

Tight Bounds for Heavy-Hitters and Moment Estimation in the Sliding Window Model *

Shiyuan Feng
EECS, Peking University
fsyo@stu.pku.edu.cn

William Swartworth
Carnegie Mellon University
wswartwo@andrew.cmu.edu

David P. Woodruff
Carnegie Mellon University
dwoodruf@cs.cmu.edu

Abstract

We consider the heavy-hitters and F_p moment estimation problems in the sliding window model. For F_p moment estimation with $1 < p \leq 2$, we show that it is possible to give a $(1 \pm \epsilon)$ multiplicative approximation to the F_p moment with $2/3$ probability on any given window of size n using $\tilde{O}(\frac{1}{\epsilon^p} \log^2 n + \frac{1}{\epsilon^2} \log n)$ bits of space. We complement this result with a lower bound showing that our algorithm gives tight bounds up to factors of $\log \log n$ and $\log \frac{1}{\epsilon}$. As a consequence of our F_2 moment estimation algorithm, we show that the heavy-hitters problem can be solved on an arbitrary window using $O(\frac{1}{\epsilon^2} \log^2 n)$ space which is tight.

1 Introduction

In some situations, such as monitoring network traffic, data is being received at an enormous rate and it is not always worthwhile to store the entire dataset. Instead we may only be interested in some statistics of the data, which we can hope to estimate from a data structure that uses much less space. This motivated the development of streaming algorithms, which process a sequence of updates one at a time, while maintaining some small sketch of the underlying data.

In some applications, it makes sense to focus on only the most recent updates. For instance, when tracking IP addresses over a network [SW02, DLOM02], we might be more interested in tracking the heavy users on a week-to-week basis rather than over longer time periods. Even if we are interested in longer time periods, we may still be interested in obtaining more fine-grained information about how the heavy users change over time.

This type of problem motivated the *sliding window model* for streaming algorithms which has a considerable line of work [DGIM02, BO07a, BGL⁺18, WZ22]. There are several versions of the sliding window model, but for us our main goal is to respond to queries about the previous n elements of the stream.

We consider perhaps the two most foundational problems in streaming algorithms: moment estimation and heavy-hitters. Both of these models have received an enormous amount of attention in the streaming literature, beginning with [CCFC02] who proposed the CountSketch algorithm for finding heavy items, all the way to [BCI⁺17] who gave the optimal space bound for insertion-only streams¹, removing one of the log factors of CountSketch. While fully tight bounds are known for heavy-hitters (up to a $\log(1/\epsilon)$ factor in insertion streams), in the sliding window model tight bounds are still unknown. Previous work [BGL⁺18] gave a $\log^2 n$ lower bound, and a $\log^3 n$ upper

*Authors William Swartworth and David Woodruff were supported by a Simons Investigator Award and Office of Naval Research award number N000142112647.

¹This means that items may be added to the stream, but not removed.

bound in the sliding window model. We close this gap, and show that a $\log^2 n$ dependence suffices for computing heavy-hitters on a particular sliding window.

In the F_p moment estimation problem, the goal is to approximate the p^{th} moment for the frequency vector of all items inserted into the stream. Like heavy-hitters, this problem has a long history going back to [AMS96]. Since then, a line of work has led to nearly tight bounds both for large moments $p > 2$, which does not admit polylogarithmic sized sketches [BYJKS04, IW05], as well as the small moments $1 \leq p \leq 2$ [Ind06, KNW10] that we consider here. While this problem has received considerable attention in the sliding window model [BO07a, BO10, WZ22], there still remains a gap in the literature. We aim to close this gap for the problem of estimating the F_p moment on a sliding window.

1.1 Our models

The sliding window model. There are several slightly different models that one can consider. By default we consider an insertion-only stream of length m consisting of items x_1, \dots, x_m . In the sliding window model, there is a window of size W consisting of the W most recent updates to the stream. A correct sliding window algorithm may be queried for the desired statistic at any time t , and it must produce a correct estimate for the statistic on the stream x_{m-W+1}, \dots, x_m of the W most recent updates. When we refer to the failure probability for such an algorithm, we mean the probability of failure for a single query (for an arbitrary value of t).

One could also consider the problem of tracking a stream statistic over the sliding window *at all times*. This could be accomplished in our framework simply by making the failure probability $1/m$.

One could also consider a more general model where there is no fixed window. Instead at a given query time t , the algorithm receives a positive integer n and must estimate the stream statistic on the portion of the stream in $(t - n, t]$. The difference is that n is now not known until runtime. Our algorithms all apply to this more general model, and our lower bounds hold in the standard sliding window model. This model may be useful if one wants to observe how the stream has changed up to the present time. For instance if we were monitoring network traffic, we might be interested in which IP addresses occurred frequently in the past hour, day, week, etc.

F_p moment estimation. In the version of the $(\varepsilon, \delta, F_p)$ moment estimation problem that we consider, we receive a series of insertion-only updates x_1, \dots, x_m . At any point in time t the algorithm may be asked to output an estimate \hat{F} of the F_p moment of the window $W = [t - n + 1, t]$ consisting of updates x_{t-n+1}, \dots, x_t . If $f_i^{(W)}$ is the frequency count of universe item i over the window, then the F_p moment over the window is defined to be $F_p(f^{(W)}) = \sum_i (f_i^{(W)})^p$. In order to be correct, the algorithm must satisfy $\hat{F} = (1 \pm \varepsilon)F_p(f^{(W)})$ with probability at least $1 - \delta$.

Heavy hitters. We say that item i is (ε, ℓ_p) -heavy for a frequency vector f if $f_i \geq \varepsilon \|f\|_p$. In the (ε, ℓ_p) -heavy hitters problem the goal is to output a collection \mathcal{H} of $O(\varepsilon^{-p})$ universe items such that all (ε, ℓ_p) -heavy items for $f^{(W)}$ are contained in \mathcal{H} and such that all elements of \mathcal{H} are at least $c\varepsilon$ -heavy where $c > 0$ is a constant.

