter, we are free to start tracking its actual identifier.

Acknowledgments. We would like to thank Jelani Nelson and Huacheng Yu for their many useful comments and discussions.

2 Preliminaries

First, we define some notation used in the paper. For $a, b \geq 0$, $\sqrt[a]{b}$ represents $b^{1/a}$. For a positive integer n, we use $[n]$ to denote the set $\{1, 2, \ldots, n\}$. Mostly through the paper, we will use capital letters to denote random variables, for example, X and Y. Given a sequence of random variables, $X_1, X_2, \ldots, X_n$, we use the notation $X_{a..b}$ (for $1 \leq a < b \leq n$) to represent the set $\{X_a, X_{a+1}, \ldots, X_b\}$ of random variables. Further, we use $X_{\leq i}$, $X_{>i}$, X_{-i} and $X_{a..b\cap S}$ to represent the set $\{X_j\}_{j\leq i}$, $\{X_j\}_{j>i}$, $\{X_1, \ldots, X_n\}\setminus\{i\}$ and $\{X_j\}_{j\in[a,b]\cap S}$ of random variables respectively (if the random variable X_0 is non empty, then the sets are defined according starting with X_0). We sometimes use these notations in superscripts when the indexing of the random variables is done in the subscript. Given a random variable X, we use the notation $x \sim X$ to denote the process that X takes value x with probability $\Pr[X = x]$. We use the notation $x \in_R S$ to represent that x is drawn uniformly at random from the set S. We use $\text{Var}[X|Y]$ and $\text{Var}[X|Y = y]$ to denote the expected variance of the random variable X conditioned on Y, and conditioned on the event $Y = y$, respectively. To represent the distribution of X conditioned on $Y = y$, we will use the notation X_y. Throughout the paper, we will denote bits by the sets $\{0, 1\}$ and $\{-1, +1\}$ interchangeably. We assume log is base 2 unless specified otherwise.

We use Stirling's approximation [32] for factorials in the paper, which states that[4]

$$\sqrt{2\pi}n^{n+\frac{1}{2}}e^{-n} \leq n! \leq en^{n+\frac{1}{2}}e^{-n}.$$

[4] Here, e, known as the Euler's number, is a mathematical constant, and the base of the natural logarithm.

We use U_n to represent the uniform distribution on $\{-1,+1\}^n$. $\mathrm{Ber}(p)$ represents the distribution on $\{0,1\}$ with the probability of 1 being p. $\mathrm{Bin}(k,p)$ denotes the Binomial distribution over k bits where each bit is 1 with probability p and 0 otherwise. $\mathrm{Bin}(k,p)$ is just the distribution of the sum of k i.i.d. (independently and identically distributed) random variables from $\mathrm{Ber}(p)$. We use the following facts about the binomial distribution in the paper.

We also use the following fact about the entropy of the $\mathrm{Bin}(n,1/2)$ distribution [1]:

$$H(\mathrm{Bin}(n,1/2)) = \frac{1}{2}\log_2(\pi\cdot e\cdot n/2) + O(1/n).$$

Next, we give a brief prelude to the tools used from information theory. For a probability distribution $P\colon \mathcal{X}\to[0,1]$, $H(P)=\sum_{x\in\mathcal{X}}P(x)\log_2(1/P(x))$ represents the entropy of the distribution P. We also use $H(X)$ to denote the entropy of the the random variable X. $I(X;Y|Z)$ represents the mutual information between X and Y conditioned on the random variable Z. $I(X;Y|Z)=H(X|Z)-H(X|Y,Z)$, where $H(X|Y)=\mathbb{E}_{y\sim Y}H(X|Y=y)$. Next, we describe some of the properties of mutual information used in the paper.

1. (Chain Rule) $I(AB;C)=I(A;C)+I(B;C|A)$.
2. If $I(D;B|A,C)=0$, then $I(A;B|C)\geq I(A;B|C,D)$.
3. If $I(D;B|C)=0$, then $I(A;B|C)\leq I(A;B|C,D)$.

Property 1 follows from the chain rule for entropy (H). Properties 2 and 3 follow from the observation that

$$I(A;B|C)+I(D;B|A,C)=I(AD;B|C)=I(D;B|C)+I(A;B|C,D).$$

As mutual information is non-negative, if $I(D;B|A,C)=0$, then $I(A;B|C)\geq I(A;B|C,D)$ (because $I(D;B|C)\geq 0$) and if $I(D;B|C)=0$, then $I(A;B|C)\leq I(A;B|C,D)$.

Given two distributions $P,Q\colon\mathcal{X}\to[0,1]$, the KL-Divergence from Q to P is defined as

$$\mathbb{D}\left(P\|Q\right)=\sum_{x\in\mathcal{X}}P(x)\log_2\left(\frac{P(x)}{Q(x)}\right).$$

Given two distributions $P,Q\colon\mathcal{X}\to[0,1]$, $\|P-Q\|_1=\sum_{x\in\mathcal{X}}|P(x)-Q(x)|$ represents the distance between the two distributions. We use Pinsker's inequality[5], which states that,

$$\|P-Q\|_1\leq\sqrt{2\,\mathbb{D}\left(P\|Q\right)}.$$

We will also use the following relationship between mutual information $I(X;Y)$ and the KL-Divergence, which can be easily verified by plugging in the definitions:

$$I(X;Y)=\mathbb{E}_{y\sim Y}\mathbb{D}\left(X_y\|X\right).$$

In words, the mutual information between random variables X and Y is the expected KL-Divergence from the distribution X to the distribution of X conditioned on $Y=y$.

We will also make use of the following chaining inequality from [6]:

Lemma 1. *(Lemma 9 of [6], restated) If $k=\Omega(1/\epsilon^2)$, and Π is a $k\times n$ matrix of i.i.d. random variables uniform in $\{+1,-1\}$ (or even 8-wise independence suffices), then*

$$\mathbf{E}[sup_t\ |\|\Pi f^{(t)}\|_2^2-k\|f^{(t)}\|_2^2|]\leq\epsilon k\|f^{(m)}\|_2^2,$$

where $f^{(1)},\ldots,f^{(m)}$ are the frequency vectors of an n-dimentional insertion-only stream. Here $f^{(t)}$ represents the frequency vector after the first t updates.

[5] https://www.cs.bgu.ac.il/~asml162/wiki.files/pollard-pinsker.pdf.

Here, $\|x\|_2$ represents the ℓ_2-norm of x, that is, $\|x\|_2^2 = \sum_i x_i^2$.

Next, we define some of the terms used in the paper in relation to streaming algorithms. Let M denote the sequence of states of a streaming algorithm on a stream of length n, defined as follows: M_i represents the memory state of the algorithm after i steps. We can treat M_i as a random variable that depends on the input stream and the randomness used by the streaming algorithm. We say M uses *private randomness*, if at all time steps, it only has access to independent sources of randomness. In fact, we prove lower bounds for all sequences of messages (random variables) $M = M_0, M_1, ..., M_n$ of the form $M_i = \mathcal{M}_i(M_{i-1}, X_i, R_i)$ where R_i is the private randomness used at step i. Here, $\mathcal{M}_i$ can be an arbitrary function of M_{i-1}, X_i and the R_is, and does not need to be defined by a low-space streaming algorithm. For the sake of our lower bounds, we will refer to streaming algorithms as encompassing such sequences of messages. We simulate the streaming algorithm M by an n-party communication protocol, where the i-th party, denoted $\mathcal{P}_i$, receives the i-th input of the stream. Communication is allowed only from $\mathcal{P}_i$ to $\mathcal{P}_{i+1}$ for all $i \in [n-1]$. To obtain an n-party protocol from a streaming algorithm M, given X_1 (the first input of the stream), $\mathcal{P}_1$ can simulate the first step of the streaming algorithm by using private randomness and X_1. After the simulation, $\mathcal{P}_1$ sends the memory state m_1 as its message to $\mathcal{P}_2$, and so on. Given m_i and X_{i+1}, $\mathcal{P}_{i+1}$ can simulate the $(i+1)$-st step of M using private randomness and send m_{i+1} as its message to $\mathcal{P}_{i+2}$. $\mathcal{P}_n$ outputs what the streaming algorithm outputs. We define a new notion of information cost for such an n-party communication protocol (B):

$$IC(B, \mu) := \sum_{i=1}^{n} \sum_{j=1}^{i} I(B_i; X_j | B_{j-1}),$$

where X_i represents the input to party $\mathcal{P}_i$, $X = (X_1, \ldots, X_n)$ is drawn from the distribution μ, and B_i represents the message sent by $\mathcal{P}_i$ to $\mathcal{P}_{i+1}$. B_i depends on B_{i-1}, X_i and the private randomness used by $\mathcal{P}_i$ (we will refer to B_0 as the random variable generating the starting message for party $\mathcal{P}_1$). In this paper, we will sometimes treat M as a communication protocol without the reduction mentioned above. We define further notions of information cost for n-party protocols as needed subsequently in the paper. As noted in Remark 1, for streaming algorithms M that only use private randomness, we show that when $X = (X_1, \ldots, X_n)$ is drawn from the uniform distribution over $\{0,1\}^n$, then

$$\sum_{i=1}^{n} \sum_{j=1}^{i} I(M_i; X_j | M_{j-1}) \leq \sum_{i=1}^{n} |M_i| \leq n \cdot (\text{space used by } M),$$

where $|M_i|$ is the length of the i-th message, which is at most the memory used by M. Thus, we can lower bound the space required by M to perform a task by lower bounding the information cost of any such n-party protocol that performs the same task.

3 Lower Bounds

In this section, we prove our main lower bound for the coin problem, and then extend it to a number of direct sum like variants which will be useful for the streaming applications. We start with a subsection collecting non-standard lemmas about the binomial distribution. We then prove a "one-scale" version of our lower bound, which only gives an $\Omega(n)$ information lower bound ($\Omega(1)$ lower bound on memory) for the streaming coin problem. We finally extend this to a "$\log n$-scales" lower bound, giving us our ultimate $\Omega(\log n)$ bit memory lower bound.

3.1 Variance-Information Properties of the Binomial Distribution

Our first claim concerns how conditioning affects the variance of a binomial random variable.

Claim 1 (The Effect of Conditioning on Variance). *There is a universal constant $C_1 > 0$ such that the following holds. Let $X \sim Bin(k, 1/2)$, and let E be an event with $\Pr[E] = \varepsilon$ (for $\varepsilon < 1/2$). Then*

$$\mathbb{E}[(X - k/2)^2 | E] \leq C_1 k \log(1/\varepsilon). \tag{4}$$

Proof. This is a corollary of Hoeffding's inequality, which asserts

$$\Pr[(X - k/2)^2 \geq tk] \leq 2e^{-2t}.$$

Therefore, $\Pr[(X - k/2)^2 \geq c\ln(1/\varepsilon)k] \leq 2\varepsilon^{2c}$. And for $\varepsilon < 1/2$,

$$\mathbb{E}\left[(X - k/2)^2 \cdot \mathbf{1}_{(X-k/2)^2 > \ln(1/\varepsilon)k}\right] \leq \sum_{c=1}^{k}(c+1)\ln(1/\varepsilon)k \cdot 2\varepsilon^{2c} = O(k\varepsilon^2 \log(1/\varepsilon)). \qquad \square$$

The next claim is more involved, and argues that if a random variable Y reveals low information about a binomial random variable X, then X still has high variance conditioned on a typical value of Y. One of the key insights of our overall strategy is using variance as a convenient measure of uncertainty of the sum of bits, and relating it to how much information is revealed about the input. The proof of the next claim uses Claim 1.

Let ϕ denote the function $\phi(\varepsilon) := \sqrt{\varepsilon}\log(1/\varepsilon)$. Note that $\phi(\varepsilon) \to 0$ as $\varepsilon \to 0$. Let $\varepsilon_\phi = \frac{1}{e^2}$. Observe that, ϕ is monotonically increasing and concave on the interval $[0, \varepsilon_\phi]$ (it is in fact concave everywhere but we do not need this).

Claim 2 (Low Information implies High Variance)**.** *Let $X \sim Bin(k, 1/2)$ be a binomial random variable, and let Y be any random variable. Then, for all $k \geq 1$, if*

$$I(X;Y) \leq \varepsilon \leq 2\varepsilon_\phi,$$

then

$$\mathbb{E}_y \operatorname{Var}[X|Y = y] \geq (1 - C_2\phi(\varepsilon/2)) \cdot (k/4),$$

for a universal constant $C_2 \geq 1$.

Proof. We have

$$\varepsilon \geq I(X;Y) = \mathbb{E}_y\mathbb{D}\left(X_y \| X\right).$$

Recall that X_y represents the distribution of X conditioned on the event $Y = y$. Note that with probability $\geq 1 - \varepsilon/(2\varepsilon_\phi)$ over the choice of Y, we have $\mathbb{D}\left(X_y\|X\right) \leq 2\varepsilon_\phi$. Consider now a fixed y such that

$$\mathbb{D}\left(X_y\|X\right) = \delta \leq 2\varepsilon_\phi.$$

By Pinsker's Inequality, we have

$$\|X_y - X\|_1 \leq \sqrt{2\delta}.$$

As $X \sim \mathrm{Bin}(k, 1/2)$ is a binomial random variable, $\operatorname{Var}(X) = k/4$. We next prove that $\operatorname{Var}(X_y)$, given that $\|X_y - X\|_1 \leq \sqrt{2\delta}$, is not much less than $k/4$. For this, we define another random variable Z which has the following joint distribution with X. Let $\mathcal{I} = \{x \mid \Pr[X = x] \leq \Pr[X_y = x]\}$.

$$\Pr[Z = z, X = x] = \begin{cases} \Pr[X = x] & \text{if } x \in \mathcal{I} \text{ and } z = x \\ \Pr[X_y = x] & \text{if } x \notin \mathcal{I} \text{ and } z = x \\ \Pr[X = x] - \Pr[X_y = x] & \text{if } x \notin \mathcal{I} \text{ and } z = \perp \\ 0 & \text{otherwise} \end{cases}$$

It is easy to see that: for each value of x, $\Pr[Z = x] = \min\{\Pr[X = x], \Pr[X_y = x]\}$. And, $\Pr[Z = \perp] = \|X_y - X\|_1/2 \leq \sqrt{\delta/2}$. Let E be the event that $Z = \perp$. Denote

$$\eta := \Pr[E] = \|X_y - X\|_1/2 \leq \sqrt{\delta/2}.$$

Let X_1, Z_1 and X_2, Z_2 be two independent sets of random variables with the above defined joint distribution. Let E_1 and E_2 be the corresponding events. Then we have (as $\eta < 1/2$)

$$\operatorname{Var}(X_y) = \frac{1}{2} \cdot \mathbb{E}_{x_1,x_2 \sim X_y}(x_1 - x_2)^2$$

$$\begin{aligned}
&\geq \frac{1}{2} \cdot \mathbb{E}_{z_1 \sim Z_1, z_2 \sim Z_2} (z_1 - z_2)^2 \cdot \mathbf{1}_{z_1 \neq \perp \wedge z_2 \neq \perp} && (\text{for all } z \neq \perp, \Pr[Z = z] \leq \Pr[X_y = z]) \\
&= \frac{1}{2} \cdot \mathbb{E}_{x_1 \sim X_1, x_2 \sim X_2} (x_1 - x_2)^2 \cdot \mathbf{1}_{\neg E_1 \wedge \neg E_2} && (\text{conditioned on event } \neg E, Z \text{ is equal to } X) \\
&= k/4 - \frac{1}{2} \cdot \mathbb{E}_{x_1 \sim X_1, x_2 \sim X_2} (x_1 - x_2)^2 \cdot \mathbf{1}_{E_1 \vee E_2} \\
&\geq k/4 - \mathbb{E}_{x_1 \sim X_1, x_2 \sim X_2} (x_1 - x_2)^2 \cdot \mathbf{1}_{E_1} \\
&= k/4 - \mathbb{E}_{x_1 \sim X_1, x_2 \sim X_2} ((x_1 - k/2) - (x_2 - k/2))^2 \cdot \mathbf{1}_{E_1} \\
&= k/4 - \mathbb{E}_{x_1 \sim X_1} (x_1 - k/2)^2 \cdot \mathbf{1}_{E_1} - \Pr[E_1] \cdot \mathbb{E}_{x_2 \sim X_2} (x_2 - k/2)^2 \\
&\geq k/4 - C_1 k \eta \log(1/\eta) - k\eta/4 && (\text{using Claim 1 and that } \Pr[E_1] = \eta) \\
&> k/4 - C' k \eta \log(1/\eta^2) && (\text{for large enough constant } C' \text{ and } \eta < 1/2) \\
&= k/4 - C' k \cdot \phi(\eta^2) \\
&\geq k/4 - C' k \cdot \phi(\delta/2).
\end{aligned}$$

The last inequality holds since ϕ is monotone increasing on $[0, \varepsilon_\phi]$, and $\delta/2$ belongs to this interval. Next, we use the concavity of the function ϕ on $[0, \varepsilon_\phi]$ to obtain:

$$\begin{aligned}
\mathbb{E}_{y \sim Y} \operatorname{Var}(X_y) &\geq \mathbb{E}_{y \sim Y} \operatorname{Var}(X_y) \cdot \mathbf{1}_{\mathbb{D}(X_y \| X) \leq 2\varepsilon_\phi} \\
&\geq \mathbb{E}_{y \sim Y} \mathbf{1}_{\mathbb{D}(X_y \| X) \leq 2\varepsilon_\phi} \cdot \left(k/4 - C' k \cdot \phi \left(\frac{1}{2} \mathbb{D}\left(X_y \| X\right) \right) \right) \\
&\geq \Pr[\mathbb{D}\left(X_y \| X\right) \leq 2\varepsilon_\phi] \cdot \left(k/4 - C' k \cdot \phi \left(\frac{1}{2} \cdot \mathbb{E}_{y \sim Y} \left[\mathbb{D}\left(X_y \| X\right) \mid \mathbb{D}\left(X_y \| X\right) \leq 2\varepsilon_\phi \right] \right) \right) \\
&\geq (1 - \varepsilon/(2\varepsilon_\phi)) \cdot (k/4 - C' k \cdot \phi(\varepsilon/2)) \\
&\geq (1 - C_2 \phi(\varepsilon/2)) \cdot (k/4)
\end{aligned}$$

The third from last inequality follows using the concavity of the function ϕ on the interval $[0, \varepsilon_\phi]$. The second to last inequality follows from the fact that $\mathbb{E}_{y \sim Y} \left[\mathbb{D}\left(X_y \| X\right) \mid \mathbb{D}\left(X_y \| X\right) \leq 2\varepsilon_\phi \right] \leq \mathbb{E}_{y \sim Y} \left[\mathbb{D}\left(X_y \| X\right) \right] \leq \varepsilon$ and that ϕ is monotone increasing on $[0, \varepsilon_\phi]$. The last inequality follows, for a large enough constant $C_2 \geq 1$, since $\phi(\varepsilon) \geq \varepsilon$ (for $\varepsilon \leq 2\varepsilon_\phi$). □

Claim 2 implies the following simple corollary involving an additional independent random variable Δ. Intuitively, the next claim shows that if the variance of the binomial random variable X conditioned on a typical value of $Y = y$ is a constant factor smaller than without the conditioning, then Y must contain information about the *magnitude* of X. To formalize the notion of "Y containing information about the magnitude of X", we claim that not just $I(X; Y) > \varepsilon$, but even if we add a Binomial random variable Δ of magnitude comparable with that of X, the mutual information $I(X + \Delta; Y) > \varepsilon$ is still non-vanishing. We will need this stronger statement later in the lower bound proof.

Claim 3 (Corollary of Low Information implies High Variance)**.** *Let k be an integer, and let $\zeta > 0$ and $m < \zeta k$. Let $X \sim Bin(k, 1/2)$ be a binomial random variable, and let Y be any random variable. In addition, let $\Delta \sim Bin(m, 1/2)$ be distributed independently from X and Y. Then if*

$$I(X + \Delta; Y) \leq \varepsilon \leq 2\varepsilon_\phi,$$

then

$$\mathbb{E}_y \operatorname{Var}[X | Y = y] \geq (1 - C_2 \phi(\varepsilon/2) - \zeta) \cdot (k/4).$$

Proof. By Claim 2 we have that

$$\mathbb{E}_y \operatorname{Var}[X + \Delta | Y = y] \geq (1 - C_2 \phi(\varepsilon/2)) \cdot ((k + \zeta k)/4) \geq (1 - C_2 \phi(\varepsilon/2)) \cdot (k/4).$$

Since Δ is independent of X and Y, we have

$$\mathbb{E}_y \operatorname{Var}[X | Y = y] = \mathbb{E}_y \operatorname{Var}[X + \Delta | Y = y] - \operatorname{Var}(\Delta) \geq (1 - C_2 \phi(\varepsilon/2) - \zeta) \cdot (k/4). \quad \square$$

One additional ingredient we will need in our proof is the following claim. One should think of S and T as intermediate messages $\mathcal{P}_b$ and $\mathcal{P}_a$ in the n-party communication protocol, with T being the message sent by a party $a < b$. The claim then lower bounds, conditioned on T, how much information S (or message sent by party b) reveals about a *subset* of the input stream from $a+1$ to b, given that it reveals a lot of information about the sum of these bits plus *noise* $(Z_{k+1}, ..., Z_{k+m})$. Note that the noise is what makes this lower bound go through — this is why we needed the stronger lower bound on $I(X+\Delta;Y)$ and not just on $I(X;Y)$ in Claim 3.

Claim 4 (Relating Information of Individual Bits to the Sum)**.** *Let k and m be integers. Let $\{Z_i\}_{i=1}^{k+m}$ be i.i.d. $\sim Ber(1/2)$ and let $W = \sum_{i=1}^{k+m} Z_i$. Let S, T be any random variables such that (1) T is independent of all Z_i's (that is, $I(T; Z_1, ..., Z_{k+m}) = 0$); (2) S and T are independent from $Z_{k+1}, \ldots, Z_{k+m}$, even when conditioned on $Z_{\le k}$ (that is, $I(ST; Z_{k+1}, ..., Z_{k+m} | Z_{\le k}) = 0$). Let $t \le k$ be an integer. Then*

$$I(Z_1, \ldots, Z_{k-t}; S|T) \ge I(W; S|T) - C_4 \cdot \frac{t}{m}, \tag{5}$$

where $C_4 > 0$ is a universal constant.

Proof. We have

$$I(W;S|T) + I(Z_{\le k-t}; S|TW) = I(W, Z_{\le k-t}; S|T) = I(Z_{\le k-t}; S|T) + I(W; S|Z_{\le k-t}T),$$

and, therefore,

$$\begin{aligned} I(Z_{\le k-t}; S|T) &= I(W;S|T) + I(Z_{\le k-t}; S|WT) - I(W; S|Z_{\le k-t}T) \\ &\ge I(W;S|T) - I(W;S|Z_{\le k-t}T) \\ &\ge I(W;S|T) - I(W;S, Z_{k-t+1..k}|Z_{\le k-t}T) \\ &= I(W;S|T) - I(W; Z_{k-t+1..k}|Z_{\le k-t}T) - I(W;S|Z_{\le k}T) \\ &= I(W;S|T) - I(W; Z_{k-t+1..k}|Z_{\le k-t}T) \quad \text{(because } I(W;S|Z_{\le k}T) = 0 \text{ as explained below)} \\ &= I(W;S|T) - I(W; Z_{k-t+1..k}|Z_{\le k-t}). \qquad (T \text{ is independent of all } Z_i\text{s; explained below)} \end{aligned}$$

The second to last equality follows from the following calculations:

$$\begin{aligned} I(W;S|Z_{\le k}T) &\le I(W, Z_{k+1}, ..., Z_{k+m}; S|Z_{\le k}T) \\ &= I(Z_{k+1}, ..., Z_{k+m}; S|Z_{\le k}T) + I(W;S|Z_{\le k+m}T) && \text{(Chain rule)} \\ &\le I(Z_{k+1}, ..., Z_{k+m}; ST|Z_{\le k}) + 0 && (W \text{ is determined by } Z_{\le k+m}) \\ &= 0. \end{aligned}$$

Next, using the chain rule,

$$\begin{aligned} I(W; Z_{k-t+1..k}|Z_{\le k-t}T) + I(T; Z_{k-t+1..k}|Z_{\le k-t}) &= I(WT; Z_{k-t+1..k}|Z_{\le k-t}) \\ &= I(W; Z_{k-t+1..k}|Z_{\le k-t}) + I(T; Z_{k-t+1..k}|Z_{\le k-t}W). \end{aligned}$$

As $I(T; Z_{k-t+1..k}|Z_{\le k-t}) = 0$ and $I(T; Z_{k-t+1..k}|Z_{\le k-t}W) = 0$, the last equality follows.

Observe that the expression $I(W; Z_{k-t+1..k}|Z_{\le k-t})$ is in terms of *i.i.d.* Bernoulli variables. We have

$$H(Bin(n, 1/2)) = \frac{1}{2}\log_2(\pi \cdot e \cdot n/2) + O(1/n),$$

and thus

$$\begin{aligned} I(W; Z_{k-t+1..k}|Z_{\le k-t}) &= H(Bin(t+m, 1/2)) - H(Bin(m, 1/2)) \\ &= \frac{1}{2}\log_2(1 + t/m) + O(1/m) < C_4 \cdot t/m, \end{aligned}$$

for a universal constant $C_4 > 0$. □

Reiterating, for the above claim to give non-trivial lower bounds on the information between S and subset of individual bits $Z_1, ..., Z_{k-t}$, it is important that, even though S is independent of $Z_{k+1}, ..., Z_{k+m}$, $I(S; \sum_{i=1}^{k} Z_i + \sum_{i=k+1}^{k+m} Z_i | T)$ is high, where $\sum_{i=k+1}^{k+m} Z_i$ is just noise to S. Without the addition of noise, the claim does not hold as S can be the parity of the bits $Z_1, ..., Z_k$ and thus reveal no information about this subset of bits. Getting ahead of ourselves, this then allows us to partition the bits of the input stream into subsets forming $\Omega(\log n)$ scales and showing that any streaming algorithm computing majority must reveal information about each of these “subsets” or *scales*.

3.2 Warmup: a Lower Bound at a Single Scale

Let $X_1, \ldots, X_n$ be a stream of uniform *i.i.d.* bits. Let a streaming algorithm M be a sequence of messages of the form $M_i = \mathcal{M}_i(M_{i-1}, X_i, R_i)$ where R_i is the private randomness used by the streaming algorithm M at step i; see Section 2 for further background. Let $|M_i|$ be the length of the i-th message. Usually we will treat M_0 as empty but the proofs hold if M_0 is a random variable dictating the starting message for M_1. For deterministic algorithms, M_0 and R_is are empty.

We start by proving a lower bound of $\Omega(n)$ for the n-party communication problem that “computes majority”, and thus an $\Omega(1)$ lower bound on the memory of a streaming algorithm. The lower bound is parameterized by *scale* k and holds for many scales.

The first claim below is technical, and the reader should think of the S_i as corresponding to states of the streaming algorithm at t equispaced positions in the stream, while the X_i will correspond to the sum of input bits between these successive states. Intuitively the claim shows that if on average most states do not reveal much information about the sum of bits (plus noise) before it, even conditioned on all previous states, then the final variance of the sum of all bits in the stream is large.

Claim 5 (Low Average Information Across Blocks Implies High Variance)**.** *Let k be an integer. Let $X_1, \ldots, X_t$ be independent random variables such that $X_i \sim Bin(k_i, 1/2)$, where $k_i \geq k, \forall i \in [t]$. Let $S_0, S_1, \ldots, S_t$ be random variables such that (1) X_i is independent of $S_{<i}$, and (2) conditioned on S_i, $(X_{\leq i}, S_{<i})$ are independent from $(X_{>i}, S_{>i})$. Let $\zeta > 0$, and let $m \leq \zeta k$ be an integer. Let $\Delta_1, \ldots, \Delta_t$ be i.i.d. $\sim Bin(m, 1/2)$ and independent of the X's and the S's. Let*

$$I_i := I(S_i; X_i + \Delta_i | S_{<i}),$$

and

$$\bar{I} := \frac{1}{t} \sum_{i=1}^{t} I_i.$$

If $\bar{I} \leq \varepsilon_\phi$, then

$$\mathbb{E}_{s_0 \sim S_0, S_t \sim S_t} \operatorname{Var}\left[\sum_{i=1}^{t} X_i \,\middle|\, S_t = s_t, S_0 = s_0\right] \geq (1 - C_3\phi(\bar{I}) - \zeta) \cdot (kt)/4 \tag{6}$$

for some universal constant $C_3 \geq 1$.

Observe that for the above claim to hold, conditioned on S_i, we want $(X_{\leq i}, S_{<i})$ to be independent of $(X_{>i}, S_{>i})$. If S_i represents the message sent by player i in the n-party communication protocol, the condition holds only when players are deterministic or only use private randomness.