1.2 Prior Work

[BO07a] introduced the smooth histogram approach for solving problems in the sliding window model. For a function f , given adjacent substreams A, B , and C , an (α, β) -smooth function demands that if $(1 - \beta)f(A \cup B) \leq f(B)$, then $(1 - \alpha)f(A \cup B \cup C) \leq f(B \cup C)$ for some parameters $0 < \beta \leq \alpha < 1$.

Throughout the stream, they use the smooth histogram to remove the unnecessary information and only keep track of the time when F_p differs by $(1 + \epsilon^p)$. For $p \in (1, 2]$, their work achieves space $O(\epsilon^{-(2+p)} \log^3 n)$.

[WZ22] introduced a new tool called a Difference Estimator. The idea is to estimate the difference of $F_p(u+v)$ and $F_p(v)$ with additive error $\epsilon \cdot F_p(v)$, given that $\max(F_p(u), F_p(u+v) - F_p(v)) \leq \gamma F(v)$, the dimension of the sketch only needs to be $\tilde{O}(\frac{\gamma^{2/p}}{\epsilon^2})$. They use the idea in [BO07a] to maintain a constant factor partition on the top level. Then using a binary tree-like structure to give a fine-grained partition, they use a Difference Estimator to exclude the part that is not in the window.

[BCI⁺17] gives a $\tilde{O}(\epsilon^{-2} \log n)$ bits of space algorithm to give strong tracking over ℓ_2 in insertion-only stream. Also, [BDN17] generalize this to $p \in (0, 2)$. They introduce a weak tracking property, which is a key motivation for our **Strong Estimator**.

For heavy hitters, [BGL⁺18] gives a $\tilde{O}(\epsilon^{-p} \log^3 n)$ algorithm, with the ℓ_2 estimation being the bottleneck. If given the ℓ_2 estimation over the window, then their algorithm can work in space $\tilde{O}(\epsilon^{-p} \log^2 n)$. This is the result we apply to obtain improved heavy-hitter bounds from our F_2 estimator. They also prove an $\Omega(\epsilon^{-p} \log^2 n)$ lower bound for the heavy hitter problem in the sliding window model.

1.3 Our Results

Problem (in Sliding Window Model)	Previous Bound	New Bound
L_p -Heavy Hitters, $p \in (0, 2]$	$\tilde{O}(\epsilon^{-p} \log^3 n)$ $\Omega(\epsilon^{-p} \log^2 n)$ [BGL ⁺ 18]	$\tilde{O}(\epsilon^{-p} \log^2 n)$
F_p Estimation, $p \in (1, 2]$	$\tilde{O}(\epsilon^{-2} \log^3 n)$ [WZ22]	$\tilde{O}(\epsilon^{-p} \log^2 n + \epsilon^{-2} \log n)$, $\Omega(\epsilon^{-p} \log^2 n + \epsilon^{-2} \log n)$

Table 1: Results in the Sliding Window Model

F_p estimation. We give an algorithm for F_p moment estimation for $1 < p \leq 2$ that achieves optimal bounds in the sliding window model, in terms of the accuracy ϵ and the window size n .

Theorem 1.1. *There is an algorithm that uses*

$$O\left((\epsilon^{-p} \log^2 n + \epsilon^{-2} \log n \log^4 \frac{1}{\epsilon})(\log \frac{1}{\epsilon} + (\log \log n)^2)\right) = \tilde{O}(\epsilon^{-p} \log^2 n + \epsilon^{-2} \log n)$$

bits of space, and solves the (ϵ, F_p) moment estimation problem in the sliding window model.

Note that this result is for approximating the F_p on any *fixed* window with $2/3$ probability. To track F_p at all times, and guarantee correctness at all times with good probability, one could apply our algorithm $O(\log m)$ times in parallel, and take the median estimate at each time.

Heavy hitters. As a consequence of our F_2 approximation scheme, we obtain an optimal heavy-hitters algorithm by combining with the results of [BGL⁺18].

Theorem 1.2. *Given $\epsilon > 0$ and $0 < p \leq 2$, there exists an algorithm in the sliding window model that, with probability at least $\frac{2}{3}$, outputs all indices $i \in [U]$ for which frequency $f_i \geq \epsilon \ell_p^W$, and reports no indices $i \in [U]$ for which $f_i \leq \frac{\epsilon}{12} \ell_p^W$. The algorithm has space complexity $O(\epsilon^{-p} \log^2 n (\log \frac{1}{\epsilon} + \log \log n))$.*

An earlier version of [BGL⁺18] claimed this bound for the stronger problem of tracking the heavy-hitters on the window at all times. Unfortunately this original bound was incorrect, and their algorithm actually gives a $\log^3 n$ dependence, even in our setting where we are interested in obtaining good success probability for a single window.

Lower bounds. We also show that our F_p estimation algorithms are tight in the sliding window model up to $\log \frac{1}{\varepsilon} \log \log n$ factors. Specifically, we show

Theorem 1.3. *Fix $p \in (1, 2]$. Suppose that \mathcal{A} is a streaming algorithm that operates on a window of size n , and outputs a $(1 \pm \varepsilon)$ approximation of F_p on any fixed window with 9/10 probability. For $n \geq U$, \mathcal{A} must use at least*

$$\Omega\left(\varepsilon^{-p} \log^2(\varepsilon U) + \varepsilon^{-2} \log(\varepsilon^{1/p} U)\right)$$

space.

The first term in the lower bound is our contribution. The second term applies to general insertion-only streams, and follows from a recent work of [BZ24]. While the lower bound only applies to $p > 1$, our proof also applies to $p = 1$ if the stream is allowed to contain empty insertions that do not affect the frequency counts, but that move the window. (Without empty insertions $p = 1$ is trivial, since F_1 is always the window size.)

1.4 Notation

We use m to denote the length of the stream, U be the universe size, and $u_1, \dots, u_m \in [U]$ to denote the elements in the stream. For sliding window, we use n to denote the window size. We use $\text{poly}(n)$ to denote a constant degree polynomial in n . We assume $m \leq \text{poly}(n)$, $U \leq \text{poly}(n)$.

For interval $[l, r] \subseteq [1, m]$, we use $x^{(l, r)}$ to denote the frequency vector of elements u_l, u_{l+1}, \dots, u_r . For $p \in (0, 2]$, we use $\ell_p^{(l, r)}$ to denote the ℓ_p norm of the frequency vector $x^{(l, r)}$, more specifically, $\|x^{(l, r)}\|_p$, and $F_p^{(l, r)}$ to denote the F_p moment of frequency vector $x^{(l, r)}$, more specifically, $\|x^{(l, r)}\|_p^p$. For window $W = [l, r]$, we also use ℓ_p^W or F_p^W to denote the ℓ_p norm or F_p frequency moment on window W . We use $a = (1 \pm \varepsilon)b$ to denote $a \in [(1 - \varepsilon)b, (1 + \varepsilon)b]$.

1.5 Technical Overview

F_p estimation. To motivate our approach, consider approximating F_2 to within a constant factor. We first recall the framework introduced by [BO07b] for the sliding window model. The rough idea is to start a new F_p estimation sketch at each point in the stream. This clearly gives a sliding window estimate, but uses far too much space. However, the observation is that if sketch i and sketch $i + 2$ give similar estimates for F_p at a given point in the stream, then they will continue to do so for the remainder of the stream. The estimate given by sketch $i + 1$ is sandwiched between the estimates from the neighboring sketches, so sketch $i + 1$ is redundant, and we may safely delete it. After deleting all redundant sketches, we are left with $O(\log n)$ sketches (for constant ε) with geometrically increasing F_p estimates.

Each of these F_2 sketches uses $O(\log n \log \frac{1}{\delta})$ bits of space to estimate F_2 with probability $1 - \delta$, so this approach requires $O(\log^2 n \log \frac{1}{\delta})$ bits. Since we only store $\log n$ sketches at a time, it might seem that we can take $\delta = O(\frac{1}{\log n})$. This is a subtle point and has been a source of error in prior work.

In fact we cannot take $\delta = O(\frac{1}{\log n})$ if we are only guaranteed that our sketches correctly estimate F_2 with probability at least $1 - \delta$. To see why, suppose that each sketch we create is correct with

probability at least $1 - \delta$ and is otherwise allowed to be adversarially incorrect with probability δ . With $\delta = O(1/\log n)$, it is likely that two consecutive sketches will eventually both be incorrect. In that case, they can conspire to in turn sandwich and wipe out all sketches in front of them, from which we will not be able to recover. While extreme, this example illustrates that an F_p estimation sketch with a failure probability guarantee is not sufficient for us. This is why prior approaches for estimating the ℓ_p norm or F_p moment in sliding windows require $\Omega(\log^3 n)$ bits of space, with one of the logarithmic factors arising from the need for sketches with high success rates. These high-success-rate sketches necessitate union bounds over all sketches, even though most of them are not kept by the algorithm when we make a query.

Of course we should not expect sketches to fail this way in practice. Could there be a somewhat stronger guarantee that rules out this type of adversarial behavior? In this work, we overcome this limitation by introducing a new tool: the **Strong Estimator**. This estimator enables a refined analysis that avoids the union-bound overhead, allowing us to achieve the first space complexity which breaks the $O(\log^3 n)$ bit barrier.

The intuition of **Strong Estimator** is for a window $[l, r]$, we do not need our estimator to be right (meaning give an ϵ approximation) on any sub-interval of this window, but rather only an additive error: ϵ fraction of the ℓ_p norm of the window suffices for maintaining the timestamps. Thus we can avoid the union bound for all sub-intervals to be right. The intuition is similar to the weak tracking in [BDN17].

In Section 3, we show that **Strong Estimator** with the simplest algorithm in [BO07a] suffices to give an ϵ approximation to the ℓ_p norm of the sliding window. However, this simple algorithm uses $\tilde{O}(\epsilon^{-3p} \log^2 n)$ bits of space, which requires a high dependence on ϵ .

In Section 4, we introduce **Difference Estimator** [WZ22] to improve the ϵ dependence. We first use our simple algorithm to give a constant factor partition on the top level. We then use a binary tree-like structure to make a more detailed partition between the top level, and use **Difference Estimator** estimator to exclude the contribution that is not in the window. The algorithm in [WZ22] requires $\tilde{O}(\epsilon^{-2} \log^3 n)$ space. The extra $\log n$ factor also comes from the high success probability requirement. Using the **Strong Estimator** and the structure of their algorithm, we can obtain an algorithm using $O(\epsilon^{-2} \log^2 n)$ bits of memory. We further discover that the bottleneck of the algorithm comes from the **Difference Estimator**, which can be stored by rounded values using the rounding procedure in [JW23]. We thus improve the space to $\tilde{O}(\epsilon^{-p} \log^2 n + \epsilon^{-2} \log n)$ bits, which gives a separation between $p = 2$ and $p < 2$. We further prove a lower bound on F_p moment estimation over sliding windows, see below, showing that our algorithm is nearly optimal.

Heavy Hitters. [BGL⁺18] showed that if one has an ℓ_2 estimation algorithm over a sliding window, then one can find the ℓ_2 heavy hitters with $\tilde{O}(\epsilon^{-2} \log^2 n)$ additional bits of space. In [JST11], they show that an $\epsilon^{p/2}$ -heavy hitter algorithm for ℓ_2 also finds ϵ -heavy hitters for ℓ_p . Thus, using our simplest algorithm to provide a constant ℓ_2 estimation over the window will give us an algorithm that uses $\tilde{O}(\epsilon^{-p} \log^2 n)$ bits of space, and returns all the ϵ - ℓ_p heavy hitters. This matches the lower bound $\Omega(\epsilon^{-p} \log^2 n)$ for heavy hitters in [BGL⁺18].