Proof. Observe first that additional conditioning can only reduce variance in expectation, and that if we fix all the S_i's, the X_i's become mutually independent. We have

$$\mathbb{E}_{s_0, s_t \sim S_0, S_t} \operatorname{Var}\left[\sum_{i=1}^{t} X_i \,\middle|\, S_t = s_t, S_0 = s_0\right]$$

$$\begin{aligned}
&\geq \mathbb{E}_{s_0\ldots s_t \sim S_0 \ldots S_t} \operatorname{Var}\left[\sum_{i=1}^{t} X_i \;\middle|\; S_0 = s_0, S_1 = s_1, S_2 = s_2, \ldots, S_t = s_t\right] \\
&= \mathbb{E}_{\vec{s}} \sum_{i=1}^{t} \operatorname{Var}[X_i|\vec{s}] \\
&= \mathbb{E}_{\vec{s}} \sum_{i=1}^{t} \operatorname{Var}[X_i|s_i, s_{<i}] && (X_i \text{ is independent of } S_{>i} \text{ conditioned on } S_i) \\
&= \sum_{i=1}^{t} \mathbb{E}_{s_{<i}} \mathbb{E}_{s_i|s_{<i}} \operatorname{Var}[X_i|s_i, s_{<i}].
\end{aligned}$$

We will treat each term of this sum separately. For any fixed i, if

$$I_i = I(X_i + \Delta_i; S_i|S_{<i}) = I(X_i + \Delta_i; S_i S_{<i}) - I(X_i + \Delta_i; S_{<i}) = I(X_i + \Delta_i; S_i S_{<i}) \leq \varepsilon_\phi,$$

then

$$\mathbb{E}_{s_{<i}} \mathbb{E}_{s_i|s_{<i}} \operatorname{Var}[X_i|s_i, s_{<i}] \geq (1 - C_2\phi(I_i/2) - \zeta) \cdot (k_i/4) \geq (1 - C_2\phi(I_i) - \zeta) \cdot (k/4)$$

by Claim 3. The number of i's such that $I_i < \varepsilon_\phi$ is at least $t \cdot (1 - \bar{I}/\varepsilon_\phi)$. Among those, the average value of I_i is at most $\bar{I}$, and by concavity of ϕ we have

$$\sum_{i=1}^{t} \mathbb{E}_{s_{<i}} \mathbb{E}_{s_i|s_{<i}} \operatorname{Var}[X_i|s_i, s_{<i}] \geq t \cdot (1 - \bar{I}/\varepsilon_\phi) \cdot (1 - C_2\phi(\bar{I}) - \zeta) \cdot (k/4) \geq (1 - C_3\phi(\bar{I}) - \zeta) \cdot (kt)/4,$$

where, here again, the second inequality follows from $\phi(\bar{I}) \geq \bar{I}$ and $C_3 \geq 1$ being a large enough constant. □

This allows us to prove the following key lemma, which applies to all streaming protocols (randomized using only private randomness or deterministic). Intuitively this lemma says that if the output of the n-party communication protocol manages to non-trivially reduce the variance of the sum, which it does by reducing the squared expectation of the bias, then the total information that the protocol has about the subset of input bits at *scale* k must be $\Omega(n)$.

Lemma 2 (Variance Reduction of the Algorithm Implies High Information at One Scale)**.** *For all $\varepsilon > c'n^{-1/20}$, there exist $\delta_2 \geq c''\varepsilon^4$ and $\alpha \geq c_6\varepsilon^5$, such that*

$$\mathbb{E}_{M_n}\left[\mathbb{E}\left[\left(\sum_{i=1}^{n} X_i - n/2\right) \;\middle|\; M_n\right]^2\right] > \varepsilon n \tag{7}$$

implies that for all integers $1/\alpha < k \leq n\varepsilon/3$, the following holds:

$$\sum_{j=1}^{n} I(M_j; X_{j-k+1..j-\lceil \alpha k \rceil}|M_{j-k}) > \delta_2 n. \tag{8}$$

Here, $c' > 0$ is a large constant and $c'', c_6 > 0$ are small enough constants.

Proof. Since

$$\mathbb{E}_{M_n}\left[\mathbb{E}\left[\left(\sum_{i=1}^{n} X_i - n/2\right)^2 \;\middle|\; M_n\right]\right] = \mathbb{E}\left[\left(\sum_{i=1}^{n} X_i - n/2\right)^2\right] = \frac{n}{4},$$

we have that (7) is equivalent to

$$\mathbb{E}_{m_n \sim M_n} \operatorname{Var}\left[\sum_{i=1}^{n} X_i \;\middle|\; M_n = m_n\right] < (1/4 - \varepsilon) \cdot n. \tag{9}$$

Let $t = \lfloor n/k \rfloor - 1 > 2/\varepsilon$. For $s \in \{1, \ldots, k\}$, consider t blocks of length k starting from location s. Let $B_1, \ldots, B_t$ be their sums. Formally,

$$B_i = \sum_{j=s+(i-1)k}^{s+ik-1} X_j.$$

Denote

$$S_i := M_{s+ik-1},$$

for $i = 0, \ldots, t$, so that S_i is the message sent after block i. From (9) we have

$$\begin{aligned}(1/4 - \varepsilon/2) \cdot tk > (1/4 - \varepsilon) \cdot n &> \mathbb{E}_{m_n \sim M_n} \operatorname{Var}\left[\sum_{i=1}^{n} X_i \;\middle|\; M_n = m_n \right] \\ &\geq \mathbb{E}_{m_n \sim M_n; (s_0, s_t) \sim (S_0, S_t)|_{m_n}} \operatorname{Var}\left[\sum_{i=1}^{n} X_i \;\middle|\; m_n, s_0, s_t \right] \\ &= \mathbb{E}_{m_n, s_0, s_t} \operatorname{Var}\left[\sum_{i=1}^{s-1} X_i + \sum_{i=1}^{t} B_i + \sum_{i=s+tk}^{n} X_i \;\middle|\; m_n, s_0, s_t \right] \\ &\geq \mathbb{E}_{m_n, s_0, s_t} \operatorname{Var}\left[\sum_{i=1}^{t} B_i \;\middle|\; m_n, s_0, s_t \right] \\ &\quad \left(\textstyle\sum_{i=1}^{t} B_i \text{ is independent of } X_{<s}, X_{\geq s+tk} \text{ conditioned on } S_0, S_t, M_n\right) \\ &= \mathbb{E}_{s_0, s_t} \operatorname{Var}\left[\sum_{i=1}^{t} B_i \;\middle|\; s_0, s_t \right].\end{aligned}$$

Next, observe that the S_i's and the B_i's satisfy the conditions of Claim 5. Set $\zeta = \varepsilon$, and $k\varepsilon/2 < m < k\varepsilon$. Let Δ_i be *i.i.d.* $\sim \text{Bin}(m, 1/2)$ random variables. Then Claim 5 implies that

$$\frac{1}{t} \sum_{i=1}^{t} I(S_i; B_i + \Delta_i | S_{<i}) > \phi^{-1}(\varepsilon/C_3) =: \delta_1 > 0. \tag{10}$$

Otherwise, if $\frac{1}{t} \sum_{i=1}^{t} I(S_i; B_i + \Delta_i | S_{<i}) < \phi^{-1}(\varepsilon/C_3)$[6], then using Claim 5,

$$\mathbb{E}_{s_0, s_t} \operatorname{Var}\left[\sum_{i=1}^{t} B_i \;\middle|\; s_0, s_t \right] \geq (1 - \varepsilon - \varepsilon) \frac{kt}{4},$$

contradicting the calculations above. As $\phi(\delta_1) \leq 4\delta_1^{\frac{1}{4}}$, $\delta_1 \geq \left(\frac{\varepsilon}{4C_3}\right)^4$. Note that

$$I(S_i; B_i + \Delta_i | S_{<i}) = I(S_i; B_i + \Delta_i | S_{i-1}). \quad \text{(conditioned on } S_{i-1}\text{, } S_{<i-1} \text{ and } S_i, B_i + \Delta_i \text{ are independent)}$$

Let $\alpha := \frac{\delta_1 \varepsilon}{8C_4} > 0$. Let $r = \lceil \alpha k \rceil$. Then, by Claim 4 we have

$$\begin{aligned} I(S_i; B_i + \Delta_i | S_{i-1}) &= I(M_{s+ik-1}; B_i + \Delta_i | M_{s+(i-1)k-1}) \\ &\leq I(M_{s+ik-1}; X_{s+(i-1)k..s+ik-1-r} | M_{s+(i-1)k-1}) + C_4 \cdot \frac{r}{m} && \text{(Claim 4)} \\ &< I(M_{s+ik-1}; X_{s+(i-1)k..s+ik-1-r} | M_{s+(i-1)k-1}) + \frac{\delta_1}{2}. && (m > k\varepsilon/2, r \leq 2\alpha k) \end{aligned}$$

[6] Here, ϕ^{-1} is a function from $(0, 1]$ to $(0, \varepsilon_\phi]$. As ϕ is monotonically increasing on $(0, \varepsilon_\phi]$ and $\phi(\varepsilon_\phi) > 1$, ϕ^{-1} is well defined. Thus, $\phi^{-1}(x) \leq \varepsilon_\phi$ by definition.

In the second to last inequality we use Claim 4 with parameters $S = M_{s+ik-1}$, $Z_1, .., Z_k = X_{s+(i-1)k}, ..., X_{s+ik}$, $\sum_{j=k+1}^{k+m} Z_j = \Delta_i$, and $T = M_{s+(i-1)k-1}$. Therefore, by (10),

$$\sum_{i=1}^{t} I(M_{s+ik-1}; X_{s+(i-1)k..s+ik-1-r} | M_{s+(i-1)k-1}) > \frac{\delta_1 t}{2}. \tag{11}$$

Summing up (11) over $s = 1, \ldots, k$ (and letting M_{i-1}, X_i be empty for $i \leq 0$), we get

$$\sum_{j=1}^{n} I(M_j; X_{j-k+1..j-\lceil \alpha k \rceil} | M_{\leq j-k}) > \frac{\delta_1 tk}{2} > \frac{\delta_1 n}{3} =: \delta_2 n. \tag{12}$$

Next, we show the bounds on α and δ_2. $\alpha = \frac{\delta_1 \varepsilon}{8C_4} \geq c_6 \varepsilon^5$ for small enough constant $c_6 > 0$. $\delta_2 = \delta_1/3 \geq c'' \varepsilon^4$ for a small enough constant $c'' > 0$. □

3.3 An $\Omega(\log n)$ Lower Bound for the Coin Problem by Combining All Scales

We can now use our lower bound at a single scale in Lemma 2 to glue together $\Omega(\log n)$ scales. Here we use a parameter α to split the input bits into $\Omega(\log n)$ geometrically growing disjoint intervals, and apply our $\Omega(1)$ single scale lower bound at each scale. We first prove Theorem 13, which concerns the variance-information tradeoff of a *deterministic* streaming algorithm, and then conclude a memory usage lower bound for computing majority from it. We eventually also prove an information cost lower bound for randomized algorithms that only use private randomness (as discussed in Section 2).

Theorem 13 (Formal statement of Theorem 1)**.** *For any $\varepsilon > c' n^{-\frac{1}{20}}$, there exists $\delta \geq c''' \epsilon^5$ (for a small enough constant $c''' > 0$ and a large enough constant $c' > 0$) such that if*

$$\sum_{i=1}^{n} I(M_i; X_{\leq i}) \leq \delta n \log n, \tag{13}$$

(where M_i is deterministic function of $X_{\leq i}$), then

$$\mathbb{E}_{M_n}\left[\mathbb{E}\left[\left(\sum_{i=1}^{n} X_i - n/2\right) \;\middle|\; M_n\right]^2\right] \leq \varepsilon n. \tag{14}$$

Note that, when the conditioning on M_n is outside of the square, we have:

$$\mathbb{E}_{M_n}\left[\mathbb{E}\left[\left(\sum_{i=1}^{n} X_i - n/2\right)^2 \;\middle|\; M_n\right]\right] = \mathbb{E}\left[\left(\sum_{i=1}^{n} X_i - n/2\right)^2\right] = \frac{n}{4}.$$

Therefore, using the definition of variance, (14) is equivalent to

$$\mathbb{E}_{m_n \sim M_n} \operatorname{Var}\left[\sum_{i=1}^{n} X_i \;\middle|\; M_n = m_n\right] \geq (1/4 - \varepsilon) \cdot n. \tag{15}$$

Proof of Theorem 13. Let M be a streaming algorithm as in the statement of the theorem, such that (14) does not hold. That is,

$$\mathbb{E}_{M_n}\left[\mathbb{E}\left[\left(\sum_{i=1}^{n} X_i - n/2\right) \;\middle|\; M_n\right]^2\right] > \varepsilon n. \tag{16}$$

By Lemma 2 this implies that there exists $\delta_2 \geq c''\varepsilon^4$ and $\alpha \geq c_6\varepsilon^5$ ($c'', c_6 > 0$ are constants) such that for all integers $\alpha^{-1} < k \leq n\varepsilon/3$,

$$\sum_{j=1}^{n} I(M_j; X_{j-k+1..j-\lceil \alpha k \rceil} | M_{j-k}) > \delta_2 n. \tag{17}$$

By the assumption that M_{j-k} is a deterministic function of $X_{\leq j-k}$ we have

$$\begin{aligned} I(M_j; X_{j-k+1..j-\lceil \alpha k \rceil} | M_{j-k}) &\leq I(M_j; X_{\leq j-k}, X_{j-k+1..j-\lceil \alpha k \rceil} | M_{j-k}) \\ &= I(M_j; X_{\leq j-k} | M_{j-k}) + I(M_j; X_{j-k+1..j-\lceil \alpha k \rceil} | M_{j-k}, X_{\leq j-k}) \\ &= I(M_j; X_{j-k+1..j-\lceil \alpha k \rceil} | X_{\leq j-k}). \end{aligned}$$

$I(M_j; X_{\leq j-k} | M_{j-k}) = 0$ as M_j is independent of $X_{\leq j-k}$ given the message sent by the $(j-k)$-th player M_{j-k}. Thus, (17) implies:

$$\sum_{j=1}^{n} I(M_j; X_{j-k+1..j-\lceil \alpha k \rceil} | X_{\leq j-k}) > \delta_2 n. \tag{18}$$

Let $k_1 := \lfloor n\varepsilon/3 \rfloor$, and $k_{i+1} := \lceil \alpha k_i \rceil$ for $i = 1..\ell$, where $\ell = \lfloor \log n / (2 \log 1/\alpha) \rfloor$. As $\alpha \geq c_5\varepsilon^5$ for a small enough constant $c_5 > 0$, $\alpha^{l+1}k_1 > 1$ for all $\varepsilon > c'n^{-1/20}$ for a large enough constant $c' > 0$.

We have, by the chain rule and (18),

$$\begin{aligned} \sum_{j=1}^{n} I(M_j; X_{\leq j}) &\geq \sum_{j=1}^{n} \sum_{i=1}^{\ell} I(M_j; X_{j-k_i+1..j-k_{i+1}} | X_{\leq j-k_i}) && \text{(Chain rule)} \\ &= \sum_{i=1}^{\ell} \sum_{j=1}^{n} I(M_j; X_{j-k_i+1..j-k_{i+1}} | X_{\leq j-k_i}) \\ &> \sum_{i=1}^{\ell} \delta_2 n \\ &= \delta_2 n\ell =: \delta n \log n, \end{aligned}$$

completing the proof. Here, $\delta \geq \delta_2/(4 \log 1/\alpha)$ (for $\varepsilon > c'n^{-1/20}$). Therefore, $\delta \geq c'''\varepsilon^5$ for a small enough constant $c''' > 0$. □

We next show that since $I(M_i; X_{\leq i}) \leq |M_i|$, Theorem 13 in fact implies that no deterministic streaming algorithm with $o(\log n)$ bits of memory can estimate the majority of n bits with a large enough constant advantage $1 - \gamma$, where, e.g., $\gamma = 1/1000$.

To do this, we use the next claim, Claim 6, to describe the advantage of a successful streaming algorithm. This advantage meets the requirement of Theorem 13, which then implies that if the streaming algorithm were to use $o(\log n)$ bits of memory, it would give an n-party communication protocol with total communication $o(n \log n)$, contradicting that the communication must be at least $\delta n \log n$ bits by Theorem 13.

Claim 6 (Computing Majority requires $\Omega(\log n)$ bits — Matching Advantages)**.** *Let M be a streaming algorithm, possibly using randomness, that computes the majority of n i.i.d. bits, $X_1, X_2, \ldots, X_n \sim Ber(1/2)$, with probability at least 999/1000 over the inputs (and the randomness). Then,*

$$\begin{aligned} \mathbb{E}_{M_n}\left[\mathbb{E}\left[\left(\sum_{i=1}^{n} X_i - n/2\right) \;\middle|\; M_n\right]^2\right] &= \Pr[M_n = 0] \cdot \mathbb{E}\left[\left(\sum_{i=1}^{n} X_i - n/2\right) \;\middle|\; M_n = 0\right]^2 \\ &\quad + \Pr[M_n = 1] \cdot \mathbb{E}\left[\left(\sum_{i=1}^{n} X_i - n/2\right) \;\middle|\; M_n = 1\right]^2 \\ &= \Omega(n) \end{aligned}$$

Proof. Given $x \in \{0,1\}^n$, let $p(x)$ be the probability that M outputs 1 on x. As M is correct with probability at least 999/1000, for large enough n,

$$\frac{1}{2^n}\left(\sum_{x:\sum_i x_i \geq n/2} p(x) + \sum_{x:\sum_i x_i < n/2} (1-p(x))\right) \geq 999/1000$$

$$\implies \frac{1}{2^n}\left(\sum_{x:\sum_i x_i > n/2} p(x) - \sum_{x:\sum_i x_i < n/2} p(x)\right) \geq 498/1000 \tag{19}$$

Next we show that

$$\mathbb{E}\left[\sum_{i=1}^n X_i - n/2 \middle| M_n = 1\right] = \Omega(\sqrt{n})$$

which proves the claim as $\Pr[M_n = 1] \geq 498/1000$ (using Equation (19)). We have:

$$\begin{aligned}\mathbb{E}\left[\sum_{i=1}^n X_i - n/2 \middle| M_n = 1\right] &= \frac{\sum_{x\in\{0,1\}^n} p(x)\left(\sum_{i=1}^n x_i - n/2\right)}{\sum_{x\in\{0,1\}^n} p(x)} \\ &\geq \frac{1}{2^n}\left(\sum_{x:\sum_i x_i \geq n/2} p(x)\left|\sum_{i=1}^n x_i - n/2\right| - \sum_{x:\sum_i x_i < n/2} p(x)\left|\sum_{i=1}^n x_i - n/2\right|\right).\end{aligned} \tag{20}$$

Let $y \in [0, n/2)$ be such that $497/1000 \leq \frac{1}{2^n}\left|\{x : n/2 \leq \sum_i x_i \leq n/2 + y\}\right| \leq 498/1000$ (such a y exists for all large enough n). Using a Chernoff bound, we can show that $\Pr_{X\in_R\{0,1\}^n}\left[\sum_i X_i \geq n/2 + \sqrt{\frac{3}{2}\ln\frac{1000}{2}}\sqrt{n}\right] \leq 2/1000$ and hence, $y \leq \sqrt{\frac{3}{2}\ln\frac{1000}{2}}\sqrt{n}$.

It is easy to see that, when satisfying Inequality (19), $\mathbb{E}\left[\sum_{i=1}^n X_i - n/2 | M_n = 1\right]$ (Expression (20)) is greater than the following quantity

$$\begin{aligned}&\frac{1}{2^n}\left(\sum_{x\ :\ n/2\leq\sum_i x_i \leq n/2+y}\left|\sum_{i=1}^n x_i - n/2\right| - \sum_{x\ :\ 0\leq\sum_i x_i < n/2-y}\left|\sum_{i=1}^n x_i - n/2\right|\right) \\ =&\frac{1}{2\cdot 2^n}\sum_{x\in\{0,1\}^n}\left|\sum_{i=1}^n x_i - n/2\right| - \frac{2}{2^n}\sum_{x\ :\ n/2+y<\sum_i x_i \leq n}\left|\sum_{i=1}^n x_i - n/2\right|.\end{aligned} \tag{21}$$

Before lower bounding $\mathbb{E}\left[\sum_{i=1}^n X_i - n/2 | M_n = 1\right]$ using the above expression, we need the following standard calculation of mean absolute deviation of a $\mathrm{Bin}(n, 1/2)$ random variable [5]:

$$\mathbb{E}_{x\in_R\{0,1\}^n}\left|\sum_{i=1}^n x_i - n/2\right| = \frac{1}{2^n}\cdot n \cdot \binom{n-1}{\lfloor n/2\rfloor} \geq \frac{n}{2\cdot 2^n}\binom{2\lfloor n/2\rfloor}{\lfloor n/2\rfloor}.$$

For the sake completeness, we also show the calculation in Appendix A. Thus, we have that

$$\begin{aligned}\frac{1}{2\cdot 2^n}\sum_{x\in\{0,1\}^n}\left|\sum_{i=1}^n x_i - n/2\right| &\geq \frac{n}{2}\frac{\binom{2\lfloor n/2\rfloor}{\lfloor n/2\rfloor}}{2\cdot 2^n} \\ &\geq \frac{n}{4}\frac{\sqrt{2\pi}\cdot 2^{2\lfloor n/2\rfloor+1/2}}{e^2\sqrt{\lfloor n/2\rfloor}\cdot 2^n} && \text{(Stirling's approximation; see Section 2)} \\ &\geq \frac{\sqrt{\pi}}{2\sqrt{2}e^2}\sqrt{n} && (2^{\frac{1}{2}}\cdot 2^{2\lfloor n/2\rfloor+1/2} \geq 2^n \text{ and } \lfloor n/2\rfloor \leq n/2)\end{aligned}$$

$$\geq 0.084\sqrt{n}. \tag{22}$$

Let

$$y' = \sqrt{\frac{3}{2}\ln\frac{1000}{2}}\sqrt{n}$$

and $p(c) = \Pr_{X \in_R \{0,1\}^n}\left[\sum_i X_i \geq n/2 + cy'\right]$. Using a Chernoff bound, $p(c) \leq (2/1000)^{c^2}$. Therefore,

$$\begin{aligned}
\frac{2}{2^n}\sum_{x\,:\,y<\sum_i x_i - n/2 \leq n/2}\left(\sum_{i=1}^{n} x_i - n/2\right) &\leq \frac{2}{2^n}\sum_{x\,:\,y<\sum_i x_i - n/2 \leq y'}\left(\sum_{i=1}^{n} x_i - n/2\right) \\
&\quad + 2\sum_{c=1}^{\sqrt{n}}(c+1)y'(p(c)-p(c+1)) \\
&\leq 2y'\left(\frac{3}{1000} + \sum_{c=1}^{\sqrt{n}}(c+1)(p(c)-p(c+1))\right) \\
&\leq 2y'\left(\frac{3}{1000} + p(1) + \sum_{c=1}^{\sqrt{n}} p(c)\right) \\
&\leq 2y'\left(\frac{5}{1000} + \sum_{c=1}^{\sqrt{n}}(2/1000)^{c^2}\right) \\
&\leq 2y'\left(\frac{5}{1000} + \frac{2}{998}\right) \\
&= 2\sqrt{\frac{3}{2}\ln\frac{1000}{2}}\left(\frac{5}{1000} + \frac{2}{998}\right)\sqrt{n} \\
&\leq 0.043\sqrt{n} \qquad (23)
\end{aligned}$$

Substituting Inequalities (22) and (23) in Expression (21) proves the claim. □

Thus far, our discussion has focused on deterministic streaming algorithms. Note that memory lower bounds for such algorithms suffice for proving lower bounds also for randomized streaming algorithms. This is because the input is distributional, so one can always fix the private randomness of the streaming algorithm.

Nevertheless, for our subsequent direct sum theorems, it will be useful to have a lower bound on the information cost (rather than just the memory) of streaming algorithms which use private randomness. The following corollary bounds the information cost and also works for streaming algorithms which use private randomness.

Corollary 14 (Information Lower Bound for Randomized Algorithms)**.** *For all $\varepsilon > c'n^{-\frac{1}{20}}$, there exists $\delta \geq c'''\epsilon^5$ (for small enough constant $c''' > 0$ and large enough constant $c' > 0$), such that if*

$$\sum_{i=1}^{n}\sum_{j=1}^{i} I(M_i; X_j | M_{j-1}) \leq \delta n \log n, \tag{24}$$

then

$$\mathbb{E}_{M_n}\left[\mathbb{E}\left[\left(\sum_{i=1}^{n} X_i - n/2\right) \;\middle|\; M_n\right]^2\right] \leq \varepsilon n. \tag{25}$$

Remark 1. *Next note that*

$$\begin{aligned}
\sum_{j=1}^{i} I(M_i;X_j|M_{j-1}) &\leq \sum_{j=1}^{i} I(M_i;X_j|X_{<j},M_{<j-1},M_{j-1}) \\
&\leq \sum_{j=1}^{i} I(M_i;X_j|X_{<j},M_{<j-1},M_{j-1}) + I(M_i;M_{j-1}|X_{<j},M_{<j-1}) \\
&= \sum_{j=1}^{i} (I(M_i;X_j,M_{j-1}|X_{<j},M_{<j-1}) \\
&= I(M_i;X_{\leq i},M_{<i}) && \text{(Chain rule)} \\
&\leq |M_i|,
\end{aligned}$$

so this measure of information does indeed lower bound the message size. The first inequality holds using Property 3 of mutual information (mentioned in Section 2), as

$$I(X_{<j},M_{<j-1};X_j|M_{j-1}) = 0.$$

As, $\sum_{j=1}^{i} I(M_i;X_j|M_{j-1}) \leq |M_i|$, Corollary 14 and Claim 6 together imply that no streaming algorithm that uses private randomness with $o(\log n)$ bits of memory can estimate the majority of n bits with a large enough constant advantage (999/1000). Indeed, such an algorithm would give rise to an n-player communication protocol with total communication $o(n \log n)$, contradicting that the communication must be at least $\delta n \log n$ bits, by the above theorem.

Proof of Corollary 14. The proof is similar to the proof of Theorem 13, and uses the following relation between $I(M_j;X_{j-k+1..j-\lceil \alpha k \rceil}|M_{j-k})$ and $\sum_{i=j-k+1}^{j-\lceil \alpha k \rceil} I(M_j;X_i|M_{i-1})$:

$$\begin{aligned}
I(M_j;X_{j-k+1..j-\lceil \alpha k \rceil}|M_{j-k}) &= \sum_{i=j-k+1}^{j-\lceil \alpha k \rceil} I(M_j;X_i|M_{j-k},X_{(j-k+1)..(i-1)}) && \text{(Chain rule)} \\
&\leq \sum_{i=j-k+1}^{j-\lceil \alpha k \rceil} I(M_j,M_{i-1};X_i|M_{j-k},X_{(j-k+1)..(i-1)}) \\
&= \sum_{i=j-k+1}^{j-\lceil \alpha k \rceil} I(M_j;X_i|M_{j-k},X_{(j-k+1)..(i-1)},M_{i-1}) \\
&\quad + \sum_{i=j-k+1}^{j-\lceil \alpha k \rceil} I(M_{i-1};X_i|M_{j-k},X_{(j-k+1)..(i-1)}) \\
&= \sum_{i=j-k+1}^{j-\lceil \alpha k \rceil} I(M_j;X_i|M_{j-k},X_{(j-k+1)..(i-1)},M_{i-1}) + 0 \\
&= \sum_{i=j-k+1}^{j-\lceil \alpha k \rceil} I(M_j;X_i|M_{i-1}).
\end{aligned}$$

The last equality follows (from Property 2 and 3 of mutual information) because

$$I(M_{j-k},X_{(j-k+1)..(i-1)};X_i|M_{i-1}) = 0$$

and

$$I(M_{j-k}, X_{(j-k+1)..(i-1)}; X_i | M_{i-1}, M_j) = 0.$$
$$\text{(conditioned on } M_{i-1},\, X_{\leq i-1}, M_{j-k} \text{ are independent of } X_i, M_j)$$

Let M be a streaming algorithm (possibly using private randomness) as in the statement of the corollary, such that (25) does not hold. That is,

$$\mathbb{E}_{M_n}\left[\mathbb{E}\left[\left(\sum_{i=1}^{n} X_i - n/2\right) \;\middle|\; M_n\right]^2\right] > \varepsilon n. \tag{26}$$

By Lemma 2 this implies that there exists $\delta_2 \geq c''\varepsilon^4$ and $\alpha \geq c_6\varepsilon^5$ ($c'', c_6 > 0$ are constants) such that for all integers $\alpha^{-1} < k \leq n\varepsilon/3$,

$$\sum_{j=1}^{n} I(M_j; X_{j-k+1..j-\lceil \alpha k \rceil} | M_{j-k}) > \delta_2 n. \tag{27}$$

The derived relation above implies:

$$\sum_{j=1}^{n} \sum_{i=j-k+1}^{j-\lceil \alpha k \rceil} I(M_j; X_i | M_{i-1}) > \delta_2 n. \tag{28}$$

Let $k_1 := \lfloor n\varepsilon/3 \rfloor$, and $k_{i+1} := \lceil \alpha k_i \rceil$ for $i = 1..\ell$, where $\ell = \lfloor \log n/(2\log 1/\alpha) \rfloor$. As $\alpha \geq c_5\varepsilon^5$ for a small enough constant $c_5 > 0$, $\alpha^{l+1}k_1 > 1$ for all $\varepsilon > c'n^{-1/20}$ for a large enough constant $c' > 0$.

We have, by (28),

$$\begin{aligned}
\sum_{j=1}^{n}\sum_{i=1}^{i} I(M_j; X_i | M_{i-1}) &\geq \sum_{j=1}^{n} \sum_{i=j-k_1+1}^{j-k_{l+1}} I(M_j; X_i | M_{i-1}) \\
&= \sum_{h=1}^{l}\sum_{j=1}^{n}\sum_{i=j-k_h+1}^{j-k_{h+1}} I(M_j; X_i | M_{i-1}) \\
&> \sum_{i=1}^{\ell} \delta_2 n \\
&= \delta_2 n\ell =: \delta n \log n,
\end{aligned}$$

completing the proof. Here, $\delta \geq \delta_2/(4\log 1/\alpha)$ (for $\varepsilon > c'n^{-1/20}$). Therefore, $\delta \geq c'''\varepsilon^5$ for a small enough constant $c''' > 0$. □

Next, we modify the proof of Corollary 14 after Equation (28) by taking $l = \lfloor \log n/(4\log 1/\alpha) \rfloor$ instead of $\lfloor \log n/(2\log 1/\alpha) \rfloor$, to conclude the following corollary.