Lower Bounds. For our lower bound for F_p estimation, we construct a stream that consists of roughly $\log(n)$ blocks, where each block has half the F_p value of the prior block. Within each block, we place ϵ^{-p} items that are each (ϵ, ℓ_p) -heavy for the block, and arrange them so that each of these heavy items occurs in a contiguous “mini-block”. Overall, there are $\frac{1}{\epsilon^p} \log(n)$ of these “mini-blocks”.

The main observation is that a correct streaming algorithm for F_p -moment estimation in the sliding window model must remember the location of each mini-block. Doing so requires $\log n$ bits of

space per mini-block as long as there are at least, say, $n^{0.1}$ disjoint positions that each block can occur in, resulting in our $\Omega(\frac{1}{\varepsilon^p} \log^2 n)$ lower bound.

To see why our algorithm must remember each location, consider one of the heavy items x in our construction, and suppose we want to decide whether the mini-block for x occurs entirely before or entirely after stream index i . We first shift the sliding window so that it begins at index i . We would like to decide whether or not x occurs in the resulting window. By the geometric decay of the blocks, x remains heavy for the entire window. Then to decide if x remains in the window we append Q copies of x to the window, where Q is a rough estimate of the F_p over the window. If x does not occur in the window, then appending the x 's increases the F_p by Q^p . On the other hand, if x does occur roughly εQ times in the window then the F_p increases by $(\varepsilon Q + Q)^p - (\varepsilon Q)^p = \Theta((1 + \varepsilon)Q^p)$. Since Q^p is approximately the F_p moment of the window, a $\Theta(\varepsilon)$ heavy-hitters algorithm can distinguish between these two cases. A minor issue is that as stated, we would need to append the x 's without shifting the window past the index i . However we can easily fix this by appending the x 's before performing appending singletons to accomplish the shift. To make the above intuition precise we give a reduction from the IndexGreater communication game in Section 5.

2 Preliminary Results

In this section, we will introduce some basic definitions and lemmas that form the foundation of our **Strong Estimator**. We will also introduce the previous results for heavy hitters. We first require the following definition for p -stable distribution.

Definition 2.1. (*p -stable distribution*). [Zol86] For $0 < p \leq 2$, there exists a probability distribution \mathcal{D}_p called the p -stable distribution so that for any positive integer n with $Z_1, \dots, Z_n \sim \mathcal{D}_p$ and vector $x \in \mathbb{R}^n$, then $\sum_{i=1}^n Z_i x_i \sim \|x\|_p Z$ for $Z \sim \mathcal{D}_p$.

The probability density function f_X of a p -stable random variable X satisfies $f_X(x) = \Theta\left(\frac{1}{1+|x|^{1+p}}\right)$ for $p < 2$, while the normal distribution corresponds to $p = 2$. Moreover, [Nol01] details standard methods for generating p -stable random variables by taking θ uniformly at random from the interval $[-\frac{\pi}{2}, \frac{\pi}{2}]$, r uniformly at random from the interval $[0, 1]$, and setting

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

These p -stable random variables are crucial to obtaining our **Strong Estimator**.

Definition 2.2. (*Strong Estimator*) Let $\epsilon_1 \leq \epsilon_2 \in (0, 1)$, $p \in (0, 2]$ and let u_1, u_2, \dots, u_m be the sequence of stream elements. Let $x^{(i,j)}$ denote the frequency vector for elements u_i, \dots, u_j . We say that f_p has the $(\epsilon_1, \epsilon_2, \delta)$ **Strong Estimator property** on the window $[l, r]$ if with probability $1 - \delta$, we have the bound $f_p(x^{(a,b)}) = (1 \pm \epsilon_1) \ell_p^{(a,b)} \pm \epsilon_2 \ell_p^{(l,r)}$ for all sub-windows $[a, b] \subseteq [l, r]$ simultaneously. We say that f_p has the $(\epsilon_1, \epsilon_2, \delta)$ **Strong Estimator property** if it has the $(\epsilon_1, \epsilon_2, \delta)$ **Strong Estimator property** on all windows $[l, r]$.

For simplicity, we will use $f_p^{(a,b)}$ to denote the estimate $f_p(x^{(a,b)})$.

To support the construction and analysis of such an estimator, we rely on several probabilistic tools. Lemma 2.3 gives a Chernoff-type concentration bound for k -wise independent random variables. Lemma 2.4 provides tail bounds on the supremum of inner products between a p -stable random vector and a sequence with an increasing frequency vector.

Lemma 2.3. [BR94] Let $X_1, \dots, X_n \in \{0, 1\}$ be a sequence of k -wise independent random variables, and let $\mu = \sum \mathbb{E}X_i$. Then

$$\forall \lambda > 0, \mathbb{P}\left(\sum X_i \geq (1 + \lambda)\mu\right) \leq \exp(-\Omega(\min\{\lambda, \lambda^2\}\mu)) + \exp(-\Omega(k))$$

Lemma 2.4. [BDN17] Let $x^{(1)}, x^{(2)}, \dots, x^{(m)} \in \mathbb{R}^n$ satisfy $0 \preceq x^{(1)} \preceq x^{(2)} \preceq \dots \preceq x^{(m)}$. Let $Z \in \mathbb{R}^n$ have k -wise independent entries marginally distributed according to \mathcal{D}_p . Then for some C_p depending only on p ,

$$\mathbb{P}\left(\sup_{k \leq m} \left|\langle Z, x^{(k)} \rangle\right| \geq \lambda \|x^{(m)}\|_p\right) \leq C_p \left(\frac{1}{\lambda^{2p/(2+p)}} + k^{-1/p}\right)$$

Finally, we include two additional lemmas that are essential for our heavy hitter detection algorithm in the sliding window model.

Lemma 2.5, due to Braverman et al. [BGL⁺18], shows that a constant-factor approximation to the ℓ_2 norm within a window suffices to recover all ϵ -heavy hitters under the ℓ_2 norm, with provable accuracy guarantees and near-optimal space complexity.

Lemma 2.5. [BGL⁺18] For a fixed window W , if given a $\frac{1}{2}$ approximation of ℓ_2^W , then there is an algorithm that outputs all the ϵ heavy hitters for ℓ_2 within the window, and can guarantee that all the output elements have frequency $\geq \frac{\epsilon}{12} \ell_2^W$. This algorithm uses $O(\epsilon^{-2} \log^2 n (\log \frac{1}{\epsilon} + \log \log n))$ and has success probability $\geq \frac{2}{3}$.

To extend this result to the ℓ_p setting for any $p \in (0, 2]$, we use Lemma 2.6, which establishes that any algorithm capable of recovering $\epsilon^{p/2}$ -heavy hitters under ℓ_2 (with a suitable tail guarantee) also successfully identifies all ϵ -heavy hitters under the ℓ_p norm.

Lemma 2.6. [JST11] For any $p \in (0, 2]$, any algorithm that returns the $\epsilon^{p/2}$ -heavy hitters for ℓ_2 satisfying the tail guarantee also finds the ϵ -heavy hitters for ℓ_p .

3 Warmup: Constant-factor F_p approximation

In this section, we will introduce our result for **Strong Estimator**. This estimator enables us to design a constant-factor approximation algorithm for F_p , and in combination with previous results, it leads to a nearly optimal algorithm for identifying ϵ -heavy hitters over sliding windows.

3.1 Constructing a Strong Estimator

First, we will show how to construct a space efficient **Strong Estimator** defined in Definition 2.2.

Construction 3.1. Let d, r and s be parameters to be specified later, the **Strong Estimator** takes $\Pi \in \mathbb{R}^{d \times U}$ to be a random matrix with entries drawn according to \mathcal{D}_p , and such that the rows are r -wise independent, and all entries within a row are s -wise independent. For frequency vector $x^{(a,b)}$, the estimate $f_p(x^{(a,b)})$ is given by $\text{median}(|(\Pi x^{(a,b)})_1|, |(\Pi x^{(a,b)})_2|, \dots, |(\Pi x^{(a,b)})_d|)$.

The following lemma shows that, by appropriately setting the parameters d, r , and s , the above construction yields a valid **Strong Estimator**.

Lemma 3.2. Let $\epsilon_1, \epsilon_2 \in (0, 1)$ with $\epsilon_2 \leq \epsilon_1$, $\delta \in (0, 1)$, and $p \in [1, 2]$. Given a stream of elements u_1, u_2, \dots, u_m , Construction 3.1 yields an $(\epsilon_1, \epsilon_2, \delta)$ **Strong Estimator** by setting the parameters as follows:

$$d = \Theta\left(\epsilon_1^{-2} \left(\log \frac{1}{\epsilon_2} + \log \frac{1}{\delta}\right)\right), \quad r = \Theta\left(\log \frac{1}{\epsilon_2} + \log \frac{1}{\delta}\right), \quad s = \Theta\left(\epsilon_1^{-p}\right).$$

To show the space efficiency, we also analyze the space complexity of storing the matrix Π used in the construction.

Lemma 3.3. *The memory required to store the matrix Π for an $(\epsilon_1, \epsilon_2, \delta)$ Strong Estimator is*

$$O\left(\epsilon_1^{-p} \left(\log \frac{1}{\epsilon_2} + \log \frac{1}{\delta}\right) \log(mU)\right) \text{ bits.}$$

Last, we will show an additional property of Strong Estimator, which is crucial for solving F_p estimation over sliding windows.

Lemma 3.4. *(Additional Property for the Strong Estimator) The $(\epsilon_1, \epsilon_2, \delta)$ Strong Estimator f_p has the following property:*

For a sequence of elements in the stream u_1, u_2, \dots, u_m and a window $[l, r]$, with probability $\geq 1 - \delta \cdot \log r$, for any interval $[a, b]$ that $1 \leq a < l, l \leq b \leq r$, $f_p(x^{(a,b)}) = (1 \pm (\epsilon_1 + 2\epsilon_2))\ell_p^{(a,b)} \pm 2\epsilon_2\ell_p^{(l,r)}$.

3.2 Constant-factor F_p approximation

First, we will introduce the algorithm given in [BO07a]: it maintains $O(\epsilon^{-p} \log n)$ timestamps to track the times at which ℓ_p changes by a factor of $(1 + \epsilon^p)$. This algorithm requires $O(\log^3 n)$ bits of space, where one $\log n$ factor accounts for the number of sketches, another $\log n$ arises from the need for high success probability in order to apply a union bound over all sketches, and the final $\log n$ comes from the fact that each entry in the sketch requires $\log n$ bits.

Our key improvement over their algorithm is that, by using the Strong Estimator, we no longer require the sketches to have high success probability. The Strong Estimator provides guarantees over all sub-intervals directly, eliminating the need for a union bound.

We present an ϵ -approximation algorithm for ℓ_p over sliding windows that uses $\tilde{O}(\epsilon^{-3p} \log^2 n)$ space. This is the first algorithm achieving $o(\log^3 n)$ space for this problem. As ℓ_p estimation and F_p estimation are fundamentally the same, we do not distinguish between the two here.