Corollary 15. *Let M be a length-n streaming algorithm (possibly using private randomness). For all constants $\varepsilon > 0$, there exists $\delta > 0$ (for all $n \geq c_\varepsilon$, where $c_\varepsilon > 0$ is a large enough constant depending on ε), such that if*

$$\sum_{i=1}^{n} \sum_{j=1}^{i-2\sqrt{n}} I(M_i; X_j | M_{j-1}) \leq \delta n \log n, \tag{29}$$

then

$$\mathbb{E}_{M_n}\left[\mathbb{E}\left[\left(\sum_{i=1}^{n} X_i - n/2\right) \;\middle|\; M_n\right]^2\right] \leq \varepsilon n. \tag{30}$$

Proof. Let $l = \lfloor \log n/(4\log 1/\alpha) \rfloor$. For all constants $\varepsilon > 0$, independent of n and $\alpha = \Omega(\varepsilon^5)$, we have $\alpha^{l+1}k_1 \geq 2\sqrt{n}$ (for all $n \geq c_\varepsilon$, where $c_\varepsilon > 0$ is a large enough constant depending on ε). Here, k_i for $i = 1, ..., (\lfloor \log n/(4\log 1/\alpha) \rfloor + 1)$ is as defined in the proof of Corollary 14. Therefore, $k_{l+1} \geq 2\sqrt{n}$ and following the proof of Corollary 14, if Equation (30) doesn't hold, we get

$$\begin{aligned}\sum_{i=1}^{n}\sum_{j=1}^{i-2\sqrt{n}} I(M_i; X_j|M_{j-1}) &\geq \sum_{j=1}^{n}\sum_{i=j-k_1+1}^{j-k_{l+1}} I(M_j; X_i|M_{i-1}) \\ &= \sum_{h=1}^{l}\sum_{j=1}^{n}\sum_{i=j-k_h+1}^{j-k_{h+1}} I(M_j; X_i|M_{i-1}) \\ &> \sum_{i=1}^{\ell} \delta_2 n \\ &= \Omega(n\log n).\end{aligned}$$

The last expression follows because $l \geq \log n/(8\log 1/\alpha)$ for large enough n. □

3.4 k-Coins Problem

We now extend our lower bound results to solving more than one copy of the coin problem. Recall the problem: let $\{Y_i\}_{i=1}^{kn}$ be updates of the form (X_i, s_i), where $X_i \sim \mathrm{Ber}(1/2)$ and $s_i \in_R [k]$. Thus, k instances to the coin problem are interleaved in the input stream. We refer to the sequence $\{s_i\}_{i\in[nk]}$ as the *order* of the stream dictating how the k instances are interleaved. We consider two formulations of the k-Coins Problem. First, we show that for a fixed *good order*, any streaming algorithm solving a good fraction of the k instances of the coin problem requires $\Omega(k\log n)$ bits of memory. Second, we show that even when the order is random, that is $s_i \in_R [k], \forall i \in [nk]$, then any streaming algorithm solving k instances of the coin problem requires $\Omega(k\log n)$ bits of memory. Given an order $\{s_i\}_{i\in[nk]}$, for all $s \in [k]$, let $Z_s \subset [kn]$ be the set of i for which $s_i = s$. Let $q_s\colon [|Z_s|] \to [nk]$ be defined as follows: $q_s(i) = j$ if X_j is the i-th element corresponding to the s-th instance of the coin problem ($s_j = s$).

We next define the notion of a good order, which says that all coins have enough coin flips and that they are well-spaced throughout the interleaved stream.

Definition 3 (Good Order)**.** *An order* $\{s_i\}_{i\in[nk]}$ *is called* good *if for all* $s \in [k]$,

$$|Z_s| \geq n/2,$$

and for all $s \in [k], \sqrt{n} \leq t$ *and* $t < j \leq |Z_s|$,

$$q_s(j) - q_s(j-t) \geq \frac{k}{2}t.$$

Remark 2. *A random order* $\{s_i \in_R [k]\}_{i\in[nk]}$ *is a good order with probability at least* $1 - n^2k^3e^{-\sqrt{n}/6} - ke^{-n/8}$.

This can be shown using the following calculations. Fix w such that $\frac{\sqrt{n}}{2}k \leq w \leq nk$, $s \in [k]$ and $w \leq j' \leq nk$. Let $Z_1, ..., Z_w$ be *i.i.d.* $\mathrm{Ber}(1/k)$ random variables. Then,

$$\Pr_{\{s_i \in_R [k]\}_{i\in[nk]}}\left[|\{i \mid j'-w+1 \leq i \leq j' \text{ and } s_i = s\}| \geq \frac{2w}{k}\right] = \Pr\left[\sum_{i=1}^{w} Z_i \geq \frac{2w}{k}\right].$$

As $\mathbb{E}[\sum_{i=1}^{w} Z_i] = w/k$, using Chernoff bounds,

$$\Pr\left[\sum_{i=1}^{w} Z_i \geq \frac{2w}{k}\right] \leq e^{-\frac{w}{3k}} \leq e^{-\sqrt{n}/6}.$$

Therefore, by a union bound,

$$\Pr_{\{s_i \in_R [k]\}_{i\in[nk]}}\left[|\{i \mid j'-w+1 \le i \le j' \text{ and } s_i = s\}| < \frac{2w}{k},\ \forall\ \sqrt{n}k/2 \le w \le nk, s \in [k], w \le j' \le nk\right] \\ \ge 1 - n^2k^3e^{-\sqrt{n}/6}.$$

Note that for a fixed $s \in [k]$, $\sqrt{n} \le t$ and $t < j \le |Z_s|$, if $q_s(j) - q_s(j-t) < \frac{k}{2}t$, then for $w = kt/2$, $j' = q_s(j)$, $|\{i \mid j'-w+1 \le i \le j' \text{ and } s_i = s\}| \ge t$.

For the first part of the good order, again let $Z_1, ..., Z_{nk}$ be *i.i.d.* $\mathrm{Ber}(1/k)$ random variables. Then, for a fixed $s \in [k]$, using Chernoff bounds,

$$\Pr\left[|Z_s| < n/2\right] = \Pr\left[\sum_{i=1}^{nk} Z_i < n/2\right] \le e^{-\frac{n}{8}}. \tag{31}$$

Using a union bound, the claim in the remark follows.

Remark 3. *The order* $\{s_i\}_{i\in[nk]}$ *such that* $s_i = ((i-1) \mod k) + 1$*, is a good order.*

This remark follows since, for all $s \in [k]$, we have $|Z_s| = n$, and for all $\sqrt{n} \le t$ and $t < j \le n$, $q_s(j) - q_s(j-t) = kt$.

We prove the following variance-information tradeoff for any streaming algorithm that is given k instances of the coin problem as a stream of nk updates of the form (X_i, s_i) where $X_i \sim \mathrm{Ber}(1/2), \forall i \in [nk]$ and $\{s_i\}_{i\in[nk]}$ is a fixed *good order*.

Theorem 16. *Let M be an nk-length streaming algorithm (that possibly uses private randomness) for the k-Coins Problem with a fixed good order $\{s_i\}_{i\in[nk]}$ (Definition 3). For all constants $\varepsilon > 0$, there exists $\delta > 0$, such that if*

$$\mathbb{E}_{M_{nk}}\left[\mathbb{E}\left[\left(\sum_{i\in Z_s} X_i - |Z_s|/2\right) \;\middle|\; M_{nk}\right]^2\right] > \varepsilon|Z_s| \tag{32}$$

for at least $1/2$ fraction of $s \in [k]$, then

$$\sum_{i=1}^{nk}\sum_{j=1}^{i} I(M_i; X_j|M_{j-1}) > \delta nk^2 \log n \tag{33}$$

Proof. We prove the theorem using a reduction to Corollary 15. Let

$$\mathcal{S} = \{s \mid s \in [k] \text{ and Equation (32) holds for } s\}.$$

Thus, $|\mathcal{S}| \ge k/2$. For all $s \in \mathcal{S}$, we prove that

$$\sum_{i=1}^{nk}\sum_{j\in[i]\cap Z_s} I(M_i; X_j|M_{j-1}) > \Omega(nk\log n) \tag{34}$$

and as

$$\begin{aligned}\sum_{i=1}^{nk}\sum_{j=1}^{i} I(M_i; X_j|M_{j-1}) &= \sum_{i=1}^{nk}\sum_{s\in[k]}\sum_{j\in[i]\cap Z_s} I(M_i; X_j|M_{j-1}) \\ &\ge \sum_{i=1}^{nk}\sum_{s\in\mathcal{S}}\sum_{j\in[i]\cap Z_s} I(M_i; X_j|M_{j-1})\end{aligned}$$

$$= \sum_{s \in \mathcal{S}} \sum_{i=1}^{nk} \sum_{j \in [i] \cap Z_s} I(M_i; X_j | M_{j-1}),$$

the theorem follows. To prove Equation (34), for all $s \in \mathcal{S}$, using M, we construct a $|Z_s|$-length streaming algorithm M^s for the coin problem as follows. Given a total of $|Z_s| \geq n/2$ *i.i.d.* uniform $\{0,1\}$ bits $X'_1, ..., X'_{|Z_s|}$ in a stream, M^s generates k instances of the coin problem by embedding $X'_1, ..., X'_{|Z_s|}$ as the s-th instance and using private randomness to generate the bits for the remaining $k-1$ instances. At the i-th step, M^s reads X'_i, simulates M for layers $q_s(i)$ to $q_s(i+1)-1$, and sends $M_{q_s(i+1)-1}$ as the message to the next step. As $\{s_i\}_{i \in [nk]}$ is fixed, M^s, at the i-th step, can generate the input to M for layers $q_s(i)$ to $q_s(i+1)-1$ ($(X_j, s_j)_{j \in \{q_s(i), ..., q_s(i+1)-1\}}$) using only private randomness and X'_i ($\{s_i\}_{i \in [nk]}$ and M are hardwired in the mapping $\mathcal{M}^s_{i+1}$). Therefore, $\forall\, 0 < i < |Z_s|, M^s_i = M_{q_s(i+1)-1}$ and $M^s_{|Z_s|} = M_{nk}$. Also,

$$\mathbb{E}_{M^s_{|Z_s|}} \left[\mathbb{E}\left[\left(\sum_{i=1}^{|Z_s|} X'_i - |Z_s|/2 \right) \;\middle|\; M^s_{|Z_s|} \right]^2 \right] = \mathbb{E}_{M_{nk}} \left[\mathbb{E}\left[\left(\sum_{i \in Z_s} X_i - |Z_s|/2 \right) \;\middle|\; M_{nk} \right]^2 \right] > \varepsilon |Z_s|.$$

Therefore, Corollary 15 implies that, for large enough n and because $|Z_s| \geq n/2$ and $2\sqrt{|Z_s|} > \sqrt{n}$:

$$\sum_{i=1}^{|Z_s|} \sum_{j=1}^{i-\sqrt{n}} I(M^s_i; X'_j | M^s_{j-1}) > \Omega(n \log n). \tag{35}$$

Note that (assuming $q_s(|Z_s|+1) = nk+1$)

$$\sum_{i=1}^{|Z_s|} \sum_{j=1}^{i-\sqrt{n}} I(M^s_i; X'_j | M^s_{j-1}) = \sum_{i=1}^{|Z_s|} \sum_{j=1}^{i-\sqrt{n}} I(M_{q_s(i+1)-1}; X_{q_s(j)} | M_{q_s(j)-1}) \tag{36}$$

Next, we analyze the quantity $\sum_{i=1}^{|Z_s|} \sum_{j=1}^{i-\sqrt{n}} I(M_{q_s(i+1)-1}; X_{q_s(j)} | M_{q_s(j)-1})$. First, we prove the following fact, which is a data processing inequality for streaming:

Fact 1. *Let M be a streaming algorithm using only private randomness. Then for all $i_1 \geq i_2 \geq j > 0$,*

$$I(M_{i_1}; X_j | M_{j-1}) \leq I(M_{i_2}; X_j | M_{j-1}).$$

Proof. Using the fact that conditioned on M_{i_2}, M_{i_1} is independent of X_j, M_{j-1}, the proof follows as below.

$$\begin{aligned} I(M_{i_1}; X_j | M_{j-1}) &\leq I(M_{i_2}, M_{i_1}; X_j | M_{j-1}) \\ &= I(M_{i_2}; X_j | M_{j-1}) + I(M_{i_1}; X_j | M_{j-1}, M_{i_2}) \\ &= I(M_{i_2}; X_j | M_{j-1}) \end{aligned} \qquad \square$$

Because, $q_s(i+1) - 1 \geq q_s(i)$, we get

$$\begin{aligned} \sum_{i=1}^{|Z_s|} \sum_{j=1}^{i-\sqrt{n}} I(M_{q_s(i+1)-1}; X_{q_s(j)} | M_{q_s(j)-1}) &\leq \sum_{i=1}^{|Z_s|} \sum_{j=1}^{i-\sqrt{n}} I(M_{q_s(i)}; X_{q_s(j)} | M_{q_s(j)-1}) \\ &= \sum_{j=1}^{|Z_s|} \sum_{i=j+\sqrt{n}}^{|Z_s|} I(M_{q_s(i)}; X_{q_s(j)} | M_{q_s(j)-1}). \end{aligned}$$

Next, we prove that

$$\sum_{j=1}^{|Z_s|} \sum_{i=j+\sqrt{n}}^{|Z_s|} I(M_{q_s(i)}; X_{q_s(j)} | M_{q_s(j)-1}) \leq \frac{2}{k} \sum_{j \in Z_s} \sum_{i=j}^{nk} I(M_i; X_j | M_{j-1}), \tag{37}$$

which along with Equation (35) and (36), implies Equation (34).

In fact, we prove that $\forall j \in [|Z_s|]$,

$$\sum_{i=j+\sqrt{n}}^{|Z_s|} I(M_{q_s(i)}; X_{q_s(j)} | M_{q_s(j)-1}) \leq \frac{2}{k} \sum_{i=q_s(j)}^{nk} I(M_i; X_{q_s(j)} | M_{q_s(j)-1}) \tag{38}$$

and summing over j gives Equation (37). Fix $j \in [|Z_s|]$. Let

$$\mathcal{I}_1(g) = \{i \mid j + \sqrt{n} \leq i \leq |Z_s| \text{ and } I(M_{q_s(i)}; X_{q_s(j)} | M_{q_s(j)-1}) \geq g\}.$$

Similarly, let

$$\mathcal{I}_2(g) = \{i \mid q_s(j) \leq i \leq nk \text{ and } I(M_i; X_{q_s(j)} | M_{q_s(j)-1}) \geq g\}.$$

Claim 7. *For all $g \in [0,1]$, $|\mathcal{I}_1(g)| \leq \frac{2}{k}|\mathcal{I}_2(g)|$.*

Proof. Let i_1 be the largest index in the set $\mathcal{I}_1$ (assuming $\mathcal{I}_1 \neq \emptyset$, otherwise the claim is trivially true). By definition, $i_1 - j \geq \sqrt{n}$. Using Fact 1, $\forall i : q_s(j) \leq i \leq q_s(i_1)$,

$$g \leq I(M_{q_s(i_1)}; X_{q_s(j)} | M_{q_s(j)-1}) \leq I(M_i; X_{q_s(j)} | M_{q_s(j)-1}).$$

As the order $\{s_i\}_{i \in [nk]}$ is *good*, by Definition 3,

$$|\mathcal{I}_2(g)| = q_s(i_1) - q_s(j) + 1 \geq \frac{k}{2}(i_1 - j) > \frac{k}{2}(i_1 - (j + \sqrt{n}) + 1) = \frac{k}{2}|\mathcal{I}_1(g)|$$

□

Let $u > 0$ be a positive integer. Then for all u,

$$\sum_{g=1}^{u} |\mathcal{I}_1(g/u)| \cdot \frac{1}{u} = \sum_{i=j+\sqrt{n}}^{|Z_s|} \lfloor u \cdot I(M_{q_s(i)}; X_{q_s(j)} | M_{q_s(j)-1}) \rfloor \cdot \frac{1}{u}.$$

Therefore for all u,

$$\sum_{g=1}^{u} |\mathcal{I}_1(g/u)| \cdot \frac{1}{u} \leq \sum_{i=j+\sqrt{n}}^{|Z_s|} I(M_{q_s(i)}; X_{q_s(j)} | M_{q_s(j)-1})$$

and

$$\sum_{i=j+\sqrt{n}}^{|Z_s|} I(M_{q_s(i)}; X_{q_s(j)} | M_{q_s(j)-1}) - \sum_{i=j+\sqrt{n}}^{|Z_s|} \frac{1}{u} \leq \sum_{g=1}^{u} |\mathcal{I}_1(g/u)| \cdot \frac{1}{u}. \tag{39}$$

Similarly, we can show that

$$\sum_{g=1}^{u} |\mathcal{I}_2(g/u)| \cdot \frac{1}{u} \leq \sum_{i=q_s(j)}^{nk} I(M_i; X_{q_s(j)} | M_{q_s(j)-1}) \tag{40}$$

Using Claim 7 together with Equations (39),(40), we get

$$\sum_{i=j+\sqrt{n}}^{|Z_s|} I(M_{q_s(i)}; X_{q_s(j)} | M_{q_s(j)-1}) - \sum_{i=j+\sqrt{n}}^{|Z_s|} \frac{1}{u} \leq \frac{2}{k} \sum_{i=q_s(j)}^{nk} I(M_i; X_{q_s(j)} | M_{q_s(j)-1}).$$

As u can be made arbitrarily large, Equation (38) follows, and this completes the proof. □

Next, using Theorem 16, we prove the following corollary.

Corollary 17. *Fix a* good order $\{s_i\}_{i\in[nk]}$. *Suppose we are given a sequence of kn i.i.d. stream updates, the j-th of which has the form* $Y_j = (X_j, s_j)$, *where* X_j *is chosen uniformly in* $\{0,1\}$. *We interpret this as k independent instances of the coin problem, where the s-th instance of the coin problem consists of the sequence of bits* $X_{q_s(i)}, \dots, X_{q_s(|Z_s|)}$. *Suppose there is a streaming algorithm M which, given an* $\ell \in_R [k]$ *at the end of the stream, outputs the majority bit of the* ℓ*-th instance of the coin problem with probability at least* $1 - \frac{1}{2000}$. *Then the algorithm uses* $\Omega(k \log n)$ *bits of memory.*

Here, the probability of success is over the input stream, the private randomness used by M, and $\ell \in_R [k]$. When the order $\{s_i\}_{i\in[nk]}$ is such that $s_i = ((i-1) \mod k) + 1$ (which is a good order — Remark 3), then we refer to the above defined k-Coins Problem for this order as the Simultaneous k-Coins Problem.

Proof. We let M_{nk} represent the memory state of the the streaming algorithm after seeing the nk updates. Let M^ℓ be the memory state after reading the index (ℓ) at the end of the stream. As M succeeds with probability at least $1 - 1/2000$ in outputting the majority of the ℓ-th instance, when $\ell \in_R [k]$, then for at least a $1/2$ fraction of $\ell \in [k]$, M succeeds in outputting the majority with probability at least $1 - 1/1000$, where the probability is over the input stream and the private randomness used by M. Let $\mathcal{L}$ be the set of indices ℓ for which the last statement is true. Then, for all $\ell \in \mathcal{L}$, Claim 6 implies that (considering all the other $k-1$ instances as randomness):

$$\mathbb{E}_{M^\ell}\left[\mathbb{E}\left[\left(\sum_{i=1}^{|Z_\ell|} X_{q_\ell(i)} - |Z_\ell|/2\right) \,\middle|\, M^\ell\right]^2\right] = \Omega(|Z_\ell|).$$

Note that

$$\begin{aligned}\mathbb{E}_{M^\ell}\left[\mathbb{E}\left[\left(\sum_{i=1}^{|Z_\ell|} X_{q_\ell(i)} - |Z_\ell|/2\right) \,\middle|\, M^\ell\right]^2\right] &\le \mathbb{E}_{M^\ell, M_{nk}}\left[\mathbb{E}\left[\left(\sum_{i=1}^{|Z_\ell|} X_{q_\ell(i)} - |Z_\ell|/2\right) \,\middle|\, M^\ell, M_{nk}\right]^2\right] \\ &= \mathbb{E}_{M_{nk}}\left[\mathbb{E}\left[\left(\sum_{i=1}^{|Z_\ell|} X_{q_\ell(i)} - |Z_\ell|/2\right) \,\middle|\, M_{nk}\right]^2\right].\end{aligned}$$

The last equality follows because conditioned on M_{nk}, $\sum_{i=1}^{|Z_\ell|} X_{q_\ell(i)}$ is independent of M^ℓ. Therefore, for at least a half fraction of $\ell \in [k]$, Equation (32) holds, and thus, using Theorem 16,

$$\sum_{i=1}^{nk}\sum_{j=1}^{i} I(M_i; X_j | M_{j-1}) \ge \Omega(nk^2 \log n).$$

Using Remark 1, $\sum_{i=1}^{nk} |M_i| \ge \Omega(nk^2 \log n)$, and hence, M uses at least $\Omega(k \log n)$ bits of memory. □

Next, we show that any streaming algorithm that solves the k-Coins Problem for a random order $\{s_i \in_R [k]\}_{i\in[nk]}$ requires $\Omega(k \log n)$ memory.

Corollary 18 (Formal statement of Theorem 2)**.** *Suppose we are given a sequence of kn i.i.d. stream updates, the j-th of which has the form* $Y_j = (X_j, s_j)$, *where* X_j *is chosen uniformly in* $\{0,1\}$ *and* s_j *is chosen uniformly in* $[k]$. *We interpret this as k independent instances of the coin problem, where the s-th instance of the coin problem consists of the sequence of bits* $X_{j_1}, \dots, X_{j_r}$, *if and only if* $s_{j_t} = s, \forall 1 \le t \le r$. *Suppose there is a streaming algorithm M which, given an* $\ell \in_R [k]$ *at the end of the stream, outputs the majority bit of the* ℓ*-th instance of the coin problem with probability at least* $1 - \frac{1}{4000}$. *Then the algorithm uses* $\Omega(k \log n)$ *bits of memory (for* $k \le 2^{n^{0.1}}$*).*

Here, the probability of success is over the updates, including the random order of the stream, $\ell \in_R [k]$ and the private randomness used by M.

Proof. As M succeeds with at least $1 - 1/4000$ overall probability, for at least a 1/2-fraction of the orders $\{s_i \in_R [k]\}_{i\in[nk]}$, M succeeds in outputting the majority with probability at least $1 - 1/2000$, where the probability is over $\{X_i\}_{i\in[nk]}$, ℓ, and the private randomness. Using Remark 2, with probability at least

$$1 - n^2k^3e^{-\sqrt{n}/6} - ke^{-n/8} \geq 1 - \frac{1}{4n}, \qquad \text{(for } k \leq 2^{n^{0.1}} \text{ and large enough } n\text{)}$$

a random order is a *good order*. Therefore, there exists a good order $\{s_i\}_{i\in[nk]}$, such that M solves the k-Coins Problem for that order with probability at least $1 - \frac{1}{2000}$. Using Corollary 17, M requires $\Omega(k \log n)$ bits of memory. □

3.5 Lower Bounds for the Gap Coin Problem

In this section we consider the following problem: given a sequence of n $\{-1, +1\}$ bits X_1, X_2,..., X_n, output 0 if $|\sum_i X_i| \leq 4\sqrt{n\alpha}$ and 1 if $|\sum_i X_i| \geq 4\sqrt{n\beta}$. We refer to this as the GapCoin(α, β) problem. Recall U_n represents the uniform distribution over $\{-1, +1\}^n$.

In Theorem 19, we use a hybrid argument along with Lemma 3, which might be of independent interest, to prove hardness of solving the GapCoin(α, β) problem for when β/α can be as large as poly$(\log n)$.

We in fact show the hardness of solving the GapCoin(α, β) problem, even under the promise that the sequence of n bits X_1,..., X_n satisfy the following property for $\gamma = 10\beta \log \beta$.

Promise 1 (Input is balanced at all times)**.** *For $\gamma > 0$, sequence of n $\{-1, +1\}$ bits $x_1, x_2, \ldots, x_n$ is γ-balanced at all times if for $i \in \{1, ..., n\}$,*

$$\left|\sum_{i_1=1}^{i} x_{i_1}\right| \leq \gamma\sqrt{n}$$

Theorem 19. *Let M be a streaming algorithm (that might use private randomness) which, under the promise that the input stream $x_1, ..., x_n \in \{-1, +1\}^n$ satisfies Promise 1 for $\gamma = 10\beta \log \beta$, outputs, with probability at least 2/3 over the private randomness, the answer 0 on every input with $|\sum_i x_i| \leq 4\sqrt{n\alpha}$ and 1 on every input with $|\sum_i x_i| > 4\sqrt{n\beta}$. Then there exists a constant $\delta > 0$ (for $\alpha \geq 1$ and $c_2 \leq \beta \leq$ poly$(\log n)$, where $c_2 > 0$ is a large enough constant) such that,*

$$\sum_{i=1}^{n}\sum_{j=1}^{i} I(M_i; X_j|M_{j-1}) > \delta n \log n/(\beta \log^2 \beta)^{11},$$

where X is drawn from U_n.

We will use the following lemma to prove the theorem above. Given a streaming algorithm B (possibly using private randomness) on n' bits, let

$$IC(B, U_{n'}) = \sum_{i=1}^{n'}\sum_{j=1}^{i} I(B_i; X_j|B_{j-1})$$

when X is drawn from $U_{n'}$.

Lemma 3 (Formal statement of Theorem 12)**.** *Let B be a (probabilistic) streaming algorithm of block of length m. Let $0 < \epsilon < \frac{1}{30}$. Then, there exists a universal constant $c_0 > 0$, such that one of two things holds:*

1. *$IC(B, U_m) > c_0 \cdot m(\log m) \cdot \epsilon^{10}$.*
2. *There exists a distribution μ on m bits, such that*
   - *$\forall x \in \{-1, +1\}^m, \mu(x) \leq 2U_m(x)$,*

- $E_{x\sim\mu}[\sum_i x_i] \geq \Omega\left(\sqrt{\frac{m}{\log(1/\epsilon)}}\right)$, *and*
- $\|B(U_m) - B(\mu)\|_1 < \epsilon$.

Proof. Let $d \geq 1$ be such that

$$\mathbb{E}_{x\sim U_m}\left[\frac{|\sum_i x_i|}{d\sqrt{m}} \cdot 1_{|\sum_i x_i| \geq d\sqrt{m}}\right] \leq \frac{\epsilon}{5}.$$

We prove that $d = O(\sqrt{\log(1/\epsilon)})$. Using a Chernoff bound, for $c > 0$, we have that

$$\Pr_{x\sim U_m}\left[|\sum_i x_i| \geq c \cdot d\sqrt{m}\right] \leq 2e^{-c^2\frac{d^2}{6}}.$$

Let $p(c) = \Pr_{x\sim U_m}[|\sum_i x_i| \geq c \cdot d\sqrt{m}]$. Then,

$$\begin{aligned}\mathbb{E}_{x\sim U_m}\left[\frac{|\sum_i x_i|}{d\sqrt{m}} \cdot 1_{|\sum_i x_i| \geq d\sqrt{m}}\right] &\leq \sum_{c=1}^{\sqrt{m}}(c+1)\Pr_{x\sim U_m}\left[cd\sqrt{m} \leq |\sum_i x_i| \leq (c+1)d\sqrt{m}\right] \\ &= \sum_{c=1}^{\sqrt{m}}(c+1)\left(p(c) - p(c+1)\right) \\ &= p(1) + \sum_{c=1}^{\sqrt{m}} p(c) \leq \sum_{c=1}^{\sqrt{m}} 2p(c) \leq \sum_{c=1}^{\sqrt{m}} 4e^{-c^2\frac{d^2}{6}}\end{aligned}$$

It is easy to see that for $\epsilon \leq 1/30$ and $d = 6\sqrt{\log(1/\epsilon)}$, $\sum_{c=1}^{\sqrt{m}} 4e^{-c^2\frac{d^2}{6}} \leq \epsilon/5$.

Definition of μ: Let trunc : $\mathbb{R} \to [-1,1]$ be defined as trunc$(p) = \max(\min(p,1),-1)$. Next, we define the distribution μ as follows:

$$\mu(x) := U_m(x) \cdot \left(1 + \operatorname{trunc}\left(\frac{\sum_i x_i}{d\sqrt{m}}\right)\right).$$

It is easy to see that $\forall x,\ \mu(x) \leq 2U_m(x)$.