The following theorem formalizes the guarantees of our approach:

Theorem 3.5. *For any fixed window W , with probability $\geq 1 - \delta$, Algorithm 1 will give an ϵ approximation of ℓ_p^W , and use $O(\epsilon^{-3p} \log^2 n (\log \frac{1}{\epsilon} + \log \frac{1}{\delta} + \log \log n))$ bits of space.*

We now apply our algorithm to the heavy hitter problem in sliding windows. Combining our $\frac{1}{2}$ -approximation for ℓ_2 with known results for heavy hitters (Lemma 2.5 and Lemma 2.6), we obtain the following result:

Theorem 3.6. *(Main Theorem on Heavy Hitters) Given $\epsilon > 0$ and $0 < p \leq 2$, there exists an algorithm in the sliding window model that, with probability at least $\frac{2}{3}$, outputs all indices $i \in [U]$ for which frequency $f_i \geq \epsilon \ell_p^W$, and reports no indices $i \in [U]$ for which $f_i \leq \frac{\epsilon}{12} \ell_p^W$. The algorithm has space complexity (in bits) $O\left(\frac{1}{\epsilon^p} \log^2 n (\log \frac{1}{\epsilon} + \log \log n)\right)$.*

Proof. **TOPROVE 0** □

Define $t_k(j)$ to be the k -th timestamps when the stream goes to j . The following lemma shows that the ℓ_p norm does not decrease significantly between two consecutive timestamps.

Lemma 3.7. *Let $p \in [1, 2]$, for fixed window $W = [i - n, i]$, with probability $\geq 1 - \delta$, either $t_1(i) + 1 = t_2(i)$, or $\ell_p^{(t_2(i), i)} \geq (1 - \frac{\epsilon}{2}) \ell_p^{(t_1(i), i)}$.*

Now we prove theorem 3.5.

Proof. **TOPROVE 1** □

Algorithm 1 Smooth Histogram with Strong Estimator

Input: Stream $a_1, a_2, \dots \in [U]$, window length n , approximate ratio $\epsilon \in (0, 1)$ and error rate $\delta \in (0, 1)$, parameter p .

Output: At each time i , output a $(1 \pm \epsilon)$ approximation to ℓ_p in window $[i - n, i]$.

Initialization:

1. Generate the matrix $\Pi \in \mathbb{R}^{d \times U}$ for an $(\frac{\epsilon^p}{64}, \frac{\epsilon^p}{128}, \delta' = \frac{\delta}{O(\log m)})$ Strong Estimator, where $d = O(\epsilon^{-2p} \log \frac{1}{\epsilon \delta'})$.
2. Random matrix $\pi \in \mathbb{R}^{d' \times U}$ for a Indyk's p -stable Sketch, where $d' = O(\epsilon^{-2} \log \frac{1}{\delta})$.

Let $f^{(D)}$ be the estimated value of $\|x^{(D)}\|_p$ from Strong Estimator on data stream D , and $g^{(D)}$ be the estimated value of $\|x^{(D)}\|_p$ from Indyk's p -stable Sketch.

During the Stream:

- 1: $s \leftarrow 0, i \leftarrow 1$.
 - 2: **for** each update a_i **do**
 - 3: $s = s + 1$.
 - 4: let $t_s = i$, start a copy of Strong Estimator using matrix Π and Indyk's p -stable Sketch using matrix π from t_s .
 - 5: **for** j in $0, 1, \dots, s$ **do**
 - 6: **while** $f^{(t_{j+2}, i)} \geq (1 - \frac{\epsilon^p}{32})f^{(t_j, i)}$ **do**
 - 7: Delete t_{j+1} , update indices, $s = s - 1$
 - 8: **end while**
 - 9: **end for**
 - 10: If $t_2 < i - n$, delete t_1 , update indices, $s = s - 1$.
 - 11: Output $g^{(t_2, i)}$ as the estimation of ℓ_p^W .
 - 12: **end for**
-

4 Improving the ϵ dependence

4.1 Preliminary Results

Here we use $F(u)$ to denote $\|u\|_p^p$ for frequency vector u .

First, we introduce the definition and construction of Difference Estimator, which is essential in our algorithm that gives tight bound on F_p estimation over sliding window.

Definition 4.1. (Difference Estimator) (Definition 8.2, [WZ22])

Given a stream \mathcal{S} and fixed times t_1, t_2 , and t_3 , let frequency vectors u and v be induced by the updates of \mathcal{S} between times $[t_1, t_2]$ and $[t_2, t_3]$. Given an accuracy parameter $\epsilon > 0$ and a failure probability $\delta \in (0, 1)$, a streaming algorithm $\mathcal{C}(t_1, t_2, t, \gamma, \epsilon, \delta)$ is a $(\gamma, \epsilon, \delta)$ -suffix difference estimator for a function F if, with probability at least $1 - \delta$, it outputs an additive $\epsilon \cdot F(v + w_t)$ approximation to $F(u + v + w_t) - F(v + w_t)$ for all frequency vectors w_t induced by $[t_3, t]$ for times $t > t_3$, given $\min(F(u), F(u + v) - F(v)) \leq \gamma \cdot F(v)$ for a ratio parameter $\gamma \in (0, 1]$.

Lemma 4.2. (Construction of Difference Estimator) (Lemma 4.9, Lemma 3.6 [WZ22])

Let the dimension $d = \mathcal{O}\left(\frac{\gamma^{2/p}}{\epsilon^2} \left(\log \frac{1}{\epsilon} + \log \frac{1}{\delta}\right)\right)$.

For $0 < p < 2$, it suffices to use a sketching matrix $A \in \mathbb{R}^{d \times U}$ of i.i.d. entries drawn from the p -stable distribution \mathcal{D}_p , with dimension d to obtain a $(\gamma, \epsilon, \delta)$ -difference estimator for F_p . To store such a matrix A requires $O(d \log n (\log \log n)^2)$ space.

Moreover, the estimation of $F(u+v) - F(v)$ is given by:

1. For a parameter $q = 3$, let each $z_i = \prod_{j=q(i-1)+1}^{qi} (Au + Av)_j^{p/q}$ and $z'_i = \prod_{j=q(i-1)+1}^{qi} (Av)_j^{p/q}$.
2. Output the arithmetic mean of $(z_1 - z'_1), (z_2 - z'_2), \dots, (z_{d/q} - z'_{d/q})$.

For $p = 2$, it suffices to use a sketching matrix $M \in \mathbb{R}^{d \times U}$ that each entry $M_{i,j}$ of M is a 4-wise independent random sign scaled by $\frac{1}{\sqrt{d}}$ to obtain a $(\gamma, \epsilon, \delta)$ -difference estimator for F_2 . To store such a matrix M requires $O(d \log n)$ space.

The estimation of $F(u+v) - F(v)$ is given by $\|M(u+v)\|_2^2 - \|Mv\|_2^2$.

Next, we introduce the rounding procedure described in [JW23]. Although this procedure is originally designed for a distributed setting, the process of merging two sketches can be interpreted as two nodes transmitting their respective sketches to a parent node, which then performs a new rounding operation based on the information received from its children.

Definition 4.3. [JW23](Rounding random variable) For any real value $r \in \mathbb{R}$, let $i_r \in \mathbb{Z}$ and $\alpha_i \in \{1, -1\}$ be such that $(1 + \gamma)^{i_r} \leq \alpha_i r \leq (1 + \gamma)^{i_r+1}$. Now fix p_r such that:

$$\alpha_i r = p_r (1 + \gamma)^{i_r+1} + (1 - p_r) (1 + \gamma)^{i_r}$$

We then define the rounding random variable $\Gamma_\gamma(r)$ by

$$\Gamma_\gamma(r) = \begin{cases} 0 & \text{if } r = 0 \\ \alpha_i (1 + \gamma)^{i_r+1} & \text{with probability } p_r \\ \alpha_i (1 + \gamma)^{i_r} & \text{with probability } 1 - p_r \end{cases}$$

Procedure 4.4. (Recursive Randomized Rounding)

1. Choose random vector $Z \in \mathbb{R}^n$ using shared randomness.
2. Receive rounded sketches $r_{j_1}, r_{j_2}, \dots, r_{j_{t_j}} \in \mathbb{R}$ from the t_j children of node j in the prior layer (if any such children exist).
3. Compute $x_j = \langle X_j, Z \rangle + r_{j_1} + r_{j_2} + \dots + r_{j_{t_j}} \in \mathbb{R}$.
4. Compute $r_j = \Gamma(x_j)$. If player $j \neq \mathcal{C}$, then send r_j to the parent node of j in T . If $j = \mathcal{C}$, then output r_j as the approximation to $\langle Z, \mathcal{X} \rangle$.

Lemma 4.5. (Lemma 2, [JW23])

Let U be the universe space, and m be the total point in the graph, d be the diameter of this graph, $\epsilon, \delta \in (0, 1)$. Fix $p \in [1, 2]$, and let $Z = (Z_1, Z_2, \dots, Z_U) \sim D_p^U$. Then the above procedure when run on $\gamma = (\epsilon \delta / (d \log(Um)))^C$ for a sufficiently large constant C , produces an estimate $r_{\mathcal{C}}$ of $\langle Z, \mathcal{X} \rangle$, held at the center vertex \mathcal{C} , such that $\mathbb{E}[r_{\mathcal{C}}] = \langle Z, \mathcal{X} \rangle$. Moreover, over the randomness used to draw Z , with probability $1 - \delta$ for $p < 2$, and with probability $1 - e^{-1/\delta}$ for Gaussian Z , we have $\mathbb{E}[(r_{\mathcal{C}} - \langle Z, \mathcal{X} \rangle)^2] \leq (\epsilon/\delta)^2 \|\mathcal{X}\|_p$. Thus, with probability at least $1 - O(\delta)$, we have

$$|r_{\mathcal{C}} - \langle Z, \mathcal{X} \rangle| \leq \epsilon \|\mathcal{X}\|_p$$

Moreover, if $Z = (Z_1, Z_2, \dots, Z_n) \in \mathbb{R}^n$ where each $Z_i \in \{1, -1\}$ is a 4-wise independent Rademacher variable, then the above bound holds with $p = 2$ (and with probability $1 - \delta$).

Now, we give our own rounding procedure.

Procedure 4.6. Suppose we want to update the rounded sketch of window $[l, r]$. The sketch has a matrix $\Pi \in D_p^{d \times U}$ for $p \in (0, 2)$. For $p = 2$, the sketch has a matrix $\Pi \in \{1, -1\}^{d \times U}$, with each entry being a Rademacher variable and each row being 4-wise independent. The precise sketch for window $[l, r]$ is $(|\Pi x^{(l,r)}|_1, \dots, |\Pi x^{(l,r)}|_d) \in \mathbb{R}^d$. Let $\mathcal{P}(\Pi x^{(l,r)}) \in \mathbb{R}^d$ be the sketch vector after the rounding procedure.

There are 2 cases: First is given a non-rounded sketch, and then rounding the sketch. In this case, for each $i \in [d]$, $\mathcal{P}(\Pi x^{(l,r)})_i = \Gamma_\gamma((\Pi x^{(l,r)})_i)$.

The second case is given the rounded sketch of 2 sub-intervals $[l, k]$ and $[k+1, r]$. For this case, $\mathcal{P}(\Pi x^{(l,r)})_i = \Gamma_\gamma(\mathcal{P}(\Pi x^{(l,k)})_i + \mathcal{P}(\Pi x^{(k+1,r)})_i)$.

Lemma 4.7. Let $[l, r]$ be a window, $\Pi \in \mathbb{R}^{d \times U}$ be the sketching matrix (meaning each entry is sampled from \mathcal{D}_p if $0 < p < 2$, or is a Rademacher variable if $p = 2$), and we are running procedure \mathcal{P} described in Procedure 4.6 with parameter $\gamma = (\frac{\epsilon \delta d}{m \log(Um)})^C$ to get a rounded vector for sketch $\Pi x^{(l,r)}$. Then for all $i \in [d]$, we have $|\mathcal{P}(\Pi x^{(l,r)})_i - (\Pi x^{(l,r)})_i| \leq \epsilon \|x^{(l,r)}\|_p$.