Lower bound on $E_{x\sim\mu}[\sum_i x_i]$:

$$\begin{aligned}E_{x\sim\mu}[\sum_i x_i] &= \sum_{x\in\{-1,1\}^m} U_m(x) \cdot \left(1 + \operatorname{trunc}\left(\frac{\sum_i x_i}{d\sqrt{m}}\right)\right) \cdot \left(\sum_i x_i\right) \\ &= \sum_{x\in\{-1,1\}^m} U_m(x) \cdot \operatorname{trunc}\left(\frac{\sum_i x_i}{d\sqrt{m}}\right) \cdot \left(\sum_i x_i\right) \\ &\geq \mathbb{E}_{x\sim U_m}\left[\frac{(\sum_i x_i)^2}{d\sqrt{m}} \cdot 1_{|\sum_i x_i| \leq d\sqrt{m}}\right]\end{aligned}$$

As $\mathbb{E}_{x\sim U_m}\left[\frac{(\sum_i x_i)^2}{d\sqrt{m}} \cdot 1_{|\sum_i x_i| \leq d\sqrt{m}}\right] + \mathbb{E}_{x\sim U_m}\left[\frac{(\sum_i x_i)^2}{d\sqrt{m}} \cdot 1_{|\sum_i x_i| > d\sqrt{m}}\right] = \mathbb{E}_{x\sim U_m}\left[\frac{(\sum_i x_i)^2}{d\sqrt{m}}\right]$,

$$\mathbb{E}_{x\sim U_m}\left[\frac{(\sum_i x_i)^2}{d\sqrt{m}} \cdot 1_{|\sum_i x_i| \leq d\sqrt{m}}\right] = \frac{\sqrt{m}}{d} - \mathbb{E}_{x\sim U_m}\left[\frac{(\sum_i x_i)^2}{d\sqrt{m}} \cdot 1_{|\sum_i x_i| > d\sqrt{m}}\right] \tag{41}$$

We show that $\mathbb{E}_{x\sim U_m}\left[\frac{(\sum_i x_i)^2}{d\sqrt{m}}\cdot 1_{|\sum_i x_i|>d\sqrt{m}}\right]\leq \epsilon\frac{\sqrt{m}}{d}$ for $d=6\sqrt{\log(1/\epsilon)}$ and $\epsilon\leq\frac{1}{30}$. Substituting in Equation (41), we get $\mathbb{E}_{x\sim U_m}\left[\frac{(\sum_i x_i)^2}{d\sqrt{m}}\cdot 1_{|\sum_i x_i|\leq d\sqrt{m}}\right]\geq\Omega\left(\sqrt{\frac{m}{\log(1/\epsilon)}}\right)$.

$$\begin{aligned}
\mathbb{E}_{x\sim U_m}\left[\frac{(\sum_i x_i)^2}{d\sqrt{m}}\cdot 1_{|\sum_i x_i|>d\sqrt{m}}\right] &\leq \sum_{c=1}^{\sqrt{m}}(c+1)^2 d\sqrt{m}\Pr_{x\sim U_m}\left[cd\sqrt{m}\leq|\sum_i x_i|\leq(c+1)d\sqrt{m}\right]\\
&= d\sqrt{m}\sum_{c=1}^{\sqrt{m}}(c+1)^2\left(p(c)-p(c+1)\right)\\
&= d\sqrt{m}\left(4p(1)+\sum_{c=2}^{\sqrt{m}}(2c+1)p(c)\right)\leq 4d\sqrt{m}\sum_{c=1}^{\sqrt{m}}cp(c)\\
&\leq 4d\sqrt{m}\sum_{c=1}^{\sqrt{m}}ce^{-c^2\frac{d^2}{6}}\leq 4d\sqrt{m}\frac{e^{-\frac{d^2}{6}}}{(1-e^{-\frac{d^2}{6}})^2}\\
&= 4d\sqrt{m}\frac{\epsilon^6}{(1-\epsilon^6)^2}\leq\epsilon\frac{\sqrt{m}}{d}
\end{aligned}$$

The last inequality follows from the fact that $6\epsilon\sqrt{\log(1/\epsilon)}\leq 1$ for $\epsilon\leq 1/30$.

Upper bound on $\|B(U_m)-B(\mu)\|_1$: We calculate $\|B(U_m)-B(\mu)\|_1$ given that $IC(B,U_m)<c_0\cdot m\log m\cdot\epsilon^{10}$ (the following upper bound works for any starting distribution of B_0).

$$\begin{aligned}
\|B(U_m)-B(\mu)\|_1 &= \sum_v\left|\sum_{x\in\{-1,1\}^m}(U_m(x)-\mu(x))\cdot\Pr[B(x)=v]\right|\\
&= \sum_v\left|\sum_{x\in\{-1,1\}^m}U_m(x)\cdot\text{trunc}\left(\frac{\sum_i x_i}{d\sqrt{m}}\right)\cdot\Pr[B(x)=v]\right|\\
&\leq \sum_v\left|\sum_{x\in\{-1,1\}^m}U_m(x)\cdot\frac{\sum_i x_i}{d\sqrt{m}}\cdot\Pr[B(x)=v]\right|\\
&\quad+\sum_v\left|\sum_{x\in\{-1,1\}^m}U_m(x)\cdot\left(\text{trunc}\left(\frac{\sum_i x_i}{d\sqrt{m}}\right)-\frac{\sum_i x_i}{d\sqrt{m}}\right)\cdot\Pr[B(x)=v]\right|\\
&\leq \sum_v\left|\sum_{x\in\{-1,1\}^m}U_m(x)\cdot\frac{\sum_i x_i}{d\sqrt{m}}\cdot\Pr[B(x)=v]\right|\\
&\quad+\sum_{x\in\{-1,1\}^m}U_m(x)\cdot\left|\text{trunc}\left(\frac{\sum_i x_i}{d\sqrt{m}}\right)-\frac{\sum_i x_i}{d\sqrt{m}}\right|\cdot\left(\sum_v\Pr[B(x)=v]\right)\\
&\leq \sum_v\left|\sum_{x\in\{-1,1\}^m}U_m(x)\cdot\frac{\sum_i x_i}{d\sqrt{m}}\cdot\Pr[B(x)=v]\right|+\mathbb{E}_{x\sim U_m}\left[\frac{|\sum_i x_i|}{d\sqrt{m}}\cdot 1_{|\sum_i x_i|>d\sqrt{m}}\right]\\
&\leq \frac{1}{d\sqrt{m}}\cdot\mathbb{E}_{v\sim B_m}\left|\mathbb{E}_{x\sim U_m}\left[\sum_i x_i\;\middle|\;B(x)=v\right]\right|+\frac{\epsilon}{5}
\end{aligned}$$

$$\leq \frac{1}{d\sqrt{m}} \cdot \sqrt{\mathbb{E}_{v\sim B_m}\left(\mathbb{E}_{x\sim U_m}\left[\sum_i x_i \;\middle|\; B(x)=v\right]\right)^2} + \frac{\epsilon}{5}$$

Corollary 14 shows that if $IC(B,U_m) < \delta' \cdot m\log m$, then

$$\mathbb{E}_{v\sim B_m}\left[\mathbb{E}_{x\sim U_m}\left[\sum_{i=1}^{n} x_i \;\middle|\; B(x)=v\right]^2\right] \leq 4\epsilon' m$$

for $\delta' = c'''\epsilon'^5$ where $c''' > 0$ is a small enough constant.

Therefore, for $\delta' = c'''\left(\frac{\epsilon^2}{16}\right)^5$, if $IC(B,U_m) < \delta' \cdot m\log m$, then

$$\mathbb{E}_{v\sim B_m}\left[\mathbb{E}_{x\sim U_m}\left[\sum_{i=1}^{n} x_i \;\middle|\; B(x)=v\right]^2\right] \leq \frac{1}{4}\epsilon^2 m$$

and hence, for $c_0 = c'''/16^5$,

$$\|B(U_m) - B(\mu)\|_1 \leq \frac{1}{d\sqrt{m}}\sqrt{\frac{1}{4}\epsilon^2 m} + \frac{\epsilon}{5} < \epsilon$$

□

Proof of Theorem 19. Let $\delta = \frac{c_0}{2\cdot 10^{10}}$. Let M be a streaming algorithm which outputs 0 with probability at least 2/3 on every input with $|\sum_i X_i| \leq 4\sqrt{n\alpha}$ and 1 with probability at least 2/3 on every input with $|\sum_i X_i| > 4\sqrt{n\beta}$, when the input satisfies Promise 1 with $\gamma = 10\beta\log\beta$, such that $I(M,U_n) < \delta n\log n/(\beta\log^2\beta)^{11}$.

Partition the stream of length n into $t = \beta\log^2\beta$ blocks of length $m = n/t$ (as $\beta \leq poly(\log n)$, $m \geq \sqrt{n}$). Define a sequence of distributions $D_i = D_{i,1} \times D_{i,2} \times \ldots \times D_{i,t}, i \in \{0,\ldots,t\}$ on n bits as follows:

$$D_{i,l} = U_m, \forall 1 \leq l \leq t-i \text{ and } D_{i,l} = \mu, \forall t-i < l \leq t$$

Using Lemma 3 with parameters $m = n/t$ and $\epsilon = \frac{1}{10t}$ (Corollary 14 works for this regime of ϵ as $\epsilon = \Omega(1/poly(\log n))$), we first show that

$$\|M(D_t) - M(D_0)\|_1 \leq \frac{1}{10}.$$

Let B^i ($i \in \{1,\ldots,t\}$) be a (probabilistic) streaming algorithm of block of length m equivalent to subprogram of M on $(i-1)\cdot m+1$ to $i\cdot m$ bits ($B^i_j = M_{(i-1)\cdot m+j}$ ($j \in [m]$), B^i_0 is a random variable independent of the input to B^i equivalent to $M_{(i-1)\cdot m}$ under the uniform distribution over the first $(i-1)\cdot m$ bits).

$$\begin{aligned} I(M,U_n) = \sum_{i=1}^{n}\sum_{j=1}^{i}(I(M_i;X_j|M_{j-1})) &\geq \sum_{i=(l-1)\cdot m+1}^{l\cdot m}\;\sum_{j=(l-1)\cdot m+1}^{i}(I(M_i;X_j|M_{j-1})) \\ &= I(B^l,U_m) \end{aligned}$$

Recall, $\delta = \frac{c_0}{2\cdot 10^{10}}$ and $\epsilon = \frac{1}{10t}$. Therefore,

$$I(B^l,U_m) \leq \frac{\delta n\log n}{(\beta\log^2\beta)^{11}} \leq \frac{c_0}{2\cdot 10^{10}} \cdot mt \cdot 2\log m \cdot \frac{1}{t^{11}} = c_0 \cdot m\log m \cdot \epsilon^{10}$$

and through Lemma 3, we have that $\|B^l(U_m) - B^l(\mu)\|_1 \leq \epsilon$, where B_0^l is $M_{(l-1)\cdot m}$ under the uniform distribution over the first $(l-1)\cdot m$ bits and the private randomness used.

By definition, both D_{t-l+1} and D_{t-l} have uniform distribution over the first $(l-1)\cdot m$ bits. Since B_0^l is $M_{(l-1)\cdot m}$ under the uniform distribution over the first $(l-1)\cdot m$, $\|M_{lm}(D_{t-l+1}) - M_{lm}(D_{t-l})\|_1 \leq \epsilon$ (where M_{lm} represents the lm-th message). Since the last $(t-l)m$ bits are drawn from the same distribution for D_{t-l+1} and D_{t-l}, the statistical distance of the output cannot increase. Hence, $\|M(D_{t-l+1}) - M(D_{t-l})\|_1 \leq \epsilon$ and by the triangle inequality, $\|M(D_t) - M(D_0)\|_1 \leq \frac{1}{10}$.

Next, we prove that if M succeeds with the probability mentioned above conditioned on the promise, then $\|M(D_t) - M(D_0)\|_1 > 1/10$, which then gives a contradiction and proves the theorem.

$$\|M(D_t) - M(D_0)\|_1 = 2 \cdot |\Pr_{x\sim D_t}[M(x) = 0] - \Pr_{x\sim D_0}[M(x) = 0]|$$

First we calculate the probability that $x \sim D_0$ (and $x \sim D_t$) satisfies Promise 1 with $\gamma = 10\beta \log \beta$. When x is drawn from the uniform distribution, a standard use of the so-called reflection principle for random walks [29][7] that

$$\Pr_{x\sim U_n}\left[\max_{i\in[n]} \sum_{j=1}^{i} x_j \geq l\right] \leq 2 \Pr_{x\sim U_n}\left[\sum_{j=1}^{n} x_j \geq l\right].$$

Therefore, the probability that $x \sim U_n$ satisfies Promise 1 with $\gamma = 10\beta\log\beta$ is at least (using a Chernoff bound)

$$1 - 2\Pr_{x\sim U_n}\left[|\sum_{j=1}^{n} x_j| \geq 10\beta\log\beta\sqrt{n}\right] \geq 1 - 4e^{-(10\beta\log\beta)^2/6} \geq 1 - 4e^{-16\beta^2} \tag{42}$$

To calculate the probability that $x \sim D_t$ satisfies Promise 1 with $\gamma = 10\beta\log\beta$, we first calculate the probability p that $x \sim \mu$ does not satisfy Promise 1 with $\gamma = 10\beta\log\beta/\sqrt{t}$. As for all $i \in [t], j \in [n/t]$,

$$\left|\sum_{g=1}^{n/t(i-1)+j} x_g\right| \leq \left(\sum_{h=1}^{i-1}\left|\sum_{g=n/t(h-1)+1}^{n/t(h)} x_g\right|\right) + \left|\sum_{g=n/t(i-1)+1}^{n/t(i-1)+j} x_g\right|,$$

the probability that $x \sim D_t$ satisfies Promise 1 with $\gamma = 10\beta\log\beta$ is at least $1-pt$. As $\mu(x) \leq 2U_m(x)$ (where $m = n/t$), p is at most 2 times the probability that $x \in U_m$ does not satisfy Promise 1 with $\gamma = 10\beta\log\beta/\sqrt{t}$, which is at most $8e^{-(10\beta\log\beta)^2/6t}$. Therefore, the probability that $x \sim D_0$ (and $x \sim D_t$) satisfies Promise 1 with $\gamma = 10\beta\log\beta$ is at least (for $\beta \geq 2$):

$$1 - 8te^{-(100\beta^2)(\log\beta)^2/6\beta\log^2\beta} \geq 99/100.$$

Then $\Pr_{x\sim D_0}[M(x) = 0] \geq \frac{2}{3}\cdot(\Pr_{x\sim U_n}[|\sum_i x_i| \leq 4\sqrt{n\alpha}] - 1/100)$. Using a Chernoff bound (for $\alpha \geq 1$),

$$\Pr_{x\sim U_n}[|\sum_i x_i| \leq 4\sqrt{n\alpha}] \geq 1 - 2e^{-\frac{(4\alpha)^2}{6}} \geq 0.85.$$

Therefore, $\Pr_{x\sim D_0}[M(x) = 0] \geq \frac{2}{3}\cdot(0.85 - 1/100) \geq \frac{1}{2}$. Next, we calculate $\Pr_{x\sim D_t}[M(x) = 0]$:

$$\Pr_{x\sim D_t}[M(x) = 0] = 1 - \Pr_{x\sim D_t}[M(x) = 1] \leq 1 - \frac{2}{3}\cdot\left(\Pr_{x\sim D_t}\left[\left|\sum_i x_i\right| \geq 4\sqrt{n\beta}\right] - 1/100\right). \tag{43}$$

Recall $D_t = \mu \times \mu \times \ldots \times \mu$. Using Lemma 3 (for $\beta \geq 2$),

$$E_{x\sim D_t}[\sum_i x_i] = \Omega\left(t\cdot\sqrt{\frac{m}{\log(1/\epsilon)}}\right) = \Omega\left(\sqrt{n}\cdot\sqrt{\frac{t}{\log(1/\epsilon)}}\right) = \Omega\left(\sqrt{n}\cdot\sqrt{\frac{\beta\log^2\beta}{\log\beta}}\right) = \Omega\left(\sqrt{n\beta}\cdot\sqrt{\log\beta}\right).$$

[7] See, e.g., the bijection argument in Claim 4.4. in `https://web.ma.utexas.edu/users/gordanz/notes/lecture4.pdf`.

Next, we compute the variance of the random variable $|X'|$ when X' is drawn from the distribution D_t. As $\forall x, \mu(x) \leq 2U_m(x)$:

$$\mathrm{Var}_{X' \sim \mu}[|X'|] \leq E_{X' \sim \mu}[|X'|^2] \leq 2 \cdot E_{X' \sim U_m}[|X'|^2] = 2m.$$

As $D_t = \mu \times \mu \times \ldots \times \mu$, $\mathrm{Var}_{X' \sim D_t}[|X'|] = t \cdot \mathrm{Var}_{X' \sim \mu}[|X'|] \leq 2mt = 2n$. Therefore, by Chebyshev's inequality (for $\beta \geq c_2$ where $c_2 > 0$ is a large enough constant for which $E_{X' \sim D_t}[|X'|] \geq 8\sqrt{n\beta}$),

$$\Pr_{x \sim D_t}\left[\left|\sum_i x_i\right| \geq 4\sqrt{n\beta}\right] \geq 1 - \frac{2n}{(4\sqrt{n\beta})^2} > \frac{91}{100}.$$

Substituting in Equation (43), we obtain $\Pr_{x \sim D_t}[M(x) = 0] < 1 - \frac{2}{3} \cdot (\frac{91}{100} - 1/100) = \frac{2}{5}$. Hence,

$$\|M(D_t) - M(D_0)\|_1 > 2\left|\frac{1}{2} - \frac{2}{5}\right| = 1/5. \quad \square$$

3.6 k-OR Promise Problem

We now consider the problem of determining the OR of multiple copies of the coin problem, and show hardness even when the input is promised to have certain nice properties.

Consider the following k-OR problem: given k sequences of n $\{-1, +1\}$ bits $X_1^j, X_2^j, \ldots, X_n^j$, $j \in \{1, 2, \ldots, k\}$, output 0 if $\forall j$, $|\sum_i X_i^j| \leq 4\sqrt{n \log k}$ and 1 if $\exists j \in [k]$, $|\sum_i X_i^j| \geq 4\sqrt{n} \log k$. We show Theorem 20 for any streaming algorithm that solves the k-OR promise problem, when $Y_i = (X_i^1, \ldots, X_i^k)$ is given as the i-th element of the stream, even if only when the input stream satisfies the following promise for a large enough constant C_{norm}.

Promise 2 (Input has small norm at all times)**.** *k sequences of n $\{-1, +1\}$ bits each, $x_1^j, x_2^j, \ldots, x_n^j, j \in [k]$, have C_{norm}-small norm at all times if for $i \in \{1, ..., n\}$,*

$$\sum_{j=1}^{k}\left(\sum_{i_1=1}^{i} x_{i_1}^j\right)^2 \leq C_{norm} \cdot k \cdot n$$

Remark 4. *Given k sequences of n bits each, $X_1^j, X_2^j, \ldots, X_n^j, j \in [k]$, such that $\forall i \in [n], j \in [k], X_i^j$ are i.i.d. uniform from $\{-1, +1\}$, these sequences satisfy Promise 2 with $C_{norm} = C$ with probability at least $1 - \frac{1}{\sqrt{k}}$, where $C > 0$ is a large enough constant.*

Proof. We will make use of a chaining inequality stated in Lemma 1. To do so, we apply Lemma 1 on the vectors $f^{(1)} = (1, 0, \ldots, 0)$, $f^{(2)} = (1, 1, 0, \ldots, 0)$, $f^{(3)} = (1, 1, 1, 0, \ldots, 0), \ldots, f^{(n)} = (1, 1, \ldots, 1)$. Such vectors define an insertion only stream, as needed to apply Lemma 1. We let Π denote the $k \times n$ matrix of the k sequences of n bits. As the k sequences are *i.i.d.* uniformly from $\{-1, +1\}^n$, Π is a $k \times n$ matrix of *i.i.d.* random variables uniform in $\{-1, +1\}$. Observe that $\|\Pi f^{(i)}\|_2^2$ is precisely $\sum_{j=1}^k \left(\sum_{i_1=1}^i x_{i_1}^j\right)^2$, when $x_1^j, x_2^j, \ldots, x_n^j, j \in [k]$ are the k sequences. By Lemma 1, there exists $\epsilon = \Theta(1/\sqrt{k})$ such that for all $i \in [n]$,

$$\left|\sum_{j=1}^{k}\left(\sum_{i_1=1}^{i} x_{i_1}^j\right)^2 - k \cdot i\right| \leq \epsilon k n.$$

Using a Markov bound, with probability $1 - \frac{1}{\sqrt{k}}$, simultaneously for all i, we have that

$$\sum_{j=1}^{k}\left(\sum_{i_1=1}^{i} x_{i_1}^j\right)^2 = O(kn).$$

The remark follows for a large enough constant $C > 0$. $\square$

Theorem 20. *There exists a large enough constant $C' > 0$ for which the following is true. Let M be a streaming algorithm which, under the promise that the k sequences of n bits, $x_1^j, x_2^j, \ldots, x_n^j$, $j \in \{1, 2, \ldots, k\}$, satisfy Promise 2 for $C_{norm} = C'$, outputs 0 with probability at least 3/4 on every 0 input of the k-OR problem, and 1 with probability at least 3/4 on every 1 input. Then there exists a constant $c > 0$[8] (for $c_3 < k < n$ where $c_3 > 0$ is a large enough constant) such that,*

$$\sum_{i=1}^{n}\sum_{j=1}^{i}(I(M_i; Y_j|M_{j-1})) > \Omega(k \cdot n \log n/(\log k \log\log k)^c).$$

Here, $\forall i \in [n], Y_i = (X_i^1, ..., X_i^k)$ is i.i.d. uniformly from $\{-1, +1\}^k$.

As observed after Corollary 14,

$$\sum_{i=1}^{n}\sum_{j=1}^{i} I(M_i; Y_j|M_{j-1}) \leq \sum_{i=1}^{n} |M_i|.$$

Therefore, M uses at least $\Omega(k \log n/(\log k \log\log k)^c)$ memory.

Proof. Using M, we construct a streaming algorithm M' that, on the stream $x'_1, x'_2, \ldots, x'_n$, outputs 0 with probability at least 2/3 on every input with $|\sum_i x'_i| \leq 4\sqrt{n \log k}$ and 1 with probability at least 2/3 on every input with $|\sum_i x'_i| \geq 4\sqrt{n \log^2 k}$ whenever $x'_1, ..., x'_n$ satisfy Promise 1 with $\gamma = 20 \log^2 k \log\log k$. For $X' \sim U_n$, we show that M' satisfies

$$\sum_{i=1}^{n}\sum_{j=1}^{i} I(M'_i; X'_j|M'_{j-1}) \leq \frac{1}{k} \cdot \sum_{i=1}^{n}\sum_{j=1}^{i} I(M_i; Y_j|M_{j-1})$$

Thus, the theorem follows from Theorem 19 by taking $\alpha = \log k$ and $\beta = \log^2 k$ (as long as $k \geq c_3$, where $c_3 > 0$ is a large enough constant for which $\log^2 k \geq c_2$). The information quantity can be written as

$$\begin{aligned}\sum_{i=1}^{n}\sum_{j=1}^{i} I(M_i; Y_j|M_{j-1}) &= \sum_{i=1}^{n}\sum_{j=1}^{i} I(M_i; X_j^1, X_j^2, \ldots, X_j^k|M_{j-1}) \\ &= \sum_{i=1}^{n}\sum_{j=1}^{i}\sum_{l=1}^{k} I(M_i; X_j^l|M_{j-1}, X_j^{<l}) \\ &= \sum_{l=1}^{k}\sum_{i=1}^{n}\sum_{j=1}^{i} I(M_i; X_j^l|M_{j-1}, X_j^{<l})\end{aligned}$$

Therefore, there exists $l \in [k]$ such that

$$\sum_{i=1}^{n}\sum_{j=1}^{i} I(M_i; X_j^l|M_{j-1}, X_j^{<l}) \leq \frac{1}{k} \cdot \sum_{i=1}^{n}\sum_{j=1}^{i} I(M_i; Y_j|M_{j-1}) \tag{44}$$

Given a stream X' of n $\{-1, +1\}$ bits $X'_1, \ldots, X'_n$, M', using private randomness, creates $k-1$ sequences of n *i.i.d.* uniform on $\{-1, +1\}$ random bits (named $X^1, \ldots, X^{l-1}, X^{l+1}, \ldots, X^k$). M' runs M on the k sequences with X' embedded as the l^{th} sequence and outputs as M outputs. M' at the i^{th} step, generates $X_i^1, \ldots, X_i^{l-1}, X_i^{l+1}, \ldots, X_i^k$, simulates the i^{th} step of M with $X_i^l = X'_i$, and only remembers M_i.

We only need to calculate the success probability of M' on input $x'_1, ..., x'_n$ satisfying Promise 1 with $\gamma = 20 \log^2 k \log\log k$. Also, M' can trust M's output only when it generates $k-1$ sequences such that the

[8] $c = 22$.

k sequences altogether satisfy the Promise 2 for $C_{norm} = C'$ (where C' is a large enough constant). As each of the $k-1$ sequences are drawn from U_n, using Remark 4, the $k-1$ sequences satisfy the Promise 2 for $C_{norm} = C$ with probability at least $1 - \frac{1}{\sqrt{k-1}}$. As the l-th sequence, $x'_1, ..., x'_n$, satisfies Promise 1 with $\gamma = 20 \log^2 k \log\log k$, the norm of the k sequences (at any time) is changed by at most $(20\log^2 k \log\log k)^2 n \le 400kn$ ($k > c_3$ where $c_3 > 0$ is a large enough constant). Therefore, for $C' = C + 400$, the k sequences altogether satisfy the Promise 2 for $C_{norm} = C'$, with probability at least $1 - \frac{1}{\sqrt{k-1}}$.

Rephrasing, M' runs M on an input satisfying Promise 2 for $C_{norm} = C'$, with probability at least $1 - \frac{1}{\sqrt{k-1}}$. Next, we calculate the probability that M' outputs 0 when $|\sum_i x'_i| \le 4\sqrt{n \log k}$ and $x'_1, ..., x'_n$ satisfies Promise 1 with $\gamma = 20 \log^2 k \log\log k$. It is at least equal to the probability that M' generates a 0 input for the k-OR problem for M (that satisfies the promise) multiplied by the probability that M outputs 0 on this 0 input. By a Chernoff bound for n *i.i.d.* Bernoulli(1/2) random variables, we have that $\forall j \in [k] - \{l\}$,

$$\Pr\left[\left|\sum_i X_i^j\right| > 4\sqrt{n\log k}\right] \le 2 \cdot e^{-\frac{16\log k}{n}\cdot\frac{n}{6}} \le \frac{2}{k^3}. \tag{45}$$

Therefore, by a union bound, the probability that M' generates a 0 input (for M) satisfying the promise for the k-OR problem is at least

$$1 - 2\cdot\frac{k-1}{k^3} - \frac{1}{\sqrt{k-1}} \ge 17/18,$$

where $k \ge c_3$ for $c_3 > 0$ a large enough constant. Thus, the probability that M' outputs 0 when $|\sum_i x'_i| \le 4\sqrt{n\log k}$, under the promise 1, is at least $17/18 \cdot 3/4 \ge 2/3$.

Next, we calculate the probability that M' outputs 1 when $|\sum_i x'_i| \ge 4\sqrt{n}\log k$ and $x'_1, ..., x'_n$ satisfies Promise 1 with $\gamma = 20\log^2 k \log\log k$. As M' would always generate a 1 input for the k-OR problem for M and an input that satisfies the promise with probability $1 - 1/\sqrt{k-1}$, the probability that M' outputs 1 is at least

$$\frac{3}{4}\left(1 - \frac{1}{\sqrt{k-1}}\right) \ge 2/3.$$

Next, we prove an information bound for the streaming algorithm M'.

$$\begin{aligned}
&\sum_{i=1}^{n}\sum_{j=1}^{i} I(M'_i; X'_j | M'_{j-1}) \\
&\quad= \sum_{i=1}^{n}\sum_{j=1}^{i} I(M_i; X_j^l | M_{j-1}) \\
&\quad= \sum_{i=1}^{n}\sum_{j=1}^{i} I(X_j^{<l}; X_j^l | M_{j-1}) + I(M_i; X_j^l | M_{j-1}, X_j^{<l}) - I(X_j^{<l}; X_j^l | M_{j-1}, M_i) \\
&\quad\le \sum_{i=1}^{n}\sum_{j=1}^{i} I(M_i; X_j^l | M_{j-1}, X_j^{<l}) \\
&\quad\le \frac{1}{k}\cdot\sum_{i=1}^{n}\sum_{j=1}^{i} I(M_i; Y_j | M_{j-1})
\end{aligned}$$

The first inequality follows from the fact that $I(X_j^{<l}; X_j^l | M_{j-1}) = 0$. The last inequality follows from Equation (44). □

The k-OR problem can be easily solved by storing the counts of all the sequences in $O(k\log n)$ memory. In Theorem 20, we prove an $\Omega(k\log n/poly(\log k))$ memory lower bound (tight up to $poly(\log k)$ factors) for any streaming algorithm that solves the k-OR Problem for gap (β/α) as large as $\log k$. In the following theorem, using a simple reduction from the Simultaneous k-Coins Problem, we in fact prove a tight $\Omega(k \log n)$ memory lower bound for any streaming algorithm that solves the k-OR Problem but for gaps (β/α) of at most $\left(1 + \frac{1}{4C_{gap}\sqrt{\log k}}\right)^2$ (where $C_{gap} > 0$ is a large enough fixed constant).

Consider the following k-OR small gap problem: given k sequences of n $\{-1,+1\}$ bits $X_1^j, X_2^j, \ldots, X_n^j$, $j \in \{1,2,\ldots,k\}$, output 0 if $\forall j,\ |\sum_i X_i^j| \le 4\sqrt{n \log k}$ and 1 if $\exists j \in [k],\ |\sum_i X_i^j| \ge 4\sqrt{n \log k} + \frac{\sqrt{n}}{C_{gap}}$. For a large enough fixed constant $C_{gap} > 0$, we prove Theorem 21 for any streaming algorithm which solves the k-OR small gap problem, even if only when the input stream satisfies Promise 2 for a large enough constant C_{norm}.

Theorem 21. *There exists a large enough constant $C' > 0$ for which the following holds. Let M be a streaming algorithm which, under the promise that the k input sequences of n bits, $x_1^j, x_2^j, \ldots, x_n^j$, $j \in \{1,2,\ldots,k\}$, satisfy Promise 2 for $C_{norm} = C'$, outputs 0 with probability at least $1 - 10^{-4}$ on every 0 input of the k-OR small gap problem, and 1 with probability at least $1 - 10^{-4}$ on every 1 input, then (for $c_4 < k \le n$ where $c_4 > 0$ is a large enough constant) M requires $\Omega(k \log n)$ memory.*

Proof. Using M, we construct a streaming algorithm M' that solves the Simultaneous k-Coins Problem on $n' \ge n/2$ bits, with probability at least $1 - \frac{1}{2000}$, and uses $O(k + \log n)$ memory in addition to the memory used by M. Hence, Corollary 17 implies the theorem. Let $n' = n - 4\sqrt{n \log k}$ ($n' \ge n/2$ for large enough n). M' is given a kn' bit input $X'_1, ..., X'_{kn'} \in \{0,1\}^{kn'}$ in a stream, such that X'_i belongs to the $((i-1) \mod k+1)^{th}$ instance of the k-Coins Problem. M' constructs the input stream for M as follows. For all $i \in [n']$, $Y_i = (2X'_{(i-1)k+1} - 1, 2X'_{(i-1)k+2} - 1, ..., 2X'_{i \cdot k} - 1)$, that is, $\forall j \in [k], i \in [n'], X_i^j = 2X'_{(i-1)k+j} - 1$ (this step requires an additional k memory bits to remember the input bits for k layers). After kn' steps, M' is given $\ell \in [k]$ and it needs to output the majority of the ℓ-th instance. For the next $n - n'$ steps, M' adds $Y_i = (1,1,...,1)$ when $(i \mod 2) = 0$, and $Y_i = (-1,-1,...1,...,-1)$ when $(i \mod 2) = 1$ to the input stream for M (this step requires an additional $O(\log n)$ memory bits). That is, M' adds $4\sqrt{n \log k}$ 1s to the ℓ-th sequence, and 1s and -1s alternatively to the rest of the sequences (the total is 0).

We first calculate the probability that M' generates an input satisfying Promise 2 for $C_{norm} = C'$ (for a large enough constant $C' > 0$). The k sequences that are input to M in the first n' layers are *i.i.d.* uniformly from $\{-1,+1\}^{n'}$ and hence, using Remark 4, they satisfy Promise 2 for $C_{norm} = C$ with probability at least $1 - \frac{1}{\sqrt{k}}$. Note that, for $i > n'$,

$$\sum_{j=1}^{k}\left(\sum_{i_1=1}^{i} x_{i_1}^j\right)^2 = \sum_{j=1}^{k}\left(\sum_{i_1=1}^{n'} x_{i_1}^j + \sum_{i_1=n'+1}^{i} x_{i_1}^j\right)^2$$

$$\le 2 \cdot \sum_{j=1}^{k}\left(\left(\sum_{i_1=1}^{n'} x_{i_1}^j\right)^2 + \left(\sum_{i_1=n'+1}^{i} x_{i_1}^j\right)^2\right).$$

And as $\forall j \neq \ell$, $|\sum_{i_1=n'+1}^{i} x_{i_1}^j| \le 1$ and for $j = \ell$, $|\sum_{i_1=n'+1}^{i} x_{i_1}^j| \le 4\sqrt{n \log k}$, for all $i > n'$,

$$\sum_{j=1}^{k}\left(\sum_{i_1=1}^{i} x_{i_1}^j\right)^2 \le 2C \cdot k \cdot n + 2k + 16 n \log k = O(kn).$$

Therefore, the input to M satisfies Promise 2 for $C_{norm} = C'$ (for a large enough constant $C' > 0$), with probability at least $1 - \frac{1}{\sqrt{k-1}}$.

M' outputs whatever M outputs. Next, we calculate the probability that M' outputs the correct answer to the majority of the ℓ-th instance. M' runs M on a 0 input if for all $j \neq \ell$,

$$|\sum_{i=1}^{n'} x_i^j| \le 4\sqrt{n \log k}, \tag{46}$$

and $8\sqrt{n \log k} \leq \sum_{i=1}^{n'} x_i^\ell \leq 0$ (majority of ℓ-th instance is 0). M' runs M on a 1 input if $\frac{\sqrt{n}}{C_{gap}} \leq \sum_{i=1}^{n'} x_i^\ell$ (majority of ℓ-th instance is 1).

Using a Chernoff bound as in Equation (45) and a union bound, the probability that $\{X_i^j\}_{j\in[k],i\in[n']}$ satisfies Inequality 46 is at least $1 - \frac{2(k-1)}{k^3}$. Next, using a Chernoff bound,

$$\Pr\left[8\sqrt{n \log k} \leq \sum_{i=1}^{n'} X_i^\ell \leq 0\right] \geq \frac{1}{2} - e^{-64 \log k/6} \geq \frac{1}{2} - k^{-8}. \tag{47}$$

There exists a large enough constant $C_{gap} > 0$ such that

$$\Pr\left[\frac{\sqrt{n}}{C_{gap}} \leq \sum_{i=1}^{n'} X_i^\ell\right] \geq \frac{1}{2} - 10^{-4}. \tag{48}$$

This is because for all $s \in [n]$,

$$\Pr\left[\sum_{i=1}^{n'} X_i^\ell = s\right] \leq \Pr\left[\sum_{i=1}^{n'} X_i^\ell = 0\right] = O(1/\sqrt{n'}) = O(1/\sqrt{n}).$$

M outputs the correct answer with probability at least $1 - 10^{-4}$ (over the private randomness) whenever the input satisfies Promise 2 for $C_{norm} = C'$ and is a valid 0 or a 1 input. Whenever Equation (46), and Equation (47) or Equation (48) are satisfied and the input to M satisfies Promise 2 for $C_{norm} = C'$, the expected outcome from M agrees with the correct answer of M'. Therefore, M' outputs correctly with probability at least

$$(1 - 10^{-4}) \cdot \left(1 - \frac{1}{\sqrt{k-1}} - \frac{2(k-1)}{k^3} - k^{-8} - 10^{-4}\right),$$

which is greater than $1 - \frac{1}{2000}$ for $k > c_4$ where $c_4 > 0$ is a large enough constant. □

4 Data Stream Applications

We apply our lower bounds for the coin problem to a number of data stream problems. Our communication lower bounds give new bounds in a number of different data stream models.

4.1 Lower Bounds with Bounded Deletions

Recall that in the bounded deletions model, at all times in the stream, the norm $\|x\|_2$ is never a constant factor less than its value at any earlier point in the stream. We will not explicitly state this constant factor, but will prove lower bounds for various problems in bounded deletion streams for some constant factor. Also, we will assume the streaming algorithm succeeds with sufficiently large constant probability for the lower bounds to hold, where in this subsection the probability is taken over its internal randomness. For all algorithms in this section, this can be achieved by repeating the algorithm a constant number of times and taking the median estimate (in the case of ℓ_2-heavy hitters, we take the median estimate of each item). Throughout we assume, unless stated otherwise, that the parameter ϵ for approximation is such that $c'' < \epsilon^{-2} \leq \min\{m,d\}^{0.9}$ (or $c'' < k \leq \min\{m,d\}^{0.9}$) where m is the length of the stream, d is the dimension of the vector x and $c'' > 0$ is a large enough constant. For an integer v, we also use the notation $x_i \leftarrow x_i + v$ to represent $|v|$ consecutive $\frac{v}{|v|}$ updates to x_i.

ℓ_∞-Estimation.

Theorem 22. *($\|x\|_\infty$-Approximation) Any streaming algorithm M in the bounded deletions model which outputs a number Z satisfying*

$$\|x\|_\infty - \frac{\|x\|_2}{\sqrt{k}} \leq Z \leq \gamma\|x\|_\infty + \frac{\|x\|_2}{\sqrt{k}}$$

with sufficiently large constant probability over its private randomness, requires $\Omega(k \log m)$ bits of memory if $\gamma = 1$, and $\Omega\left(\frac{k \log m}{polylog(k)}\right)$ bits of memory[9] *for any $1 < \gamma \leq c'\sqrt{\log k}$ (where $c' > 0$ is a small enough constant) and $c'' < k \leq \min\{m^{0.5}, d\}$ (where $c'' > 0$ is a large enough constant).*

Proof. We prove the lower bound via a reduction from the k-OR problem for $k' = ck$ sequences of n bits, where $c > 0$ is a sufficiently small constant.

Given an input stream to the k-OR problem on k' sequences, we construct a streaming algorithm M' that creates a stream of updates to x, the underlying d-dimensional vector, and outputs the correct answer using the approximation that M returns. Thus, if the k-OR problem is hard for low-memory algorithms, the approximation of $\|x\|_\infty$ is also hard. Each of the k' coins of the k-OR problem corresponds to one of the k' coordinates of x, and the updates to that coin correspond to the corresponding stream updates to that coordinate of x. Before making the updates corresponding to the k-OR problem, we add $\sqrt{n}$ 1s each to an additional k' coordinates (assuming $2k' \leq k \leq d$). That is, $x_i \leftarrow x_i + \sqrt{n}$ for $k' + 1 \leq i \leq 2k'$.

Under the promise of the k-OR problem (Promise 2) and due to the additional k' coordinates, we have that the resulting stream satisfies the bounded deletions property, and that $\|x\|_2 = \Theta(\sqrt{nk'})$ at all times during the stream.

If $\gamma = 1$ we reduce from the k-OR small gap problem, using Theorem 21. In this case, in the underlying GapCoin(α, β) problem, we have $\alpha = \log k'$ and $(\beta/\alpha) = \left(1 + \frac{1}{4C_{gap}\sqrt{\log k'}}\right)^2$, where C_{gap} is the constant defined in the k-OR small gap problem in Theorem 21. Otherwise, $\gamma > 1$ and we choose $\alpha = \log k'$ and $\beta = \log^2 k'$. For such a choice of β, we claim that the following inequality holds for all x generated under the promise for the k-OR problem:

$$4\sqrt{n\beta} > 4\gamma\sqrt{n\alpha} + \frac{2\|x\|_2}{\sqrt{k}}. \tag{49}$$

Notice that since $\gamma > 1$ and $\|x\|_2 = O(\sqrt{nk'})$ under the promise ($k' = ck$ for small enough constant $c > 0$), there exists $\beta = O(\gamma^2 \log k)$ that satisfies this inequality. Since we also assume that $\gamma \leq c'\sqrt{\log k}$, in this case we have that $O(\gamma^2 \log k) \leq O(c' \log^2 k) \leq \log^2 k'$ (for small enough constant $c' > 0$ and large enough k), and thus $\beta = \log^2 k'$ suffices. Therefore, we apply Theorem 20 for this choice of $\alpha = \log k'$ and $\beta = \log^2 k'$.

Although we have chosen α and β to satisfy (49) when $\gamma > 1$, we claim that our choice of α and β when $\gamma = 1$ also satisfies (49). Observe that

$$4\sqrt{n\beta} = 4\sqrt{n\alpha}\left(1 + \frac{1}{4C_{gap}\sqrt{\log k'}}\right) = 4\sqrt{n\alpha} + \frac{\sqrt{n}}{C_{gap}}.$$

Consequently, $4\sqrt{n\beta} - 4\sqrt{n\alpha} = \frac{\sqrt{n}}{C_{gap}} > \frac{2\|x\|_2}{\sqrt{k}}$ for $k' = ck$ for a sufficiently small constant $c > 0$. Indeed, to see this inequality, since $\|x\|_2 = O(\sqrt{nk'})$ we have $\frac{2\|x\|_2}{\sqrt{k}} = O(\sqrt{cn})$, and for $c > 0$ sufficiently small, this is at most $\frac{\sqrt{n}}{C_{gap}}$, and so (49) holds also for $\gamma = 1$.

If the GapCoin(α, β) problem evaluates to 1, then $\|x\|_\infty \geq 4\sqrt{n\beta}$. In this case if the streaming algorithm succeeds, then

$$Z \quad \geq \quad \|x\|_\infty - \frac{\|x\|_2}{\sqrt{k}}$$

[9] $\Omega\left(\frac{k \log m}{(\log k \log\log k)^{22}}\right)$ to be precise.

$$\begin{aligned} &\geq 4\sqrt{n\beta} - \frac{\|x\|_2}{\sqrt{k}} \\ &> 4\gamma\sqrt{n\alpha} + \frac{\|x\|_2}{\sqrt{k}}, \end{aligned}$$

where we have used (49). On the other hand, if the GapCoin(α, β) problem evaluates to 0, then $\|x\|_\infty \leq 4\sqrt{n\alpha}$, and so

$$\begin{aligned} Z &\leq \gamma\|x\|_\infty + \frac{\|x\|_2}{\sqrt{k}} \\ &\leq 4\gamma\sqrt{n\alpha} + \frac{\|x\|_2}{\sqrt{k}} \end{aligned}$$

Thus, from the output of the streaming algorithm, we can solve the GapCoin(α, β) problem. Noting that the stream length $m = nk$ and that we assume $m/k > m^{0.5}$ $(n > k)$, we have $\log n = \Theta(\log m)$. The conclusion now follows by applying Theorem 21 when $\gamma = 1$ and Theorem 20 when $\gamma > 1$. □

Our next three applications all make use of the following claim, which is that a certain stream generated using the input to the SIMULTANEOUS k-COINS PROBLEM satisfies the bounded deletions property.

Consider an instance of the SIMULTANEOUS k-COINS PROBLEM. Let x be an underlying d-dimensional vector in a stream, initialized to 0^d, where we assume $d \geq (k+1)$. We identify an input, which is 1 or 0, in the i-th copy of the SIMULTANEOUS k-COINS PROBLEM, with an update to the i-th coordinate of x, where $x_i \leftarrow x_i + 1$ if the update is a $+1$, or $x_i \leftarrow x_i - 1$ if the update is a 0. We start the stream by performing the update $x_{k+1} \leftarrow x_{k+1} + \sqrt{kn}$. We then update x as defined by the above update rule given the inputs to the SIMULTANEOUS k-COINS PROBLEM. At the end of the stream, for some index $j \in \{1, 2, \ldots, k\}$, we add a single number of any value to coordinate x_j.

We call this a *simultaneous stream.*

Claim 8. *(Bounded Deletion Property for* SIMULTANEOUS k-COINS PROBLEM*) For any constant $\delta > 0$, for k greater than a sufficiently large constant (depending on δ), the probability that a simultaneous stream satisfies the bounded deletions property is at least* $1 - \delta$.

Proof. We will make use of the chaining inequality stated in Lemma 1. Let α be a sufficiently small constant to be determined. We show that at any times t' and t in the stream, with $t' > t$, we have $\|x^{(t')}\|_2^2 \geq \alpha\|x^{(t)}\|_2^2$, where $x^{(t)}$ is the value of the vector x after receiving t stream updates. To do so, we apply Lemma 1 on the vectors $f^{(1)} = (1, 0, \ldots, 0)$, $f^{(2)} = (1, 1, 0, \ldots, 0)$, $f^{(3)} = (1, 1, 1, 0, \ldots, 0), \ldots, f^{(n)} = (1, 1, \ldots, 1)$. Such vectors define an insertion only stream, as needed to apply Lemma 1. We let Π denote the $k \times n$ matrix of inputs to the SIMULTANEOUS k-COINS PROBLEM. Observe that $\|\Pi f^{(i)}\|_2^2$ is precisely $\|x^{(ik)}\|_2^2 - x_{k+1}^2$, where $x^{(ik)}$ denotes the underlying vector x after having seen exactly $i \cdot k$ updates, for an integer i. By Lemma 1 and a Markov bound (and using that $\epsilon k = \Theta(\sqrt{k})$), with probability $1 - O(1/\sqrt{k})$, simultaneously for all i, we have that $\|x^{(ik)}\|_2^2 - x_{k+1}^2 = O(kn)$. Let us call this event $\mathcal{G}$ and condition on it.

Next, notice that

$$|\|x^{(ik)}\|_2^2 - \|x^{(ik+\ell)}\|_2^2| \leq 2\|x^{(ik)}\|_2\sqrt{k} + k = O(k\sqrt{n} + k),$$

for any integer $\ell \in \{0, 1, 2, \ldots, k-1\}$. It follows that $\|x^t\|_2^2 - x_{k+1}^2 = O(kn)$ at all times t in the stream. Notice though that $x_{k+1}^2 = kn$ at all times after the initial insertion at the beginning of the stream, and consequently, with probability $1 - O(1/\sqrt{k})$ (greater than $1 - \delta/2$), our stream satisfies the bounded deletion property with a small enough constant α.

Finally, at the end of the stream, we add a number of any value v to a coordinate x_j. Observe that if v has the same sign as x_j, this only increases $\|x\|_2$, and so the bounded deletions property still holds. On the other hand, if v has the opposite sign as x_j, then this is equivalent to first reducing the current x_j to 0, and then replacing the value from 0 to $v + x_j$. By choosing a large enough constant $C > 0$, we have that, right before adding the value v, that $|x_j| \leq C\sqrt{n}$ with arbitrarily large constant probability (greater than $1 - \delta/2$).

Thus, by the triangle inequality, reducing x_j to 0 decreases the norm of x by at most $C\sqrt{n}$. However, since we have $\|x\|_2 \geq \sqrt{kn}$ right before adding the value v, it follows that we still have $\|x\|_2 = \Theta(\sqrt{kn})$ (for sufficiently large k depending on C and hence δ) after reducing x_j to 0, so the bounded deletions property still holds. Finally, adding $v + x_j$ to coordinate x_j only increases $\|x\|_2$, so the bounded deletions property holds at the end of the stream as well. □

We now use Claim 8 to prove lower bounds in the bounded deletions model for other problems.

ℓ_2-Estimation. In [21], it was asked if one could obtain an upper bound better than $O((\log m)/\epsilon^2)$ memory for ℓ_2-estimation in the bounded deletions model, namely, if the ℓ_2-norm of the input vector never drops by more than an α-fraction of what it is at any earlier point in the stream. The only known lower bound is $\Omega((\log(1/\alpha))/\epsilon^2)$. Here we show a lower bound of $\Omega((\log m)/\epsilon^2)$ even for constant α, thus resolving the question in [21] in the negative.

Theorem 23. *Any streaming algorithm in the bounded deletions model which, with sufficiently large constant probability, outputs a number $Z = (1 \pm \epsilon)\|x\|_2^2$ at the end of the stream, for every set of stream updates and ϵ smaller than a small enough constant, requires $\Omega((\log m)/\epsilon^2)$ bits of memory.*

Proof. We reduce from the Simultaneous k-Coins problem for a value $k = \Theta(\frac{1}{\epsilon^2\delta})$ ($k + 1 \leq d$), where $\delta > 0$ is a sufficiently small constant. We create a simultaneous stream as defined above. In our case, the last player has the index j in the Simultaneous k-Coins problem and adds the value $C\sqrt{kn}$ to x_j at the end of the stream, where $C > 1$ will be an arbitrarily large constant. By Claim 8, for large enough k, with arbitrarily large constant probability $(1 - \delta)$, the simultaneous stream satisfies the bounded deletions property.

Suppose the output of the streaming algorithm is in $(1 \pm c'\epsilon)\|x\|_2^2$ with probability at least $1 - \delta$, where $c' > 0$ is an arbitrarily small constant. We will show an $\Omega(\epsilon^{-2} \log m)$ memory lower bound for streaming algorithms achieving such an approximation. It will then follow that by rescaling ϵ by ϵ/c', we will have an $\Omega(\epsilon^{-2} \log m)$ lower bound for algorithms outputting a number in $(1 \pm \epsilon)\|x\|_2^2$. Let $\mathcal{E}$ be the event that the streaming algorithm succeeds in outputting a number in $(1 \pm c'\epsilon)\|x\|_2^2$.

Right before adding $C\sqrt{kn}$ to x_j, let X_j be the current value of the j-th coordinate of x. By the anti-concentration of the binomial distribution (which follows from Stirling's approximation given in Section 2) and assuming k is greater than a sufficiently large constant, it follows that there is a sufficiently small constant $\gamma > 0$ (depending on δ) for which

$$\Pr[\sqrt{kn} \geq |X_j| > \gamma\sqrt{n}] \geq 1 - \delta. \tag{50}$$

Here $\delta > 0$ is as above. Note that (50) holds for any choice of $C > 1$, provided k is sufficiently large. Let $\mathcal{F}$ denote the event that $\sqrt{kn} \geq |X_j| > \gamma\sqrt{n}$.

If $\mathcal{F}$ occurs, then if $X_j \geq 0$, we have:

$$(C\sqrt{kn} + X_j)^2 \geq (C\sqrt{k} + \gamma)^2 n. \tag{51}$$

On the other hand, if $X_j < 0$, we have:

$$(C\sqrt{kn} + X_j)^2 \leq (C\sqrt{k} - \gamma)^2 n. \tag{52}$$

Note that $C > 1$ and so given event $\mathcal{F}$, we have that $C\sqrt{kn} + X_j$ is positive.

Let x_{-j} be the vector at the end of the stream, excluding the j-th coordinate. Observe that $\|x_{-j}\|_2^2 = \|S \cdot 1^n\|_2^2$, where $S \in \{-1, 1\}^{(k-1)\times n}$ is uniformly random, and 1^n is a vector of all 1s. It follows by standard sketching guarantees for estimating the Euclidean norm of a vector in a stream (see, e.g., the proof of Theorem 2.2 of [2], where the first and second moments of $\|S \cdot 1^n\|_2^2$ are calculated), and our choice of $k = \left(\Theta(\frac{1}{\epsilon^2\delta})\right)$, that

$$\Pr[\|x_{-j}\|_2^2 = (1 \pm \epsilon)n(k-1)] \geq 1 - \delta.$$

Let us call this event $\mathcal{G}$.

By a union bound, $\Pr[\mathcal{E} \wedge \mathcal{F} \wedge \mathcal{G}] \geq 1 - 3\delta$. We condition on $\mathcal{E} \wedge \mathcal{F} \wedge \mathcal{G}$ occurring in what follows. A deterministic case analysis then finishes the proof:

Case: $X_j \geq 0$: In this case,

$$\begin{aligned} \|x\|_2^2 &\geq nk - n - \epsilon nk + (C\sqrt{k} + \gamma)^2 n \\ &\geq nk - \epsilon nk + C^2 kn + 2C\sqrt{k}\gamma n + \gamma^2 n \\ &\geq (C^2 nk + nk) + (2C\sqrt{k}\gamma n - \epsilon nk - n) + \gamma^2 n. \end{aligned}$$

Case: $X_j < 0$: In this case,

$$\begin{aligned} \|x\|_2^2 &\leq nk + \epsilon nk + (C\sqrt{k} - \gamma)^2 n \\ &\leq nk + \epsilon nk + C^2 kn - 2C\sqrt{k}\gamma n + \gamma^2 n \\ &\leq (C^2 nk + nk) - (2C\sqrt{k}\gamma n - \epsilon nk) + \gamma^2 n. \end{aligned}$$

We can make C an arbitrarily large constant, depending on an already fixed γ (this will only affect our choice of c' below, and ultimately degrade our lower bound by a constant factor). Note also $\epsilon nk = \Theta(\sqrt{k}n)$.

By making C an arbitrarily large constant, we have that if $X_j \geq 0$, then $\|x\|_2^2 \geq (C^2 nk + nk) + C\sqrt{k}\gamma n$. On the other hand, if $X_j < 0$, then $\|x\|_2^2 \leq (C^2 nk + nk) - C\sqrt{k}\gamma n$. Using that $C\sqrt{k}\gamma n = \Theta(\epsilon) \cdot (C^2 nk + nk)$, it follows that a data stream algorithm which provides a $(1 \pm c'\epsilon)$-approximation to $\|x\|_2^2$ for a sufficiently small constant $c' > 0$, can distinguish the two cases. Consequently, it can solve the Simultaneous k-Coins problem with probability $1 - 4\delta$, and requires $\Omega(\epsilon^{-2} \log n)$ bits of memory (Corollary 17). Noting that $m = nk$ here ($k \leq m^{0.9}$), we see that such an algorithm requires $\Omega(\epsilon^{-2} \log m)$ bits of memory. Rescaling ϵ by c' completes the proof. □

Point Query. By using the distributional version of the Simultaneous k-Coins problem, we are able to show an $\Omega(\epsilon^{-2} \log m)$ bit lower bound in the bounded deletions model. This is also optimal given an $O(\epsilon^{-2} \log m)$ bit upper bound using the CountSketch data structure [11] (here we only need a constant number of rows in the data structure of [11], since we only need to be correct on a fixed index j).

Theorem 24. *(Point Query Problem) Any streaming algorithm which, in the bounded deletions model, with sufficiently large constant probability, solves the ℓ_2-Point Query Problem, requires $\Omega(\epsilon^{-2} \log m)$ bits of memory for small enough ϵ.*

Proof. We reduce from the Simultaneous k-Coins problem for a value $k = \Theta(\epsilon^{-2})$, creating a simultaneous stream as defined above. In our case, the last player has the index j in Simultaneous k-Coins and adds nothing to x_j at the end of the stream. By Claim 8, for large enough ϵ^{-2} (such that $k = c'\epsilon^{-2}$, for a sufficiently small constant $c' > 0$ to be determined later, still has k being large), with arbitrarily large constant probability, the simultaneous stream satisfies the bounded deletions property.

We have $\mathbf{E}[F_2] = kn$, and thus by a Markov bound, with probability at least $1 - c^2$, $\|x\|_2 \leq (1/c)\sqrt{kn}$, where $c > 0$ is an arbitrarily small constant. Let us call this event $\mathcal{E}$. Under the event $\mathcal{E}$, for any given ϵ, one can choose a sufficiently small constant value of $c' > 0$, where $k = c'\epsilon^{-2}$, for which $\epsilon\|x\|_2 < \delta\sqrt{n}$, for a sufficiently small constant $\delta > 0$

The last player holds an index $j \in \{1, 2, \ldots, k\}$, and computes the output E of the streaming algorithm on query j, which produces an estimate $\hat{x}_j$ with $|\hat{x}_j - x_j| \leq \epsilon\|x\|_2$ with arbitrarily large constant probability. Let us call this latter event $\mathcal{F}$.

Given the occurrence of events $\mathcal{E}$ and $\mathcal{F}$, as well as the bounded deletions property, it follows that we learn the sum on the j-th coin up to additive error at most $\delta\sqrt{n}$ for an arbitrarily small constant $\delta > 0$ (and thus obtain a constant success probability arbitrarily close to 1 for outputting the majority on the j-th coin). Consequently, we can solve the Simultaneous k-Coins problem with arbitrarily large constant probability. Hence, the streaming algorithm must use $\Omega(k \cdot \log n)$ bits of memory (Corollary 17), which is $\Omega(k \cdot \log m)$ bits of memory since $m = k \cdot n$ and we assume $m/k > m^{0.1}$. □

ℓ_2-Heavy Hitters. We next show an $\Omega(\epsilon^{-2}\log m)$ lower bound in the bounded deletions model for the ℓ_2-Heavy Hitters problem, which together with the trivial $\Omega(\epsilon^{-2}\log d)$ lower bound, implies the algorithms of [6, 7] are optimal up to a $\log(1/\epsilon)$ factor, given the observation of [21] that these algorithms apply in the bounded deletions model (where as usual, the parameter α in the bounded deletions model is a constant).

Theorem 25 (ℓ_2-Heavy Hitters Problem)**.** *Any streaming algorithm which, in the bounded deletions model, solves the ℓ_2-Heavy Hitters Problem with sufficiently large constant probability, requires $\Omega(\epsilon^{-2}\log m)$ bits of memory.*

Proof. We reduce from the Simultaneous k-Coins problem for a value $k = \Theta(\epsilon^{-2})$, creating a simultaneous stream as defined above. In our case, again the last player has the index j in Simultaneous k-Coins problem and adds nothing to x_j at the end of the stream. By Claim 8, for large enough ϵ^{-2} (such that $k = c'\epsilon^{-2}$, for sufficiently small constant $c' > 0$ to be determined later, is still large), with arbitrarily large constant probability, the simultaneous stream satisfies the bounded deletions property.

We have that $\mathbf{E}[F_2] = kn$, and thus by a Markov bound, with probability at least $1-c^2$, $\|x\|_2 \leq (1/c)\sqrt{kn}$, where $c > 0$ is an arbitrarily small constant. Let us call this event $\mathcal{E}$. Under the event $\mathcal{E}$, one can choose a sufficiently small constant value of $c' > 0$, where $k = c'\epsilon^{-2}$, for which $\epsilon\|x\|_2 < \delta\sqrt{n}$, for a sufficiently small constant $\delta > 0$.

The last player holds an index $j \in \{1, 2, \ldots, k\}$, and computes the output of the streaming algorithm. By the guarantee of the streaming algorithm, we have that from the output we can produce an estimate $\hat{x}_j$ with $|\hat{x}_j - x_j| \leq \delta\sqrt{n}$ with arbitrarily large constant probability. Indeed, by the ℓ_2-Heavy Hitters guarantee, we have that either j is in the output set of the heavy hitters algorithm, in which case its estimate $\hat{x}_j$ satisfies this guarantee, or it is not in the output set, in which case the estimate $\hat{x}_j = 0$ satisfies this guarantee. Let us call the event that $\hat{x}_j$ satisfies this guarantee the event $\mathcal{F}$.

Given the occurrence of events $\mathcal{E}$ and $\mathcal{F}$, as well as the bounded deletions property, it follows that we learn the sum on the j-th coin up to additive error at most $\delta\sqrt{n}$ for an arbitrarily small constant $\delta > 0$. Consequently, we can solve the Simultaneous k-Coins problem problem with arbitrarily large constant probability. Hence, the streaming algorithm must use $\Omega(k \cdot \log n)$ bits of memory (Corollary 17), which is $\Omega(k \cdot \log m)$ bits of memory since $m = k \cdot n$ and we assume $m/k > m^{0.1}$. □

4.2 Lower Bounds in the Random Order Model

Point Query. We next show the first optimal $\Omega(\epsilon^{-2}\log m)$ lower bound for the ℓ_2-Point Query Problem if the stream (of insertions and deletions) occurs in a random order.

Theorem 26. *Any streaming algorithm in the random order model which succeeds with sufficiently large constant probability, over the random order and private randomness, in solving the ℓ_2-Point Query Problem requires $\Omega(\epsilon^{-2}\log m)$ bits of memory.*

Proof. We reduce from the k-Coins Problem for the random order for a value $k = \Theta(\epsilon^{-2})$ (Corollary 18). Let x be the underlying d-dimensional vector in the stream, initialized to 0^d, where we assume $d \geq k$. We identify an input, which is 1 or 0, in the i-th copy of the k-Coins Problem with an update to the i-th coordinate of x, where $x_i \leftarrow x_i + 1$ if the update is a +1, or $x_i \leftarrow x_i - 1$ if the update is a 0. By definition of the k-Coins Problem for random order, the updates in the stream, even when the set of updates are fixed, appear in a uniformly random order. Using Equation (31), we observe with probability at least $1 - k \cdot e^{-n/8}$, the number of updates to each coordinate is at least $n/2$. Let us call this event $\mathcal{G}$.

The last player for the k-Coins Problem holds an index $j \in \{1, 2, \ldots, k\}$, and computes the output E of the streaming algorithm for the ℓ_2-Point Query Problem, on query j, which produces $\hat{x}_j$ with $|\hat{x}_j - x_j| \leq \epsilon\|x\|_2$ with arbitrarily large constant probability over the random order and internal randomness. Let us call this latter event $\mathcal{E}$.

We have $\mathbf{E}[F_2] = kn$, and thus by a Markov bound, with probability at least $1 - c^2$, $\|x\|_2 \leq (1/c)\sqrt{kn}$, where $c > 0$ is an arbitrarily small constant. Let us call this event $\mathcal{F}$. Under the event $\mathcal{F}$, one can choose a sufficiently small constant value of $c' > 0$, where $k = c'\epsilon^{-2}$, such that $\epsilon\sqrt{F}_2 < \delta\sqrt{n}$, for a sufficiently small constant $\delta > 0$. Thus, under events $\mathcal{E}$ and $\mathcal{F}$, we learn the sum on the j-th coin up to additive error of at most $\delta\sqrt{n}$ for an arbitrarily small constant $\delta > 0$. Under $\mathcal{G}$, learning the sum on the j-th coin up to additive error at most $\delta\sqrt{n}$ identifies the majority of the jth instance, with arbitrarily large constant probability, for small enough δ. In all, we can solve the k-Coins problem (under events $\mathcal{E}$, $\mathcal{F}$ and $\mathcal{G}$) with arbitrarily large constant probability. Hence, the streaming algorithm must use $\Omega(k \cdot \log n)$ bits of memory, which is $\Omega(k \cdot \log m)$ bits of memory since $m = k \cdot n$ and we assume $m/k = m^{0.1}$. □

ℓ_2-Heavy Hitters. We next show an $\Omega(\epsilon^{-2} \log m)$ bit lower bound for the ℓ_2-Heavy Hitters Problem.

Theorem 27. *Any streaming algorithm which, in the random order model, solves the ℓ_2-Heavy Hitters Problem with sufficiently large constant probability, over the random order and private randomness, requires $\Omega(\epsilon^{-2} \log m)$ bits of memory.*

Proof. We again reduce from the k-Coins problem for a random order for a value $k = \Theta(\epsilon^{-2})$ (Corollary 18). Let x be the underlying d-dimensional vector in the stream, initialized to 0^d, where we assume $d \geq k$. We identify an input, which is 1 or 0, in the i-th copy of the k-Coins problem with an update to the i-th coordinate of x, where $x_i \leftarrow x_i + 1$ if the update is a $+1$, or $x_i \leftarrow x_i - 1$ if the update is a 0. By definition of the k-Coins Problem for a random order, the updates in the stream, even when the set of updates are fixed, appear in a uniformly random order. Using Equation (31), we observe with probability at least $1 - k \cdot e^{-n/8}$, the number of updates to each coordinate is at least $n/2$. Let us call this event $\mathcal{G}$.

We have $\mathbf{E}[F_2] = kn$, and thus by a Markov bound, with probability at least $1 - c^2$, $\|x\|_2 \leq (1/c)\sqrt{kn}$, where $c > 0$ is an arbitrarily small constant. Let us call this event $\mathcal{F}$. Under the event $\mathcal{F}$, one can choose a sufficiently small constant value of $c' > 0$, where $k = c'\epsilon^{-2}$, such that $\epsilon\sqrt{F_2} < \delta\sqrt{n}$, for a sufficiently small constant $\delta > 0$.

The last player for the k-Coins problem holds an index $j \in \{1, 2, \ldots, k\}$, and computes the output of the streaming algorithm for the ℓ_2-Heavy Hitters Problem. By the guarantee of the streaming algorithm, we have that from the output we can produce an estimate $\hat{x}_j$ with $|\hat{x}_j - x_j| \leq \delta\sqrt{n}$ with arbitrarily large constant probability. Indeed, by the ℓ_2-Heavy Hitters guarantee, we have that either j is in the output set of the heavy hitters algorithm, in which case its estimate $\hat{x}_j$ satisfies this guarantee, or it is not in the output set, in which case the estimate $\hat{x}_j = 0$ satisfies this guarantee. Let us call the event that $\hat{x}_j$ satisfies this guarantee event $\mathcal{E}$.

It follows that we learn the sum on the j-th coin up to additive error at most $\delta\sqrt{n}$ for an arbitrarily small constant $\delta > 0$. Under $\mathcal{G}$, learning the sum on the j-th coin up to additive error at most $\delta\sqrt{n}$ identifies the majority of the j-th instance, with arbitrarily large constant probability, for small enough δ. In all, we can solve the k-Coins problem (under events $\mathcal{E}$, $\mathcal{F}$ and $\mathcal{G}$) with arbitrarily large constant probability. Consequently, we can solve the k-Coins problem with arbitrarily large constant probability. Hence, the streaming algorithm must use $\Omega(k \cdot \log n)$ bits of memory, which is $\Omega(k \cdot \log m)$ bits of memory since $m = k \cdot n$, and we assume $m/k = m^{0.1}$. □

5 Algorithms

5.1 ℓ_2-Heavy Hitters and Point Query in Random Order Streams

We use F_2 to denote $\sum_{i=1}^n x_i^2$. We consider, as is standard, the setting in which each stream update is of the form $x_i \leftarrow x_i + 1$ for some value of i. We describe an algorithm for solving a stronger problem than the ℓ_2-Heavy Hitters Problem in the random order model; this version will be useful for the ℓ_2-Point Query Problem as well, which we describe later in this section. It is strengthened in that we achieve relative error for each of the frequencies output, rather than just additive error, without incurring any additional overhead in memory.

Since each update in the stream has the form $x_i \leftarrow x_i + 1$, we can think of being given a stream $a_1, \ldots, a_m$ with each $a_i \in [d] = \{1, 2, \ldots, d\}$, and that the stream is given to us in a random order. We show how to achieve an algorithm which, with probability at least 4/5, outputs a set $S \subseteq \{1, 2, \ldots, d\}$ such that for all $i \in [d]$,

1. if $x_i^2 \geq \epsilon^2 F_2$, then $i \in S$. Further, for all $i \in S$, we output an estimate $\hat{x}_i$ with $\hat{x}_i = (1 \pm \epsilon)x_i$, and
2. if $x_i^2 \leq \frac{\epsilon^2}{2} F_2$, then $i \notin S$.

The probability of 4/5 can be increased to any constant strictly less than 1 with minor modifications.

We first describe our main algorithm $\textsc{Main1}(\tilde{F}_2, m, a_1, \ldots, a_m)$ assuming we are given an estimate $\tilde{F}_2$ for which $F_2 \leq \tilde{F}_2 \leq 1.1F_2$, as well as given $m = F_1 = \sum_{i=1}^{d} x_i$. We later give our main algorithm Main which works without being given $\tilde{F}_2$ and m.

We start with the intuition for our first algorithm Main1. The algorithm takes in the above values $\tilde{F}_2$ and m, together with the stream elements $a_1, \ldots, a_m$ one at a time. The variable H denotes the output, which we initialize to the empty set. The algorithm then breaks the stream into t contiguous blocks $B_1, \ldots, B_t$. It defines hash functions $h^1, \ldots, h^{10\ln(1/\epsilon)}$ with the following meaning. For each item j, if $h^\ell(j) = i$, then B_i is one of the $10\ln(1/\epsilon)$ blocks assigned to item j. If further j actually occurs in the block B_i, then j is excited in B_i, as per the definition of excited; see Definition 2. The set S^i is used to store the elements j that are excited in B_i. For memory considerations, we do not store the actual item identities j in S^i, but rather only their hashed values, under a hash function g. We also initialize a flag $c^i(g(a_j))$ to 1 for each hash of an item j that was excited in block B_i; this flag is useful when processing future blocks. Note also that the variable e^i for block i is used to indicate an error, namely, that there were too many items in block B^i that were excited, in which case $e^i = 1$ informs us that we should not pay attention to block B^i going forward in the stream.

When processing a block B^i, we check to see if there is an item j in B^i which has a hash value equal to k (and that $h^q(j) = i - 1$ for some $q \in [10\ln(1/\epsilon)]$), and for which k is also in S^{i-1}. Note that heavy items are likely to occur in most blocks, and so for many values of i, a heavy item will occur in both B^{i-1} and B^i. Moreover, this should also be the case for a few blocks B^{i-1} in which the heavy item is excited. When doing this check, we also check that there was no error indicated in block B^{i-1}, namely, that $e^{i-1} = 0$. In this case, we flag the hash value k, setting $c^{i-1}(k) = c^{i-1}(k) + 1$. Finally, after processing B^i, we check if there was exactly one item in S^{i-1} that was flagged while processing the i-th block. If so, then it is likely to be a heavy hitter, as it is very unlikely for non-heavy items to occur in consecutive blocks. However, it is still the case that with probability $\text{poly}(\epsilon)$, a non-heavy hitter may pass these checks. Thus, we really need to make sure the item is a heavy hitter, so we initialize an instance of our Identify subroutine to verify this fact.

We next explain the pseudocode for our Identify subroutine. We initialize an identity variable id which indicates the identity of the item we are tracking. We then check, walking through the next $100\ln(1/\epsilon)$ blocks, if there is a unique item j which occurs in one of those blocks and with the hash value (under g) equal to value k, which was an input to the Identify subroutine. We keep checking such blocks until we find the first such identity j, or fail to find one. While looking at $100\ln(1/\epsilon)$ blocks is enough to obtain an error probability of $\text{poly}(\epsilon)$, this error probability is not enough to rule out false positives over a stream of length $\text{poly}(m)$, if m is very large compared to ϵ. Having passed this test, we then check if item j occurs in a $(1 - 1/e)$ fraction of the the next $4000\epsilon^{-2}\ln m$ blocks; if it does then we have that with probability $1 - 1/\text{poly}(m)$, that j really is a heavy hitter. We do not want to directly run this high probability test without first passing the lower probability test, for memory considerations; this is why we have a two-stage check. We note that, while we perform this latter check, we also obtain a very precise estimate to the count of item j, which is included in our output.

Finally we explain the pseudocode for our Main algorithm. We maintain an approximate value $\tilde{F}_2^{(k)}$ of F_2 at each time k. We also maintain a counter to keep track of k. Each time k increases additively by the next power of 2, we initialize a new instance of Main1, denoted $\textsc{Main1}^i$. Note that the reason for doing this is that we always run our instances of Main1 on a value of m which is within a factor of 2 of the actual value of m, and also because of the random order property, the value $\tilde{F}_2^{(k)}$ when a new instance of Main1 is invoked is a good estimate for the actual F_2-value of the substream that Main1 is invoked on. Note that if the stream ends before the i-th instance of Main1 has completed, we return the output H^{i-1} of the previous instance of Main1. We also discard memory contents associated with earlier H^j, for $j < i-1$, as they are not needed for the output, and allow us to reuse memory.

We now formally give the pseudocode and analysis for our algorithms.

Algorithm 1 Main1($\tilde{F}_2, m, a_1, \ldots, a_m$) algorithm given $\tilde{F}_2$ and m

```
$H \leftarrow \emptyset$
Partition the stream into $t = \epsilon\sqrt{\tilde{F}_2}$ contiguous blocks each of length $\frac{F_1}{\epsilon\sqrt{\tilde{F}_2}}$. Denote them by $B_1, \ldots, B_t$
Let $h^1, \ldots, h^{10\ln(1/\epsilon)} : [n] \to [t]$ and $g : [n] \to [\epsilon^{-13}\ln^2(dm)]$ all be chosen independently from each other,
  and each function is drawn randomly from a pairwise independent family of hash functions.
while processing block $B_i$, do
    $S^i \leftarrow \emptyset$; $e^i = 0$; initialize $c^i$ to a vector of all 0s.
    First process all of the items $a_j \in B_i$.
    while processing item $a_j$, do
        If $e^i = 0$ and $h^q(a_j) = i$ for some $q \in [10\ln(1/\epsilon)]$, then $S^i \leftarrow S^i \cup \{g(a_j)\}$
        If $|S^i| > 100\epsilon^{-2}\ln(1/\epsilon)$, then $e^i = 1$
        If $g(a_j) \in S^{i-1}$, $h^q(a_j) = i-1$ for some $q \in [10\ln(1/\epsilon)]$ and $e^{i-1} = 0$,
        then $c^{i-1}(g(a_j)) = c^{i-1}(g(a_j)) + 1$
    end while
    if $e^{i-1} = 0$, and there is a unique $k \in S^{i-1}$ with $c^{i-1}(k) = 1$ then
        if there is no Identify instance already running (and not yet completed) and $|H| \leq 2\epsilon^{-2}$ then
            $H \leftarrow H \cup$ Identify($\{h^q\}_q, g, i, k, m, a_{iF_1/(\epsilon\sqrt{\tilde{F}_2})+1}, \ldots, a_m$)
        end if
    end if
end while
```

Finally, we put it all together with our main algorithm Main. Let F_2-sketch be an algorithm which, for any particular time k in the stream, succeeds with probability at least 99/100 in returning an estimate $\tilde{F}_2^{(k)}$ with $F_2^{(k)} \leq \tilde{F}_2^{(k)} \leq 1.01F_2^{(k)}$, where $F_2^{(k)}$ is the F_2 value after processing the first k stream elements. Note that the algorithm succeeds at any particular time with probability 99/100, but not necessarily at all times k; however this suffices for our purposes. The algorithm of [3] achieves this in $O(\log(dm))$ bits of space.

Our algorithm Main described above will be run assuming that $m \geq \text{poly}((\log d)\epsilon^{-1})$. Here m is the value of F_1 at the end of the stream. If this condition does not hold, then we can hash $[d]$ to $\text{poly}((\log d)\epsilon^{-1})$ and run the CountSketch streaming algorithm [11] on the hashed items. Note that with arbitrarily large constant probability, all of the distinct items in the stream will have distinct hashed identities. In addition, we maintain a heap data structure, as in [11], with the top $O(1/\epsilon^2)$ actual (non-hashed) identities in $[d]$. The CountSketch data structure uses $O(\epsilon^{-2}(\log(1/\epsilon) + \log\log d)\log m) = O(\epsilon^{-2}(\log(1/\epsilon) + \log\log d)(\log(1/\epsilon + \log\log d)))$ bits of space to run on the hashed identities and to store counters in the stream. Given our heap, we can recover the actual identities using $O(\epsilon^{-2}\log d)$ bits of memory. Note that here it is essential that distinct items in the stream have distinct hashed identities. This gives $O(\epsilon^{-2}(\log d + \log^2(1/\epsilon)))$ total bits of space, which will be within our budget.

This space bound is sufficient for us, so in our analysis below, we will assume that $m \geq \text{poly}((\log d)\epsilon^{-1})$ for a sufficiently large polynomial.

We note that for the ℓ_2-Point Query Problem, when $m < \text{poly}((\log d)\epsilon^{-1})$ we do not need to store the actual identities, so we do not need the additional heap data structure above. Indeed, given a query j, we compute its hash value, and then obtain an approximation to its count using the CountSketch data structure on hashed identities using $O(\epsilon^{-2}(\log^2(1/\epsilon) + \log^2\log(dm)))$ bits of memory, as described above.

Space Complexity. We start by bounding the overall memory required of Main($a_1, \ldots, a_m$).

Theorem 28. Main($a_1, \ldots, a_m$) *can be implemented using* $O(\epsilon^{-2}(\log d + \log^2(1/\epsilon) + \log^2\log m) + \log m)$ *bits of space.*

Algorithm 2 $\textsc{Identify}(\{h^q\}_q, g, i, k, m, a_{iF_1/(\epsilon\sqrt{\tilde{F}_2})+1}, \dots, a_m)$ subroutine

```
$id \leftarrow \emptyset$ and $\ell = 1$
**repeat**
    **if** $|\{j \in B_{i+\ell} \mid g(j) = k$ and $h^q(j) = i - 1$ for some $q \in [10\ln(1/\epsilon)]\}| = 1$ **then**
        $id \leftarrow j$
    **else** $\ell \leftarrow \ell + 1$.
    **end if**
**until** $id \neq \emptyset$ or $\ell > 100\ln(1/\epsilon)$.
**if** $id = \emptyset$ **then**,
    return $\emptyset$.
**end if**
$p = 4000\epsilon^{-2}\ln m$.
Let $\xi$ be the number of occurrences of $id$ in the multiset $\uplus_{v=1}^{p} B_{i+\ell+v}$.
**if** $\xi \geq (1.01/\sqrt{2})p$ **then**,
    $\hat{x}_{id} = \frac{\epsilon\sqrt{\tilde{F}_2}}{p} \cdot \xi$.
    return $(id, \hat{x}_{id})$
**else** return $\emptyset$.
**end if**
```

Algorithm 3 $\textsc{Main}(a_1, \dots, a_m)$ algorithm

```
Initialize an instance of an $F_2$-sketch with $\tilde{F}_2^{(k)}$ being its output at time $k$
**while** processing $a_k$, if $k = \sum_{j=0}^{i-1} 2^j$ for an $i \geq 1$ **do**,
    Let $H^{i-1}$ be the output of $\textsc{Main}^{i-1}$ with parameter $\frac{\epsilon}{10}$, or if $i = 1$ let $H^0 = a_1$
    Initialize an instance $\textsc{Main1}^i(1.01 \cdot \tilde{F}_2^{(k)}, 2^i, a_{k+1}, a_{k+2}, \dots a_{k+2^i})$ with parameter $\frac{\epsilon}{10}$
    Discard $H^{i-2}$ and the corresponding memory contents associated with the $\textsc{Main1}^{i-2}$ instance
**end while**
At the end of the stream, let $i$ be largest for which we are running an instance of $\textsc{Main1}^i$
Return those $(j, \frac{m}{2^i} \cdot \hat{x}_j)$ in $H^{i-1}$ for which $\frac{m}{2^i} \cdot \hat{x}_j \geq (1 - \epsilon/5)\epsilon\frac{\sqrt{\tilde{F}_2^{(m)}}}{1.01}$
```

Proof. We use $O(\log(dm))$ bits of memory for the F_2-sketch described above.

Next, note that if $m \leq \text{poly}((\log d)\epsilon^{-1})$, then using the analysis for CountSketch discussed above, the total memory is as claimed. Otherwise, we can assume that $m > \text{poly}((\log d)\epsilon^{-1})$ for a sufficiently large polynomial.

Line 1 takes $O(\log m)$ bits of space [6], and we can maintain the counter k with $O(\log m)$ bits of space. For the remaining steps, because of the discarding of previous instances of Main1, the space is at most twice the space complexity of MAIN1$^i(\tilde{F}_2^{(k)}, 2^i, a_t, \ldots, a_m)$, maximized over i.

To bound the space complexity of Main1$(\tilde{F}_2^{(k)}, 2^i, a_t, \ldots, a_m)$, note that we can store all of the h^q and g using $O((\log d)\log(1/\epsilon))$ bits, since they are drawn from pairwise independent hash function families and $1 \leq q \leq 10\ln(1/\epsilon)$. Here we use the standard fact that functions from such families can be stored with $O(\log d)$ bits [9].

In the while loop in line 4 of Main1, because of the filtering in the while loop in line 7, each S^i satisfies $|S^i| \leq 100\epsilon^{-2}\ln(1/\epsilon)$, and moreover, each element $g(a_j) \in S^i$ can be stored using $O(\log(1/\epsilon) + \log\log m)$ bits. Consequently, each S^i can be stored using $O(\epsilon^{-2}(\log(1/\epsilon))(\log(1/\epsilon) + \log\log m))$ bits. Because of line 12, we only need to maintain the last two sets S^{i-1} and S^i, giving a total of $O(\epsilon^{-2}(\log(1/\epsilon)(\log(1/\epsilon) + \log\log m)))$ bits for storing both S^{i-1} and S^i in the stream.

It remains to analyze the space complexity of line 14. While processing block B_i, at most one instance of Identify will be created. It is not hard to see that Identify can be implemented using $O((\log d)(\log(1/\epsilon)) + \log\log m)$ bits of space to store $\{h^q\}_q, g, i, k$, as well as a few counters. Notice also that a given instance of Identify will be executed over the next $4000\epsilon^{-2}\ln m$ blocks, but we enforce in line 13 that there is never more than one instance of Identify running at any time during the stream.

Finally, we must account for the memory required to store H in the stream. Note that we enforce that $|H| \leq 2\epsilon^{-2} + 1$ in line 13, and so this also takes $O(\epsilon^{-2}\log d)$ bits of space.

In total, we meet the desired space bound. □

Correctness. We will need a tail bound for hypergeometric random variables, which follows from negative dependence.

Fact 2. *(Theorem 4 of [20]) Suppose we draw n balls from an urn with M green balls and N total balls. Let i be the number of green balls we draw. For $\epsilon > 0$ less than a sufficiently small constant,*

$$\Pr\left[\left|i - \frac{nM}{N}\right| > \epsilon\frac{nM}{N}\right] \leq 2e^{-\frac{\epsilon^2}{3}\frac{nM}{N}}.$$

Notice that $\mathbf{E}[i] = \frac{nM}{N}$ in Fact 2, so this is similar to the usual multiplicative Chernoff bound.

Before showing correctness of Main1$(\tilde{F}_2, m, a_1, \ldots, a_m)$ and Main$(a_1, \ldots, a_m)$, we establish properties of our Identify$(\{h^q\}_q, g, i, k, m, a_{iF_1/(\epsilon\sqrt{\tilde{F}_2})+1}, \ldots, a_m)$ subroutine. This is needed in our analysis of Main1 and Main.

Let HH be the set of items j for which $x_j^2 \geq \epsilon^2 F_2$, i.e., the actual heavy hitters.

The Identify Subroutine. We first argue that items with small frequencies are never returned by any invocation of Identify.

Lemma 4. *(No False Positives) With probability $1 - 1/m$, there is no instance invoked of* Identify$(\{h^q\}_q, g, i, k, m, a_{iF_1/(\epsilon\sqrt{\tilde{F}_2})+1}, \ldots, a_m)$ *which ever returns an element $j \neq \emptyset$ for which $x_j^2 \leq \frac{\epsilon^2 F_2}{2}$.*

Proof. Consider an instance Identify$(\{h^q\}_q, g, i, k, m, a_{iF_1/(\epsilon\sqrt{\tilde{F}_2})+1}, \ldots, a_m)$. In order for $j \neq \emptyset$ and $x_j^2 \leq \frac{\epsilon^2 F_2}{2}$, we must have that $id = j$ in line 4 of the subroutine.

Thus, to prove the lemma, it suffices to show that, with probability $1 - 1/m$, simultaneously *for every* sequence of $4000\epsilon^{-2}\ln m$ blocks $B_{z+1}, B_{z+2}, \ldots, B_{z+4000\epsilon^{-2}\ln m}$, the number of occurrences of such a j among these blocks is less than $(1.01/\sqrt{2}) \cdot 4000\epsilon^{-2} \ln m$. This implies that in line 11 of any invocation of IDENTIFY, if an element $j \neq \emptyset$ is returned, then necessarily $x_j^2 > \frac{\epsilon^2 F_2}{2}$.

Fix an arbitrary j with $x_j^2 \leq \frac{\epsilon^2 F_2}{2}$ and an arbitrary sequence of $4000\epsilon^{-2} \ln m$ blocks, denoted by $B_{z+1}, B_{z+2}, \ldots, B_{z+4000\epsilon^{-2} \ln m}$. Let ξ be the total number of occurrences of j in the multiset union of $B_{z+1}, B_{z+2}, \ldots, B_{z+4000\epsilon^{-2} \ln m}$.

Note that ξ is hypergeometrically distributed, where there are $M = x_j$ occurrences of item j among $N = m$ items, and we select $n = \frac{m}{\epsilon\sqrt{\tilde{F}_2}} \cdot 4000\epsilon^{-2} \ln m = \frac{4000 m \ln m}{\epsilon^3 \sqrt{\tilde{F}_2}}$ of them. Note that $\frac{nM}{N} = \frac{4000 x_j \ln m}{\epsilon^3 \sqrt{\tilde{F}_2}} \leq (4000/\sqrt{2})\epsilon^{-2} \ln m$, using that $x_j \leq \epsilon\sqrt{F_2/2} \leq \epsilon\sqrt{\tilde{F}_2/2}$.

By Fact 2,

$$\begin{aligned} \Pr[Y \geq (1+\gamma)\frac{nM}{N}] &\leq 2e^{-2\gamma^2 \frac{nM}{N}} \\ &\leq 2e^{-2\gamma^2 (4000/\sqrt{2})\epsilon^{-2} \ln m} \\ &\leq \frac{1}{m^6}, \end{aligned}$$

where $\gamma > 0$ is an arbitrarily small constant and the $1/m^6$ is arbitrary. We choose γ so that

$$\Pr[\xi \geq 4000 \cdot (1.01/\sqrt{2})\epsilon^{-2} \ln m] \leq \frac{1}{m^6}.$$

There are at most m sequences of $4000\epsilon^{-2} \ln m$ contiguous blocks. Also, the number of items j is at most m, and so the lemma follows by a union bound over all pairs of sequences of blocks and all j with $x_j^2 \leq \frac{\epsilon^2 F_2}{2}$. □

Corollary 29. *($|H|$ is Small) With probability $1 - 1/m$, at all times during the stream we have $|H| \leq 2\epsilon^{-2}$.*

Proof. By Lemma 4, with probability at least $1 - 1/m$, throughout the entire stream any item j added to H necessarily satisfies $x_j^2 \geq \frac{\epsilon^2 F_2}{2}$. Since there can be at most $2\epsilon^{-2}$ distinct such items j, the corollary follows. □

We next argue that heavy hitters occur many times in streaks of consecutive blocks.

Lemma 5. *(Heavy Hitters Occur in Streaks) With probability $1 - 1/m^{.05}$, simultaneously for every $j \in HH$ and every sequence of $4000\epsilon^{-2} \ln m$ consecutive blocks $B_{z+1}, \ldots, B_{z+4000\epsilon^{-2} \ln m}$, item j occurs in at least $4000 \cdot (1.01/\sqrt{2})\epsilon^{-2} \ln m$ of these blocks.*

Proof. Fix an arbitrary $j \in HH$ and an arbitrary sequence of $4000\epsilon^{-2} \ln m$ blocks $B_{z+1}, B_{z+2}, \ldots, B_{z+4000\epsilon^{-2} \ln m}$. Since $j \in HH$, $x_j^2 \geq \epsilon F_2$.

As in the proof of Lemma 4, ξ is hypergeometrically distributed, where there are $M = x_j$ occurrences of item j among $N = m$ items, and we select $n = \frac{m}{\epsilon\sqrt{\tilde{F}_2}} \cdot 4000\epsilon^{-2} \ln m = \frac{4000 m \ln m}{\epsilon^3 \sqrt{\tilde{F}_2}}$ of them. Note that

$$\frac{nM}{N} = \frac{4000 x_j \ln m}{\epsilon^3 \sqrt{\tilde{F}_2}} \geq \frac{4000}{\sqrt{1.01}} \cdot \epsilon^{-2} \ln m,$$

using that $x_j \geq \epsilon\sqrt{F_2} \geq \epsilon\sqrt{\tilde{F}_2/1.01}$.

By Fact 2,

$$\begin{aligned} \Pr\left[Y \leq (1-\gamma)\frac{nM}{N}\right] &\leq 2e^{-2\gamma^2 \frac{nM}{N}} \\ &\leq 2e^{-2\gamma^2 (4000/\sqrt{2})\epsilon^{-2} \ln m} \\ &\leq \frac{1}{m^6}, \end{aligned}$$

where $\gamma > 0$ is an arbitrarily small constant and the $1/m^6$ is arbitrary. We choose γ so that $\Pr[\xi \geq 4000 \cdot (1.01/\sqrt{2})\epsilon^{-2} \ln m] \geq 1 - \frac{1}{m^6}$.

As in the proof of Lemma 4, the number of sequences of $4000\epsilon^{-2} \ln m$ blocks is at most m. Also, the number of heavy hitters j is at most ϵ^{-2}, which is at most m^C, for an arbitrarily small constant $C > 0$, assuming $m \geq \text{poly}((\log d)\epsilon^{-1})$ for a sufficiently large polynomial. The lemma now follows by a union bound over all pairs of sequences of $4000\epsilon^{-2} \ln m$ blocks and heavy hitters. □

Lemma 6. *With probability* $1 - 1/m^{98}$*, for every* j *for which* $x_j^2 \geq \frac{\epsilon^2}{2} F_2$ *and for every sequence of* $p = 4000\epsilon^{-2} \ln m$ *contiguous blocks in the stream, the number* ξ *of occurrences of* j *in total in the multiset union* $\uplus_{v=1}^{p} B_{i+\ell+v}$ *of these blocks sastisfies:*

$$\xi = (1 \pm \epsilon) \cdot \frac{p \cdot x_j}{\epsilon\sqrt{\tilde{F}_2}}.$$

Proof. Fix a j for which $x_j^2 \geq \frac{\epsilon^2}{2} F_2$ and consider any sequence of $4000\epsilon^{-2} \ln m$ contiguous blocks in the stream.

We apply Fact 2 with the N of that fact equal to our m, the M of that fact equal to our x_j for some j, the n of that fact equal to $\frac{pm}{\epsilon\sqrt{\tilde{F}_2}} = \frac{(4000 \ln m)m}{\epsilon^3\sqrt{\tilde{F}_2}}$.

Then,

$$\Pr\left[\xi = (1 \pm \epsilon)\frac{p \cdot x_j}{\epsilon\sqrt{\tilde{F}_2}}\right] \leq 2e^{-\frac{\epsilon^2}{3}\frac{nM}{N}} \tag{53}$$

$$= 2e^{-\frac{\epsilon^2}{3}\cdot\frac{4000\epsilon^{-2}\ln m}{\sqrt{1.1}}} \tag{54}$$

$$< \frac{1}{m^{100}}, \tag{55}$$

using that $x_j \geq \epsilon\sqrt{F_2} \geq \epsilon\sqrt{\tilde{F}_2}/\sqrt{1.1}$. We can union bound over the at most $2\epsilon^{-2}$ such different j and at most m such contiguous blocks to conclude that this holds simultaneously for all such j and all such contiguous sequences of blocks with probability at least $1 - 1/m^{98}$, using that $m > 2\epsilon^{-2}$. □

The Main1 **Algorithm.** We next show the correctness of Main1$(F_2, m, a_1, \ldots, a_m)$ given F_2 and m in advance. The following is the formal version of Definition 2 given earlier.

Definition 4. *For a block* B_i*, an element* $j \in B_i$ *is said to be excited if* $h^q(j) = i$ *for some* $q \in [10\ln(1/\epsilon)]$.

The next lemma is our main technical lemma concering our algorithm.

Lemma 7. *(Heavy Hitters Included in the Sets* S^i*, and* Identify *is Invoked on Every Heavy Hitter) With probability at least* 39/40*, simultaneously for all* $j \in HH$*, there is some* $q \in [10\ln(1/\epsilon)]$ *for which:*

1. $h^q(j) = i$ *and for which both* $e^i = 0$ *and* $g(j) \in S^i$ *after processing* B_i *in line 10*
2. $g(j)$ *is the unique value* $k \in S^i$ *with* $c^i(k) = 1$
3. *when line 14 is invoked on* $k = g(j)$*, there is no* Identify *instance already running (and not yet completed), and the value of* $|H|$ *will be at most* $2\epsilon^{-2}$.

Proof. Note by Corollary 29, with probability at least $1 - 1/m$, at all times during the stream we have $|H| \leq 2\epsilon^{-2}$, so we will assume this holds and add $1/m$ to our final error probability.

We now fix a $j \in HH$, and show the events in the lemma statement hold. We will later apply a union bound over all $j \in HH$.

We define a sequence of events in this proof. It will be convenient to define I to be the set of blocks i for which $h^q(j) = i$ for some $q \in [10\ln(1/\epsilon)]$. Note that here we are fixing a particular $j \in HH$, and the lemma concerns a particular $j \in HH$.

For a block B_i, we let $\text{seq}(B_i)$ denote the set of $4000\epsilon^{-2} \ln m$ blocks immediately preceding B_i, including B_i.

Event $\mathcal{E}$. For any i, the expected number of excited items in a block B_i is

$$\frac{|B_i| 10 \ln(1/\epsilon)}{t} = \frac{\epsilon^{-1} F_1 \cdot 10 \ln(1/\epsilon)}{\epsilon \tilde{F}_2} \leq 10\epsilon^{-2} \ln(1/\epsilon),$$

since $F_2 \geq F_1$ and $\tilde{F}_2 \geq F_2$. Conditioned on an item j' in a block B_i being excited, the event that an item $j'' \neq j'$ in B_i is excited is negatively correlated, since each h^q is drawn from a pairwise-independent hash function family and also if j' occurs in B_i then $j'' \leq j'$ is less likely to occur in B_i. Therefore, if X is the number of excited items in B_i, then $\mathbf{Var}[X] \leq 10\epsilon^{-2} \ln(1/\epsilon)$. By Chebyshev's inequality,

$$\Pr[|X – \mathbf{E}[X]| > 90\epsilon^{-2} \ln(1/\epsilon)] \leq \frac{10\epsilon^{-2} \ln(1/\epsilon)\epsilon^4}{8100 \ln^2(1/\epsilon)} = \frac{\epsilon^2}{810 \ln(1/\epsilon)}.$$

Finally, note that S^i is the set of $g(a_j)$ for those a_j that are excited, and so the size of S^i is at most the number of excited items in B_i. This shows that $e^i = 0$. Since there are $10 \ln(1/\epsilon)$ choices of q, it follows by a union bound that $e^i = 0$ for every $i \in I$, with probability at least $1 - \epsilon^2/81$.

For any i, let Y be the number of excited items in $\text{seq}(B_i)$. Then $\mathbf{E}[Y] \leq 10\epsilon^{-2} \ln(1/\epsilon) \cdot 4000\epsilon^{-2} \ln m$, and again since the covariance is non-positive, $\mathbf{Var}[Y] \leq 40000\epsilon^{-4} \ln(1/\epsilon) \ln m$. By Chebyshev's inequality

$$\Pr[|Y - \mathbf{E}[Y]| > 9 \cdot 10 \cdot 4000\epsilon^{-4} \ln(1/\epsilon) \ln m] \leq \frac{10 \cdot 4000\epsilon^{-4} \ln(1/\epsilon) \ln m}{81 \cdot 10^2 \cdot 4000 \cdot \epsilon^{-8} \ln^2(1/\epsilon) \ln^2 m} \leq \frac{\epsilon^4}{81 \cdot 10 \cdot 4000 \ln(1/\epsilon) \ln m}.$$

Since there are $10 \ln(1/\epsilon)$ choices of q, it follows by a union bound that with probability at least $\frac{\epsilon^4}{81 \cdot 4000 \ln m}$ that for every $i \in I$, there at most $100 \cdot 4000\epsilon^{-4} \ln(1/\epsilon) \ln m$ excited items in $\text{seq}(B_i)$.

We define the event $\mathcal{E}$ to be that

1. $e^i = 0$ for each $i \in I$, and
2. the number of excited items in block B_i is at most $100\epsilon^{-2} \ln(1/\epsilon)$ for every $i \in I$
3. the number of excited items in $\text{seq}(B_i)$ is at most $100 \cdot 4000 \cdot \epsilon^{-4} \ln(1/\epsilon) \ln m$, for every $i \in I$.

By the above,

$$\Pr[\mathcal{E}] \geq 1 - \epsilon^2/81 - \frac{\epsilon^4}{81 \cdot 4000 \ln m} \geq 1 - \epsilon^2/80. \tag{56}$$

We condition on $\mathcal{E}$ in what follows and add $\epsilon^2/80$ to the overall error probability.

Note that conditioned on $\mathcal{E}$, we also have $|S^i| \leq 100\epsilon^{-2} \ln(1/\epsilon)$ for each i for which $h^q(j) = i$ for some q (since if the number of excited items is at most $100\epsilon^{-2} \ln(1/\epsilon)$, then this also upper bounds each $|S^i|$.)

Event $\mathcal{F}$. We next define the event $\mathcal{F}$, which is very similar to event $\mathcal{E}$. The same calculation shows that the expected number of items j' in B_{i+1} for which $h^q(j') = i$ for some q is

$$\frac{|B_{i+1}| 10 \ln(1/\epsilon)}{t} \leq 10\epsilon^{-2} \ln(1/\epsilon).$$

Conditioned on an item j' in a block B_{i+1} satisfying $h^q(j') = i$ for some q, the event that an item $j'' \neq j'$ in B_{i+1} also sastisfies $h^q(j'') = i$ for some q is independent since each h^q is drawn from a pairwise-independent hash function family. Therefore, if X is the number of items j' in B_{i+1} for which $h^q(j') = i$ for some q, we have $\mathbf{Var}[X] \leq 10\epsilon^{-2} \ln(1/\epsilon)$. By Chebyshev's inequality,

$$\Pr[|X – \mathbf{E}[X]| > 90\epsilon^{-2} \ln(1/\epsilon)] \leq \frac{10\epsilon^{-2} \ln(1/\epsilon)\epsilon^4}{8100\epsilon^{-4} \ln^2(1/\epsilon)} = \frac{\epsilon^2}{810 \ln(1/\epsilon)}.$$

By a union bound over $q \in [10 \ln(1/\epsilon)]$, for all $i \in I$ the number of items j' in B_{i+1} for which $h^q(j') = i$ is at most $100\epsilon^{-2} \ln(1/\epsilon)$.

We will also need to reason about $\mathrm{seq}(B_i)$ for each $i \in I$. For a given such i, let Y be the number of items j' in B_{r+1}, for some B_r in $\mathrm{seq}(B_i)$, for which $h^q(j') = r$ for some q. We have $\mathbf{E}[Y] \leq 10\epsilon^{-2}\ln(1/\epsilon) \cdot 4000\epsilon^{-2} \ln m$, and again by Chebyshev's inequality (using negative association):

$$\Pr[|Y - \mathbf{E}[Y]| > 90\epsilon^{-4}\ln(1/\epsilon) \cdot 4000 \ln m] \leq \frac{10\epsilon^{-4}\ln(1/\epsilon) \cdot 4000 \ln m}{8100\epsilon^{-8}\ln^2(1/\epsilon)4000^2 \ln^2 m} = \frac{\epsilon^4}{810\ln(1/\epsilon)4000\ln m}.$$

Union bounding over all $i \in I$, we have that with probability at least $1 - \frac{\epsilon^4}{810 \ln(m)}$, simultaneously for all $i \in I$, the number of items j' in B_{r+1} for some B_r in $\mathrm{seq}(B_i)$ for which $h^q(j') = t$ for some q is at most $100\epsilon^{-4}\ln(1/\epsilon) \cdot 4000 \ln m$.

We define the event $\mathcal{F}$ to be that:

1. for all $i \in I$, the number of items j' in B_{i+1} for which $h^q(j') = i$ is at most $100\epsilon^{-2}\ln(1/\epsilon)$.
2. for all $i \in I$, the total number of items j' in B_{r+1} for some B_r in $\mathrm{seq}(B_i)$, for which $h^q(j') = r$ for some q and some r, is at most $100\epsilon^{-4}\ln(1/\epsilon) \cdot 4000 \ln m$.

By a union bound,

$$\Pr[\mathcal{F}] \geq 1 - \epsilon^2/81 - \epsilon^4/(810\ln(m)) \tag{57}$$

We condition on $\mathcal{F}$ in what follows and add $\epsilon^2/81 + \epsilon^4/(810\ln(m))$ to the overall error probability.

Note that event $\mathcal{F}$ is needed because there could be an item $j' \in B_{i+1}$ for which $h^q(j') = i$ for some q, but j' does not actually occur in B_i. In this case, there would be a problem if $g(j') = g(j'')$ for an item j'' which is excited in B_i.

Event $\mathcal{G}$. Let r be the index of a block B_r occurring in $\mathrm{seq}(B_i)$ for some $i \in I$. Let W be the set of items j' for which either j' is excited in B_r for which $h^q(j) = r$ for some q, or j' is in B_{r+1} and $h^q(j') = r$ for some q. In other words, W is the union of the set of items j' that are excited in some block in $\mathrm{seq}(B_i)$ (including B_i) for some $i \in I$, together with the set of items j' for which j' occurs in B_{r+1} and $h^q(j') = r$ for some B_r occuring in $\mathrm{seq}(B_i)$ for some $i \in I$.

We now define the following event $\mathcal{G}$: the hash values $g(j')$ for all $j' \in W$ are distinct. Because of events $\mathcal{E}$ and $\mathcal{F}$, the size of W is at most $O(\epsilon^{-4}\ln^2(1/\epsilon)\ln m)$. Since the range of g is $\epsilon^{-13}\ln^2 m$, the probability there exist two items $j' \neq j'' \in W$ which hash to the same value under g is much less than, say, ϵ^3, and thus,

$$\Pr[\mathcal{G}] \geq 1 - \epsilon^3. \tag{58}$$

We condition on $\mathcal{G}$ in what follows and add ϵ^3 to the overall error probability.

One consequence of $\mathcal{E} \wedge \mathcal{F} \wedge \mathcal{G}$ is the following. Consider each item j' that occurs in B_{r+1} and satisfies $h^q(j') = r$ for some q, where here $r \in \mathrm{seq}(B_i)$ for some $i \in I$. Then if $g(j') \in S^r$, then j' is excited in B_r. This event says it cannot be that there is an item j' which occurs in B_{r+1} which has a hash value $g(j')$ which occurs in S^r, unless j' is excited in B_r, meaning, j' must also occur in B_r rather than just having a hash value $g(j')$ which is equal to some $g(j'')$ for a $j'' \neq j'$ which is excited in B_r. Indeed, this follows since $g(j'')$ and $g(j')$ are distinct, because both j'' and j' belong to W and $j'' \neq j'$.

Although we will show below that it is likely that $c^i(k) = 1$ where $k = g(j)$ and j occurs in B_{i+1} for some $i \in I$ (recall we have fixed the heavy hitter j for now, and will later perform a union bound over all heavy hitters j), the worry is that there may be some $k' \neq k$ for which $c^i(k')$ is also equal to 1, causing the second conclusion of the lemma not to hold. We rule this out by simply arguing that there is *no* j' for which j' is excited in B_i and j' occurs in B_{i+1}, for any $i \in I$. In fact, to conclude the third part of the lemma, we will need this not only to hold for the B_i for $i \in I$, but also for all B_r in $\mathrm{seq}(B_i)$ for some $i \in I$.

Event $\mathcal{H}$. Consider the event $\mathcal{H}$ that for all $j' \neq j$ for which $x_{j'}^2 \geq \epsilon^{12} F_2$ and for all $q, q' \in [10 \ln(1/\epsilon)]$, we have $|h^q(j) - h^{q'}(j')| > 4000\epsilon^{-2} \ln m$. Note that this in particular implies $h^q(j) \neq h^{q'}(j')$, which would suffice for the second part of the lemma, though we need this stronger form for the third part of the lemma.

The chance that $|h^q(j) - h^{q'}(j')| \leq 4000\epsilon^{-2} \ln m$ for a fixed j', q and q' is at most $2 \cdot \frac{4000\epsilon^{-2} \ln m}{t}$, and therefore the probability that $\mathcal{H}$ does not occur is at most:

$$O(\epsilon^{-12}) \cdot O(\ln^2(1/\epsilon)) \cdot \frac{4000 \ln m}{t} \leq \epsilon^3,$$

where the inequality holds since $t = \epsilon\sqrt{\tilde{F}_2}$, which by our assumption is at least a sufficiently large poly$((\ln m)/\epsilon)$. Consequently

$$\Pr[\mathcal{H}] \geq 1 - \epsilon^3. \tag{59}$$

We condition on $\mathcal{H}$ in what follows and add ϵ^3 to the overall error probability.

Note that $\mathcal{H}$ rules out the possibility that a "heavy" j' is excited in one of the same blocks as j, i.e., a block $i \in I$. In fact, it is not even excited in seq(B_i) for any $i \in I$.

Event $\mathcal{I}$. There will naturally be items of small frequency that are excited in a block B_i in which j is excited, or a block B_r in seq(B_i) for some $i \in I$. However, it is very unlikely such items occur in two consecutive blocks. We now create an event $\mathcal{I}$ to capture this, so we escape this possibility.

For an item j' with $x_{j'}^2 < \epsilon^{12} F_2$, its expected number of occurrences in any particular block B_r is at most $\frac{x_{j'}}{t} = \frac{x_{j'}}{\epsilon\sqrt{\tilde{F}_2}} = \frac{\epsilon^5}{\ln^2 m}$, say, since F_2 (and thus $\tilde{F}_2$) is a sufficiently large poly$((\ln m)/\epsilon)$. This quantity also bounds the probability that j' occurs in B_r.

Now fix an $i \in I$. Over the blocks B_r in seq(B_i), the expected number of $j' \in B_r$ with $x_{j'}^2 < \epsilon^{12} F_2$ that are excited in B_r and for which $j' \in B_{r+1}$ is at most

$$O(\epsilon^{-4} \ln(1/\epsilon) \ln m) \cdot \epsilon^5 / \ln^2 m = O((\epsilon \ln(1/\epsilon))/ \ln m),$$

using that the number of excited items in seq(B_i) is $O(\epsilon^{-4} \ln(1/\epsilon) \ln m)$ because event $\mathcal{E}$ occurs, and that conditioned on j' being excited in B_i, the probability that $j' \in B_{i+1}$ is $O(\epsilon^5)$, since the events that $j' \in B_i$ and $j' \in B_{i+1}$ are negatively correlated (and independent of satisfying $h^q(j') = i$ for some $q \in [10 \ln(1/\epsilon)]$). Union bounding over all $i \in I$, the probability there exists an $i \in I$ and an r in seq(B_i) for which there is a j' with $x_{j'}^2 < \epsilon^{12} F_2$ for which j' is excited in B_r and $j' \in B_{r+1}$ is at most $O(\epsilon \ln^2(1/\epsilon))$.

Let $\mathcal{I}$ be the event that there exists an $i \in I$ and an r in seq(B_i) for which there is a j' with $x_{j'}^2 < \epsilon^{12} F_2$ for which j' is excited in B_r and j' occurs in B_{i+1}. Thus,

$$\Pr[\mathcal{I}] \geq 1 - O(\epsilon \ln^2(1/\epsilon)). \tag{60}$$

Given events $\mathcal{E}, \mathcal{F}, \mathcal{G}, \mathcal{H}$, and $\mathcal{I}$, for each $i \in I$ and each r in seq(B_i), we have that there is no item $j' \neq j$ for which j' occurs in B_{r+1} and j' is excited in B_r. This in particular implies there is no instance of Identify already running (and not yet completed) after processing B_{i+1} for each $i \in I$.

Event $\mathcal{J}$. It only remains to show that j occurs in one of the B_i for which $h^q(j) = i$ (i.e., the set I), and that j also occurs in B_{i+1}. Fix two consecutive blocks B_a and B_{a+1}. The probability that j occurs in both of these blocks is at least (1− probability j doesn't occur in at least one of the blocks):

$$\begin{aligned} 1 - 2 \cdot (1 - x_j/F_1)^{\frac{F_1}{\epsilon\sqrt{\tilde{F}_2}}} &\geq 1 - 2 \cdot (1 - \epsilon\sqrt{F_2}/F_1)^{\frac{F_1}{\epsilon\sqrt{\tilde{F}_2}}} \\ &\geq (1 - 2e^{-1/1.1}) \\ &\geq 0.19, \end{aligned}$$

where we used that $\tilde{F}_2 \leq 1.1 F_2$.

Now, with probability $1 - O(\epsilon^3)$, we have that for all $q \neq q'$, $|h^q(j) - h^{q'}(j)| \geq 1$, given our assumption that $F_1 \geq \text{poly}(\epsilon^{-1})$ for a sufficiently large polynomial. Given this, there are $10\ln(1/\epsilon)$ disjoint pairs $(B_{h^q(j)}, B_{h^q(j)+1})$ of blocks.

Note that under conditioning of j not occurring in both of particular r pairs of blocks, for any $0 \leq r < 10\ln(1/\epsilon)$, the probability that j occurs in another pair of block only increases. Let B' be the set of these r pairs of blocks and $\bar{B}'$ the rest of the blocks. Let x' be the number of occurrences of j in $\bar{B}'$. The note holds because the probability that j occurs consecutively in a pair of block in $\bar{B}'$ increases with increasing x' and the probability that j does not occur consecutively in any of the r pairs of blocks also increases with increasing x'.

Thus, the probability that j does not occur in both blocks in any of these $10\ln(1/\epsilon)$ pairs is at most

$$(1 - 0.19)^{10\ln(1/\epsilon)} \leq \epsilon^{2.1}.$$

Let $\mathcal{J}$ be the event that j occurs in one of the B_i for $i \in I$, and also in B_{i+1}. Thus,

$$\Pr[\mathcal{J}] \geq 1 - O(\epsilon^3) - \epsilon^{2.1}. \tag{61}$$

Putting It All Together. By (56), (57), (58), (59), (60), (61), and also union bounding over the event $\mathcal{K}$ that $|H| \leq 2\epsilon^{-2}$ at all times during the stream, which occurs with probability at least $1 - 1/m$, we have:

$$\Pr[\mathcal{E} \wedge \mathcal{F} \wedge \mathcal{G} \wedge \mathcal{H} \wedge \mathcal{I} \wedge \mathcal{J} \wedge \mathcal{K}] \geq 1 - \frac{\epsilon^2}{81} - \frac{\epsilon^2}{81} - O(\epsilon^3) - O(\epsilon^3) - O(\epsilon^3 \ln^2(1/\epsilon) - O(\epsilon^3) - \epsilon^{2.1} - \frac{1}{m} \geq 1 - \frac{\epsilon^2}{40},$$

provided ϵ is less than a sufficiently small constant.

If these events occur, then the three conclusions of the lemma hold for this particular $j \in HH$. Finally, since there are at most ϵ^{-2} different $j \in HH$, we can union bound over all of them to conclude that the lemma holds with probability at least $1 - \epsilon^{-2} \cdot \frac{\epsilon^2}{40} = \frac{39}{40}$, as desired. □

We conclude this subsubsection with the following theorem.

Theorem 30. *(*Main1 *is a Correct Algorithm) With probability at least* $9/10$, Main1$(F_2, m, a_1, \ldots, a_m)$ *outputs a set H which contains a pair $(j, \hat{x}_j)$ for every j for which $x_j^2 \geq \epsilon^2 F_2$, and no j for which $x_j^2 \leq \frac{\epsilon^2}{2} F_2$. Further, if $(j, \hat{x}_j)$ is in H, then $\hat{x}_j = (1 \pm \epsilon)x_j$.*

Proof. First, by Lemma 4, with probability $1 - 1/m$, we never return an element j for which $x_j^2 \leq \frac{\epsilon^2}{2} F_2$.

Given this, it follows by Lemma 6, with probability $1 - 1/m^{98}$, if $(j, \hat{x}_j)$ is in H, then $\hat{x}_j = (1 \pm \epsilon)x_j$.

Next, by Lemma 7, with probability at least $39/40$, simultaneously for all $j \in HH$, Identify will be invoked in line 14. By Lemma 5, with probability at least $1 - 1/m^{.05}$, every heavy hitter will occur in at least $4000\epsilon^{-2}(1 - 1.1/e)\ln m$ of the next blocks after Identify is invoked. By a union bound, both of these events occur with probability at least $39/40 - 1/m^{.05} \geq 38/39$, for large enough m.

Fix a $j \in HH$. It only remains to show in line 3 of Identify, there will be a block among the $100\ln(1/\epsilon)$ blocks after the i-th block, if $h^q(j) = i$ for some $q \in [10\ln(1/\epsilon)]$, for which j occurs in the block and there is no $j' \neq j$ in the block for which $g(j') = g(j)$ and $h^{q'}(j') = i$ for some $q' \in [10\ln(1/\epsilon)]$. Further, this holds for the first block for which there is a j' occurring in the block for which $g(j') = g(j)$ and $h^{q'}(j') = i$.

Notice that the total number of distinct stream items in these $100\ln(1/\epsilon)$ blocks is at most $|B_i| 100\ln(1/\epsilon) = \frac{100\ln(1/\epsilon)F_1}{\epsilon\sqrt{F_2}}$. For an item j', the probability that $h^{q'}(j') = i$ for some $q' \in [10\ln(1/\epsilon)]$ is at most $(10\ln(1/\epsilon))/t$, and so the expected number of items j' for which $h^{q'}(j') = i$ for some $q' \in [10\ln(1/\epsilon)]$ is

$$\frac{100\ln(1/\epsilon)F_1}{\epsilon\sqrt{F_2}} \cdot \frac{10\ln(1/\epsilon)}{t} \leq \frac{(1000\ln^2 1/\epsilon)F_1}{\epsilon^2 F_2} \leq 1000\epsilon^{-2}\ln^2(1/\epsilon).$$

If X is the number of items j' for which $h^{q'}(j') = i$ for some $q' \in [10\ln(1/\epsilon)]$, then $\mathbf{E}[X] \leq 1000\epsilon^{-2}\ln^2(1/\epsilon)$, and since h is drawn from a pairwise independent family, $\mathbf{Var}[X] \leq 1000\epsilon^{-2}\ln^2(1/\epsilon)$. So by Chebyshev's inequality, $X = O(\epsilon^{-2}\ln^2(1/\epsilon))$ with probability $1 - O(\epsilon^2/\ln^2(1/\epsilon))$. Since g is independent of the h^q and drawn from a pairwise independent family with range size $\epsilon^{-13}\ln^2 m$, all of these items are perfectly hashed under g with probability larger than $1 - \epsilon^{-3} - O(\epsilon^2/\ln^2(1/\epsilon))$. Given this, in order for line 3 to succeed, it follows that we just need that for one of the $100\ln(1/\epsilon)$ blocks processed in line 3, the heavy hitter j occurs. Since conditioned on not occurring in one block, j is only more likely to occur in another, the probability it does not occur in all $100\ln(1/\epsilon)$ iterations is at most

$$(1 - x_j/F_1)^{\frac{100 F_1 \ln(1/\epsilon)}{\epsilon\sqrt{F_2}}} \leq e^{-100\ln(1/\epsilon)} = \epsilon^{100}.$$

Hence, for a given heavy hitter j, line 3 succeeds with probability at least $1 - \epsilon^3 - O(\epsilon^2/\ln^2(1/\epsilon)) - \epsilon^{100} \geq 1 - O(\epsilon^2/\ln^2(1/\epsilon))$. By a union bound, this holds simultaneously for every heavy hitter j with probability at least $1 - O(1/\ln^2(1/\epsilon))$.

The theorem thus follows, with error probability at most $1/m + 1/m^{98} + 1/39 + O(1/\ln^2(1/\epsilon)) \leq 1/10$. □

The Main **Algorithm.** We now show the correctness of our overall algorithm $\textsc{Main}(a_1, \ldots, a_m)$.

Theorem 31. *(*Main *is a Correct Algorithm) With probability at least* $4/5$, $\textsc{Main}(a_1, \ldots, a_m)$ *outputs a set* H *which contains every* j *for which* $x_j^2 \geq \epsilon^2 F_2$ *and no* j *for which* $x_j^2 \leq \frac{\epsilon^2}{2} F_2$.

Proof. Let i be the largest integer for which $\textsc{Main1}^{i-1}$ has completed. Note that i is a deterministic function of m. Note that H^{i-1} was the output of $\textsc{Main1}^{i-1}$.

Notice that $\textsc{Main1}^{i-1}$ starts at stream position 2^{i-1} and is run on a stream of 2^{i-1} updates. The algorithm $\textsc{Main1}^{i-1}$ is given $1.01 \cdot \tilde{F}_2^{(k)}$ as input, where $k = 2^{i-1} - 1$.

A Concentrated Second Moment Estimate. Let $\mathcal{A}$ be the event that the argument $1.01 \cdot \tilde{F}_2^{(k)}$ given to $\textsc{Main1}^{i-1}$ satisfies

$$F_2^{[k+1,k+2^{i-1}]} \leq 1.01 \cdot \tilde{F}_2^{(k)} \leq 1.1 F_2^{[k+1,k+2^{i-1}]}, \tag{62}$$

where $F_2^{[k+1,k+2^{i-1}]} = \sum_{j\in[n]}(x_j^{i-1})^2$, where x_j^{i-1} represents the number of occurrences of item j from $k+1$ to $k + 2^{i-1}$. We first bound $\Pr[\mathcal{A}]$.

The intervals $[1,k]$ and $[k+2, k+2^{i-1}]$ have the same length and $F_2^{[1,k]}$ and $F_2^{[k+2,k+2^{i-1}]}$ are identically distributed random variables on $2^{i-1} - 1$ updates. Let us call an instance of this random variable X. We first show that X is concentrated around its expectation. Let X_j be the number of occurrences of item j in an interval of length $2^{i-1} - 1$. Let $X = \sum_j X_j^2$.

Note that X_j is hypergeometrically distributed with expectation $x_j \cdot \frac{2^{i-1}-1}{m}$. Note that $\mathbf{E}[X_j] = \frac{2^{i-1}-1}{m} x_j$, and by our choice of i, this is $\Theta(x_j)$. Say an index j is large if $\mathbf{E}[X_j] = \omega(\ln m)$.

We apply Fact 2 to a large j to conclude,

$$\begin{aligned} \Pr[X_j = (1 \pm \gamma)\mathbf{E}[X_j]] &\geq 1 - 2 \cdot e^{-2\gamma^2 \mathbf{E}[X_j]} \\ &\geq 1 - \frac{1}{m^2}, \end{aligned}$$

where the m^2 is arbitrary and $\gamma > 0$ is an arbitrarily small constant. Thus, by a union bound over the items j that occur at least once in the stream, of which there can be at most m items, we have that with probability $1 - 1/m$, simultaneously for all j for which $x_j = \omega(\ln m)$, we have $X_j = (1 \pm \gamma) \cdot \frac{2^{i-1}-1}{m} x_j$, and thus

$$X_j^2 = (1 \pm 3\gamma) \cdot \left(\frac{2^{i-1}-1}{m}\right)^2 x_j^2. \tag{63}$$

Let $\mathcal{B}$ be the event that simultaneously for all large j, we have that (63) holds. Then,

$$\Pr[\mathcal{B}] \geq 1 - \frac{1}{m}.$$

Let us refer to all non-large indices j as *small* items. Let $Y = \sum_{\text{small } j} X_j^2$. The X_j^2 are negatively correlated, and consequently

$$\mathbf{Var}[Y] \leq \sum_{\text{small } j} \mathbf{Var}[X_j^2] \leq \sum_{\text{small } j} \mathbf{E}[X_j^4] = O(m \ln^3 m),$$

again using that we only need to sum over small j that occur at least once in the stream, and there can be at most m such items.

Thus, by Chebyshev's inequality,

$$\Pr[|Y - \mathbf{E}[Y]| \geq \sqrt{m} \ln^2 m] = o(1).$$

Let $\mathcal{C}$ be the event that $|Y - \mathbf{E}[Y]| < \sqrt{m} \ln^2 m$. Then, $\Pr[\mathcal{C}] = 1 - o(1)$.

Write $X = \sum_{\text{large } j} X_j^2 + Y$. By a union bound over events $\mathcal{B}$ and $\mathcal{C}$, we have that with probability at least $1 - 1/m - o(1)$,

$$X = (1 \pm 3\gamma)\mathbf{E}[\sum_j X_j^2] \pm \sqrt{m} \ln^2 m.$$

Note that $\mathbf{E}[\sum_j X_j^2] \geq \mathbf{E}[\sum_j X_j] = \Omega(m)$, since $2^{i-1} - 1 = \Omega(m)$, and $\mathbf{E}[\sum_j X_j] = 2^{i-1} - 1$. It follows that with probability $1 - 1/m - o(1)$,

$$X = (1 \pm 4\gamma)\mathbf{E}[\sum_j X_j^2].$$

Now let $Y = \sum_j Y_j^2$ where Y_j is the number of occurrences of item j in an interval of length 2^{i-1}. Then $Y \in [X, X + 2\sqrt{X} + 1]$. It follows from $\mathbf{E}[X] = \Omega(m)$ that with probability $1 - 1/m - o(1)$,

$$Y = (1 \pm 4\gamma)\mathbf{E}[\sum_j X_j^2].$$

Note that Y and $F_2^{[k+1,k+2^{i-1}]}$ are identically distributed. Thus, with probability $1 - 1/m - o(1)$, we have

$$F_2^{[k+1,k+2^{i-1}]} = (1 \pm 4\gamma)\mathbf{E}[\sum_j X_j^2]. \tag{64}$$

By standard guarantees of the F_2-sketch [3], with probability 99/100,

$$1.001 \cdot F_2^{[1,k]} \leq 1.01\tilde{F}_2^{(k)} \leq 1.011 F_2^{[1,k]}, \tag{65}$$

and with probability $1 - 1/m - o(1)$,

$$F_2^{[1,k]} = (1 \pm 4\gamma)\mathbf{E}[\sum_j X_j^2]. \tag{66}$$

Putting (64), (65), and (66) together, for a sufficiently small constant $\gamma > 0$, we have the condition of event $\mathcal{A}$. It follows that $\Pr[\mathcal{A}] \geq 99/100 - o(1)$.

Showing that $\textsc{Main1}^{i-1}$ **succeeds.** Let $\mathcal{E}$ be the event that $\textsc{Main1}^{i-1}$ succeeds with parameter $\frac{\epsilon}{10}$, namely, $\textsc{Main1}^{i-1}$ outputs a set H^{i-1} which contains every j for which $(x_j^{i-1})^2 \geq \frac{\epsilon^2}{100} \cdot F_2^{[k+1,k+2^{i-1}]}$ and no j for which $(x_j^{i-1})^2 \leq \frac{\epsilon^2}{200} F_2^{[k+1,k+2^{i-1}]}$. Moreover, for every $(j, \hat{x}_j)$ in H^{i-1}, we have $\hat{x}_j = (1 \pm \epsilon/10)x_j^{i-1}$.

This amounts to understanding when event $\mathcal{A}$ occurs, and how its occurrence conditions the underlying random permutation.

We first bound $\Pr[\mathcal{E} \mid \mathcal{A}]$. Note that conditioning on $\mathcal{A}$ may also bias the random order Π of the stream, since although the F_2-sketch has its own private randomness and succeeds on every stream with probability at least $99/100$, it may succeed a little bit more on certain streams than on others.

Note that for any two stream orderings π, π', we can apply Bayes rule to conclude:

$$\begin{aligned}
\frac{\Pr[\Pi = \pi \mid \mathcal{A}]}{\Pr[\Pi = \pi' \mid \mathcal{A}]} &= \frac{\Pr[\Pi = \pi \mid \mathcal{A}]\Pr[\mathcal{A}]}{\Pr[\Pi = \pi' \mid \mathcal{A}]\Pr[\mathcal{A}]} \\
&= \frac{\Pr[\mathcal{A} \mid \Pi = pi]\Pr[\Pi = pi]}{\Pr[\mathcal{A} \mid \Pi = \pi']\Pr[\Pi = \pi']} \\
&\geq \frac{\frac{99}{100}}{1} \\
&= \frac{99}{100}. \qquad (67)
\end{aligned}$$

By Theorem 30, for any argument $1.01 \cdot \tilde{F}_2^{(k)}$ given to $\textsc{Main1}^{i-1}$, if it satisfies (62), then the probability that $\textsc{Main1}^{i-1}$ succeeds, over the uniform distribution Π, is at least $9/10$. Conditioned on $\mathcal{A}$, we have that the argument $1.01 \cdot \tilde{F}_2^{(k)}$ given to $\textsc{Main1}^{i-1}$ satisfies (62), and (67) implies that the distribution of Π conditioned on $\mathcal{A}$ has total variation distance at most $1/100$ from the uniform distribution on Π. Hence, $\Pr[\mathcal{E} \mid \mathcal{A}] \geq 9/10 - 1/100$.

Consequently, using our earlier bound on $\Pr[\mathcal{A}]$,

$$\Pr[\mathcal{E}] \geq \Pr[\mathcal{E} \wedge \mathcal{A}] = \Pr[\mathcal{E} \mid \mathcal{A}] \cdot \Pr[\mathcal{A}] \geq \left(\frac{9}{10} - \frac{1}{100}\right) \cdot \left(\frac{99}{100} - o(1)\right) \geq \frac{17}{20}.$$

Note that with probability $99/100$, we have

$$\sqrt{F}_2 \leq \sqrt{\tilde{F}_2^{(m)}} \leq 1.01\sqrt{F}_2.$$

Let $\mathcal{F}$ be the event that: $\sqrt{F}_2 \leq \sqrt{\tilde{F}_2^{(m)}} \leq 1.01\sqrt{F}_2$.

$\textsc{Main}$ **is a correct algorithm.** If $\mathcal{E}$ occurs, then $\textsc{Main1}^{i-1}$ succeeds, that is, $\textsc{Main1}^{i-1}$ succeeds with parameter $\frac{\epsilon}{10}$, namely, $\textsc{Main1}^{i-1}$ outputs a set H^{i-1} which contains every j for which $(x_j^{i-1})^2 \geq \frac{\epsilon^2}{100} \cdot F_2^{[k+1,k+2^{i-1}]}$ and no j for which

$$(x_j^{i-1})^2 \leq \frac{\epsilon^2}{200} F_2^{[k+1,k+2^{i-1}]}. \qquad (68)$$

Further, if $(j, \hat{x}_j)$ is in H^{i-1}, then $\hat{x}_j = (1 \pm \epsilon/10)x_j^{i-1}$.

Let H' be the set of j such that $x_j \geq \frac{\epsilon}{40}\sqrt{m}$ $(HH \subseteq H')$. Next, we calculate the probability that $\frac{m}{2^{i-1}} \cdot x_j^{i-1} = (1 \pm \epsilon/10)x_j$. Using Fact 2, this probability is at least $1 - 2e^{-\frac{\epsilon^2}{300}\frac{x_j}{4}} \geq 1 - \frac{\epsilon^4}{m^2}$, for a fixed $j \in H'$. Using union bound, $\frac{m}{2^{i-1}} \cdot x_j^{i-1} = (1 \pm \epsilon/10)x_j$, for all $j \in H'$ with probability at least $1 - \epsilon^2$. We call this event $\mathcal{G}$. As $2^{i-1} \geq m/4$, $F_2^{[k+1,k+2^{i-1}]} \geq m/4$.

$$\frac{\epsilon}{10\sqrt{2}}\sqrt{F_2^{[k+1,k+2^{i-1}]}} > \frac{\epsilon}{40}\sqrt{m},$$

and therefore under events $\mathcal{E}$ and $\mathcal{G}$, $H^{i-1} \subseteq H'$. Thus, under $\mathcal{G}$ and $\mathcal{E}$, for all $j \in H^{i-1}$,

$$\frac{m}{2^{i-1}} \cdot \hat{x}_j = \left(1 \pm \frac{\epsilon}{5}\right) x_j.$$

Showing all $j \in HH$ are output by Main**:** We first explain why event $\mathcal{E}$, together with the event $\mathcal{F}$ and $\mathcal{G}$ occurring, implies for all $j \in HH$, we have $j \in H^{i-1}$. This is because (as $2^{i-1} \geq m/4$):

$$x_j^{i-1} \geq \frac{2^{i-1}}{m} \cdot (1 - \epsilon/10) \cdot x_j \geq \frac{2^{i-1}}{m} \cdot (1 - \epsilon/10) \cdot \epsilon\sqrt{F_2} \geq \frac{\epsilon}{4} \cdot (1 - \epsilon/10)\sqrt{F_2^{[k+1,k+2^{i-1}]}}$$

. Consequently, $j \in H^{i-1}$. Notice that this holds for all $j \in HH$. Because of events $\mathcal{E}$, $\mathcal{G}$ and $\mathcal{F}$ occurring, we have that for a $j \in HH$ that

$$\frac{m}{2^{i-1}} \cdot \hat{x}_j \geq (1 - \epsilon/5)x_j \geq (1 - \epsilon/5)\epsilon\sqrt{F_2} \geq (1 - \epsilon/5)\epsilon\frac{\sqrt{\tilde{F}_2^{(m)}}}{1.01}. \tag{69}$$

Therefore, every $j \in HH$ is returned in Line 8 of Main.

Showing no j for which $x_j^2 \leq \frac{\epsilon^2}{2}F_2$ is output by Main**:** It remains to show there is no j for which $x_j^2 \leq \frac{\epsilon^2}{2}F_2$ is output by Main. In this case Line 8 of Main will check if $\frac{m}{2^{i-1}}\hat{x}_j \geq (1 - \epsilon/5)\epsilon\frac{\sqrt{\tilde{F}_2^{(m)}}}{1.01}$.

Since $\tilde{F}_2^{(m)} \geq \sqrt{F_2}$ conditioned on $\mathcal{F}$ occurring, this holds only if $\frac{m}{2^{i-1}}\hat{x}_j \geq (1 - \epsilon/5)\epsilon\frac{\sqrt{F_2}}{1.01}$, but because event $\mathcal{E}$ and $\mathcal{G}$ holds (and $j \in H^{i-1}$), this implies

$$x_j \geq \frac{1 - \epsilon/5}{1 + \epsilon/5}\epsilon\frac{\sqrt{F}_2}{1.01}.$$

The latter cannot hold, because for ϵ less than a sufficiently small constant, it would contradict that $x_j^2 \leq \frac{\epsilon^2}{2}F_2$.

The overall error probability is $\frac{3}{20} + \epsilon^2 + \frac{1}{100}$, to union bound over either $\mathcal{E}$ or $\mathcal{F}$ or $\mathcal{G}$ not occurring. This error probability is at most 1/5, which completes the proof of the theorem. □

ℓ_2-Point Query. In the ℓ_2-Point Query Problem, at the end of the stream we are given an index j and asked to output an estimate $\hat{x}_j$ such that $\hat{x}_j = x_j \pm \epsilon\sqrt{F_2}$ with probability at least 4/5. As an immediate corollary of our algorithm for ℓ_2-Heavy Hitters we obtain an algorithm for ℓ_2-Point Query. Indeed, if S is the set output by Main, if S does not contain the query j, then we can simply output 0. If Main succeeds, then if $j \notin S$, then $x_j \leq \epsilon\sqrt{F_2}$, and so $\hat{x}_j = 0$ is a correct answer. On the other hand, if $j \in S$, then the estimate $\hat{x}_j$ returned by Main is guaranteed to satisfy $\hat{x}_j = (1 \pm \epsilon)x_j = x_j \pm \epsilon x_j = x_j \pm \epsilon\sqrt{F_2}$, using that $x_j \leq \sqrt{F_2}$ for all j.

The total memory required is as given by Theorem 28, namely, $O(\epsilon^{-2}(\ln d + \ln^2(1/\epsilon) + \ln^2 lnm) + \ln m)$. However, unlike for the ℓ_2-Heavy Hitters Problem, in the ℓ_2-Point Query Problem, the dependence on $\ln d$ is unnecessary. Here we give a simple way of removing this term from the algorithm's space complexity.

Corollary 32. *The ℓ_2-Point Query Problem can be solved in $O(\epsilon^{-2}(\ln^2(1/\epsilon) + \ln^2 \ln m) + \ln m + (\ln(1/\epsilon) \ln d))$ bits of memory.*

Proof. The only modification we make to our algorithm for the ℓ_2-Heavy Hitters Problem is in line 14 of Main1, where instead of adding a non-empty output $(id, \hat{x}_{id})$ to our output set H, we choose a pairwise independent hash function $\tau : [d] \to [200/\epsilon^2]$, and instead add $(\tau(id), \hat{x}_{id})$ to our output set H. From the proof of Theorem 28 that this reduces the memory to $O(\epsilon^{-2}(\ln^2(1/\epsilon + \ln^2 \ln m)) + \ln m)$, up to the memory required for storing the hash functions. The hash functions can be stored using $O(\ln(1/\epsilon) \ln d)$ bits of memory. See also the discussion above Theorem 28 on handling the case $m \leq \text{poly}(\epsilon^{-2} \ln d)$, in which case we also achieve this memory bound.

When given an index j at the end of the stream, we can compute $\tau(j)$ and check if there is a pair of the form $(\tau(j), v)$ for some value v in the output H of Main. If so, we output v, otherwise we output 0. Note that if $j \notin HH$, then the probability $\tau(j) \in \{\tau(j') \mid (j', v') \in H \text{ for some } v'\}$ is at most $\frac{1}{100}$, since $|H| \leq \frac{2}{\epsilon^2}$. Thus, with probability at least $\frac{99}{100}$, we output 0, which is a correct answer if $j \notin HH$. Otherwise, if $j \in HH$, with probability $\frac{99}{100}$, conditioned on Main succeeding, there will be a unique pair $(\tau(j), v) \in H$ with first coordinate equal to $\tau(j)$. In this case, we again succeed given that Main succeeds. The overall success probability is therefore at least $4/5 - 1/100 \geq 2/3$, union bounding over the failure probability of Main and collisions under the hash function τ. □

5.2 ℓ_∞-Estimation in Turnstile Streams

We consider the problem of estimating $\|x\|_\infty$ up to additive error $\frac{1}{\sqrt{k}}\|x\|_2$. The best upper bound for this problem in the turnstile model involves finding the so-called ℓ_2-heavy hitters, and has space $O(k(\log d) \log m)$. We observe that we can reduce this to $O(k(\log k + \log\log m)\log(dm))$ bits in the turnstile model.

Theorem 33. *($\|x\|_\infty$-Approximation) There is a turnstile streaming algorithm which approximates $\|x\|_\infty$ up to additive error $\frac{1}{\sqrt{k}}\|x\|_2$ with probability at least $2/3$ and using $O(k(\log(dm))(\log k + \log\log m))$ bits of memory.*

Proof. The proof is inspired from a universe reduction technique of [23], which shows for $0 < p < 2$, there is a randomized oblivious linear map which takes a vector x and produces a vector y so that with good probability, if i is a so-called ℓ_p-heavy hitter of x, then $y_{h(i)}$ is an ℓ_p-heavy hitter of y, where h corresponds to a hash function defining y. Here, we need to argue this holds for $p = 2$, which in a certain sense simplifies the analysis of [23], but on the other hand, we also need to argue that there are no spurious heavy hitters, i.e., entries of y that are heavy but do not correspond to any heavy hitter in x, which was not needed in [23].

Let $N = \text{poly}(k \log d)$ be a sufficiently large polynomial. Let $h : [d] \to [N]$ be drawn from a pairwise independent family, and let $\sigma : [d] \to \{-1, 1\}$ be drawn from an $O(\log k + \log\log d)$-wise independent family. Let $y_j = \sum_{i | h(i) = j} \sigma(i) x_i$.

The event $\mathcal{E}$. We call an item i for which $x_i^2 \geq \frac{1}{C(C')^2 k^2 \log^2 N}\|x\|_2^2$ a *large* item, otherwise it is *small*. Here $C, C' > 0$ are sufficiently large constants. The choice for these constants will soon become apparent. For N large enough, we have the event $\mathcal{E}$ that every large item i goes to its own separate hash bucket. There are only $O(k^2 \log^2 N)$ large items and so if $N = \Omega(k^4 \log^4 N)$, then event $\mathcal{E}$ happens with probability at least 99/100. We condition on $\mathcal{E}$ in what follows.

The event $\mathcal{F}$. Let $i_1, \ldots, i_r$ be the large items and let $j_1, \ldots, j_r$ be the coordinates of y containing a large item. Let $\mathcal{F}$ be the event that for each coordinate j of y:

1. if there is no large item i with $h(i) = j$, then $y_j^2 \leq \frac{8}{C'k}\|x\|_2^2$,

2. if there is a single large item i_ℓ with $h(i_\ell) = j_\ell$, then $|y_{j_\ell} - \sigma(i_\ell) x_{i_\ell}|^2 \leq \frac{8}{C'k}\|x\|_2^2$.

To bound $\Pr[\mathcal{F}]$, let $Z_j = \sum_{small\ i} \delta(h(i) = j) x_i^2$. Then $\mathbf{E}[Z_j] = \frac{\|x\|_2^2}{N}$. Also,

$$\begin{aligned}
\mathbf{Var}[Z_j^2] &= \sum_{small\ i} \mathbf{E}[\delta(h(i) = j)^2] x_i^4 + \sum_{small\ i \neq i'} \mathbf{E}[\delta(h(i) = j)\delta(h(i') = j)] x_i^2 x_{i'}^2 - \mathbf{E}^2[Z_j] \\
&\leq \frac{1}{N} \sum_{small\ i} x_i^4 \\
&\leq \frac{1}{N} \max_{small\ i} x_i^2 \sum_{small\ i} x_i^2 \\
&\leq \frac{1}{NC(C')^2 k^2 \log^2 N}\|x\|_2^4
\end{aligned}$$

By Chebyshev's inequality,

$$\Pr[Z_j \geq \frac{1}{C'k \log N} \|x\|_2^2] \leq \frac{(C')^2 k^2 (\log^2 N) \|x\|_2^4}{C(C')^2 N k^2 (\log^2 N) \|x\|_2^4} \leq \frac{1}{CN}.$$

By a union bound this holds simultaneously for all N values of j with probability at least $1 - 1/C$. We call this event $\mathcal{F}_1$ and condition on this in what follows.

We now use the following lemma:

Lemma 8. *(Lemma 17 of [23]) For $x \in \mathbb{R}^n$, $\lambda > 0$, with λ^2 a multiple of* 8*, and random $z \in \{-1,1\}^n$ drawn from a $(\lambda^2/4)$-wise independent family,* $\Pr[|\langle x, z\rangle|^2 > \lambda^2 \|x\|_2^2] < 2^{-\lambda^2/4}$.

Setting $\lambda^2 = 8 \log N$ in Lemma 8, it follows that with probability $1 - 1/N$, simultaneously for all $j \in [N]$,

1. if y_j does not contain a large item, then

$$y_j^2 \leq Z_j \cdot 8 \log N \leq \frac{8 \log N}{C'k \log N} \|x\|_2^2 = \frac{8}{C'k} \|x\|_2^2.$$

2. if y_j does contain a large item j_ℓ, then

$$(y_{j_\ell} - \sigma(i_\ell) x_{i_\ell})^2 \leq Z_j \cdot 8 \log N \leq \frac{8 \log N}{C'k \log N} \|x\|_2^2 \leq \frac{8}{C'k} \|x\|_2^2.$$

It follows that $\Pr[\mathcal{F}] \geq 1 - 1/C - 1/N \geq 99/100$, by choosing $C > 0$ to be a sufficiently large constant.

The Event $\mathcal{G}$. Note that $\mathbf{E}[\|y\|_2^2] = \|x\|_2^2$, and since y can be viewed as the image of a CountSketch map (with more independence than required on the hash functions h and σ), we have that $\Pr[\|y\|_2^2 \geq (1+\epsilon)\|x\|_2^2] \leq 1/(\epsilon^2 N) \leq 1/100$, by choosing N to be sufficiently large (see, e.g., [34] for background on CountSketch for estimating the 2-norm).

The Event $\mathcal{H}$. We apply the CountSketch data structure to y. Note that this is a composition of oblivious linear maps on x, and each map can be specified with limited independent hash functions (we detail the space usage below) so can be maintained in a stream. Since y is only N-dimensional, we have the guarantee that CountSketch approximates each y_j up to additive error $\frac{1}{100\sqrt{k}} \|y|\|_2 \leq \frac{1}{99\sqrt{k}} \|x\|_2$ using $O(k (\log N) \log m) = O(k(\log \log d + \log k) \log m)$ bits of memory (see, e.g., Section 1.3.2 of [16]; the claimed space bound follows by setting $\delta = 1/\operatorname{poly}(d)$ and applying a union bound over all coordinates of y), and failure probability $1/N$, which can be made at most $1/100$. Here we use the event $\mathcal{G}$.

Putting it all together. Note that events $\mathcal{E}, \mathcal{F}, \mathcal{G}$, and $\mathcal{H}$ all simultaneously occur with probability at least $1 - 4/100$, which we condition on.

Let z_j be the estimate CountSketch gives to y_j. Then

$$|z_j - y_j| \leq \frac{1}{99\sqrt{k}} \|x\|_2,$$

simultaneously for all j.

We also have that $y_j^2 \leq \frac{8}{C'k} \|x\|_2^2$ if there is no large item i with $h(i) = j$. By setting $C' \geq 800$, we have $|y_j| \leq \frac{1}{10\sqrt{k}} \|x\|_2$. Consequently, $|z_j| \leq \frac{1}{2\sqrt{k}} \|x\|_2$.

On the other hand if there is a large item i with $h(i) = j$, then $|y_j| = |x_i| \pm \frac{1}{10\sqrt{k}} \|x\|_2$ by choosing $C' \geq 800$. Consequently, $|z_j| = |x_i| \pm \frac{1}{2\sqrt{k}} \|x\|_2$.

It follows that $\|z\|_\infty = \|x\|_\infty \pm \frac{1}{\sqrt{k}} \|x\|_2$ with probability at least $24/25$, so we can use $\|z\|_\infty$ as our estimator to solve this problem.

For the total memory, we use $O(k(\log \log d + \log k) \log m)$ bits of memory to store the CountSketch data structure, $O(\log d)$ bits to store h, and $O(\log(d)(\log \log d + \log k))$ bits to store σ. □

References

[1] José A Adell, Alberto Lekuona, and Yaming Yu. Sharp bounds on the entropy of the Poisson law and related quantities. *IEEE transactions on information theory*, 56(5):2299–2306, 2010.

[2] Noga Alon, Yossi Matias, and Mario Szegedy. The space complexity of approximating the frequency moments. *J. Comput. Syst. Sci.*, 58(1):137–147, 1999.

[3] Noga Alon, Yossi Matias, and Mario Szegedy. The space complexity of approximating the frequency moments. *J. Comput. Syst. Sci.*, 58(1):137–147, 1999.

[4] Ziv Bar-Yossef, Thathachar S Jayram, Ravi Kumar, and D Sivakumar. An information statistics approach to data stream and communication complexity. *Journal of Computer and System Sciences*, 68(4):702–732, 2004.

[5] Daniel Berend and Aryeh Kontorovich. A sharp estimate of the binomial mean absolute deviation with applications. *Statistics & Probability Letters*, 83(4):1254–1259, 2013.

[6] Vladimir Braverman, Stephen R. Chestnut, Nikita Ivkin, Jelani Nelson, Zhengyu Wang, and David P. Woodruff. BPTree: An L2 heavy hitters algorithm using constant memory. In *Proceedings of the 36th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2017, Chicago, IL, USA, May 14-19, 2017*, pages 361–376, 2017.

[7] Vladimir Braverman, Stephen R Chestnut, Nikita Ivkin, and David P Woodruff. Beating countsketch for heavy hitters in insertion streams. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, pages 740–753. ACM, 2016.

[8] Joshua Brody and Elad Verbin. The coin problem and pseudorandomness for branching programs. In *51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010, October 23-26, 2010, Las Vegas, Nevada, USA*, pages 30–39, 2010.

[9] Larry Carter and Mark N. Wegman. Universal classes of hash functions. *J. Comput. Syst. Sci.*, 18(2):143–154, 1979.

[10] Amit Chakrabarti, Graham Cormode, and Andrew McGregor. Robust lower bounds for communication and stream computation. *Theory Comput.*, 12(1):1–35, 2016.

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

[12] Arkadev Chattopadhyay, Jaikumar Radhakrishnan, and Atri Rudra. Topology matters in communication. In *2014 IEEE 55th Annual Symposium on Foundations of Computer Science*, pages 631–640. IEEE, 2014.

[13] Arkadev Chattopadhyay and Atri Rudra. The range of topological effects on communication. In *International Colloquium on Automata, Languages, and Programming*, pages 540–551. Springer, 2015.

[14] Gil Cohen, Anat Ganor, and Ran Raz. Two sides of the coin problem. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2014)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2014.

[15] Graham Cormode. Sketch techniques for approximate query processing.

[16] Graham Cormode. Sketch techniques for approximate query processing. *Foundations and Trends in Databases. NOW publishers*, 2011.

[17] Graham Cormode. Sketch techniques for massive data. In Graham Cormode, Minos Garofalakis, Peter J. Haas, and Chris Jermaine, editors, *Synopses for Massive Data: Samples, Histograms, Wavelets, Sketches*, volume 4 of *Foundations and Trends in Databases*, pages 1–294. Now Publishers Inc., Hanover, MA, USA, January 2012.

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

[19] Graham Cormode and Shan Muthukrishnan. What's new: Finding significant differences in network data streams. *IEEE/ACM Transactions on Networking (TON)*, 13(6):1219–1232, 2005.

[20] Svante Janson. Large deviation inequalities for sums of indicator variables. *arXiv preprint arXiv:1609.00533*, 2016.

[21] Rajesh Jayaram and David P. Woodruff. Data streams with bounded deletions. In *Proceedings of the 37th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, Houston, TX, USA, June 10-15, 2018*, pages 341–354, 2018.

[22] Rajesh Jayaram and David P. Woodruff. Towards optimal moment estimation in streaming and distributed models. *CoRR*, abs/1907.05816, 2019.

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

[24] Daniel M Kane, Jelani Nelson, and David P Woodruff. On the exact space complexity of sketching and streaming small norms. In *Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete Algorithms*, pages 1161–1178. Society for Industrial and Applied Mathematics, 2010.

[25] Chin Ho Lee and Emanuele Viola. The coin problem for product tests. *TOCT*, 10(3):14:1–14:10, 2018.

[26] Ahmed Metwally, Divyakant Agrawal, and Amr El Abbadi. Efficient computation of frequent and top-k elements in data streams. In *International Conference on Database Theory*, pages 398–412. Springer, 2005.

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

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

[29] Peter Mörters and Yuval Peres. *Brownian motion*, volume 30. Cambridge University Press, 2010.

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

[31] Jelani Nelson. Sketching and streaming algorithms for processing massive data. *XRDS: Crossroads, The ACM Magazine for Students*, 19(1):14–19, 2012.

[32] Herbert Robbins. A remark on Stirling's formula. *The American mathematical monthly*, 62(1):26–29, 1955.

[33] John P. Steinberger. The distinguishability of product distributions by read-once branching programs. In *Proceedings of the 28th Conference on Computational Complexity, CCC 2013, K.lo Alto, California, USA, 5-7 June, 2013*, pages 248–254, 2013.

[34] Mikkel Thorup and Yin Zhang. Tabulation-based 5-independent hashing with applications to linear probing and second moment estimation. *SIAM J. Comput.*, 41(2):293–331, 2012.

A Absolute Deviation from Mean for Binomial Random Variable

In this section, we prove that

$$\mathbb{E}_{x\in\{0,1\}^n}\left|\sum_{i=1}^{n} x_i - n/2\right| = \frac{1}{2^n}\cdot n\cdot\binom{n-1}{\lfloor n/2\rfloor}$$

using the following calculations on binomial coefficients.

$$\begin{aligned}\sum_{x\in\{0,1\}^n}\left|\sum_{i=1}^{n} x_i - n/2\right| &= 2\cdot\sum_{i=0}^{\lfloor n/2\rfloor}\frac{n!}{(n-i)!i!}\cdot(n/2-i)\\ &= n\cdot\sum_{i=0}^{\lfloor n/2\rfloor}\frac{n!}{(n-i)!i!} - 2\cdot\sum_{i=0}^{\lfloor n/2\rfloor}\frac{n!}{(n-i)!i!}\cdot i\\ &= n\cdot\sum_{i=0}^{\lfloor n/2\rfloor}\binom{n}{i} - 2n\cdot\sum_{j=0}^{\lfloor n/2\rfloor-1}\binom{n-1}{j}.\end{aligned}$$

When n is odd the above expression is equivalent to

$$n\cdot 2^{n-1} - 2n\cdot\frac{1}{2}\cdot\left(2^{n-1}-\binom{n-1}{(n-1)/2}\right) = n\cdot\binom{n-1}{(n-1)/2} = n\cdot\binom{n-1}{\lfloor n/2\rfloor},$$

and when n is even, it is equivalent to

$$n\cdot\frac{1}{2}\cdot\left(2^n+\binom{n}{n/2}\right) - n\cdot 2^{n-1} = \frac{n}{2}\binom{n}{n/2} = n\cdot\binom{n-1}{\lfloor n/2\rfloor}.$$