4.2 Algorithm Overview

The structure of our algorithm follows [WZ22]. Let the current time be r and query window be $W = [r - n + 1, r]$. Our algorithm first uses Algorithm 1 to maintain a constant factor partition on the top level. We will have $O(\log n)$ timestamps on the top level. Let them be $t_1 < t_2 < \dots < t_s$.

For each of the $O(\log n)$ pieces of the stream, we will partition the substream $[t_i, t_{i+1}]$ into more detailed timestamps. We will have a binary tree-like structure, with $\beta = O(\log \epsilon^{-p})$ at the top level, and each level j will keep roughly $O(2^j)$ timestamps. Define the k -th timestamp on level j between top level i to be $t_{i,j,k}$.

We will use **Difference Estimator** to estimate the difference of F_p between 2 nearby timestamps and the current time r , to exclude the contribution from t_1 to $r - n$. As the **Difference Estimator** is maintained by a binary tree-like structure, we only need to query the **Difference Estimator** a constant number of times on each level. So if each **Difference Estimator** gives an η additive error of F_p^W , then our final approximation will have an $O(\eta \cdot \beta)$ additive error of F_p^W . So a value $\eta = O(\frac{\epsilon}{\log \frac{1}{\epsilon}})$ suffices and the **Difference Estimator** uses $O(\epsilon^{-2} \log^2 n \text{poly}(\log \frac{1}{\epsilon}))$ space in total.

To maintain the binary tree-like structure for the **Difference Estimator**, the algorithm by [WZ22] uses a p -stable sketch with high success probability, and thus by union bound, these sketch always can rule out unnecessary timestamps and retain only the essential ones.

Our algorithm relies on the analysis of **Strong Estimator**, and no such union bound will be used (thus saving a factor of $\log n$), so it will require a more careful analysis. In the following content, we introduce our new techniques.

Algorithm 2 Moment Estimation in the Sliding Window with Optimal Space

Input: Stream $u_1, u_2, \dots \in [U]$, window length n , approximate ratio $\epsilon \in (0, 1)$ and error rate $\delta \in (0, 1)$, parameter p .

Output: At each time i , output a $(1 + \epsilon)$ -approximation to F_p for window $[i - n, i]$.

Initialization: $\beta \leftarrow \left\lceil \log \frac{100 \max(p, 1)}{\epsilon \max(p, 1)} \right\rceil$, $\eta \leftarrow \frac{\epsilon}{2^{25} \cdot \log \frac{1}{\epsilon}}$, $\gamma_j \leftarrow 2^{3-j}$ for all $j \in [\beta]$, $\delta_{dif} = \frac{\delta}{2^{25} \cdot \log \frac{1}{\epsilon}}$

1. Let $\epsilon_0 = \frac{1}{2}$, prepare the matrix Π_{top} for $(\frac{\epsilon_0^p}{64}, \frac{\epsilon_0^p}{128}, \delta_{SE} = \frac{\delta}{\log O(\log m)})$ **Strong Estimator** to maintain a constant factor partition on the top level.
2. Prepare the matrix π for a Indyk's p-stable Sketch with dimension $O(\epsilon^2 \log \frac{1}{\delta})$.
3. To maintain the subroutine level, for each $j \in [\beta]$, prepare $C \cdot 2^j \log n$ random matrices $\Pi_i \in d$ for a large constant C , each responsible for an $(\frac{1}{8}, \frac{2^{-1/p} \epsilon}{2^{32}}, \delta_{sub} = \frac{\delta}{O(C \epsilon^{-p} \log n)})$ **Strong Estimator**. Store them in a queue q_j .
4. For each level $j \in [\beta]$, generate the matrix Π_{dif} for the $(\gamma_j, \eta, \delta_{dif})$ -Difference Estimator, with dimension $d = O(\frac{\gamma_j^{2/p}}{\eta^2} (\log \frac{1}{\eta} + \log \frac{1}{\delta_{dif}}))$.

```

1: for each update  $a_i$  do
2:   NewsSW( $i$ )                                 $\triangleright$  Create new subroutines for each update
3:   MergeSW( $i$ )                                 $\triangleright$  Removes extraneous subroutines
4:   Output StitchSW
5: end for

```

```

1: function NEWSW( $i$ )
2:   Let  $s$  be the number of copies in the histogram,  $t_j$  be the starting time of the  $j$ -th copy.
3:    $t_{s+1} \leftarrow i$ .
4:   Start a new top level Strong Estimator using matrix  $\Pi_{top}$ .
5:   for  $j \in [\beta]$  do                                 $\triangleright$  Start instances of each granularity for each copy
6:      $t_{s+1,j,0} \leftarrow i$ 
7:     Start a new instance of  $(\gamma_j, \eta, \delta_{dif})$  - Difference Estimator SDiffEst( $s + 1, j, 0$ )
8:     Start a new instance of Strong Estimator  $f_{s+1,j,0}$ , using the matrix from  $\text{top}(q_j)$ ,  $\text{pop}(q_j)$ .
9:     Use procedure 4.6 to round the Difference Estimator ending at time  $i$ .
10:  end for
11: end function

```

```

1: function MERGESW( $t$ )
2:   Use Algorithm 1 with  $\epsilon = \frac{1}{2}$  to maintain a constant factor partition at the top level, and an
   Indyk's p-stable Sketch along with each timestamps. The only difference is when deleting sketch
   or adding new sketch, we use Procedure 4.6 to calculate the result of rounding.
3:   for  $i \in [s], j \in [\beta]$  do                                      $\triangleright$  Difference estimator maintenance
4:     Let  $r$  be the number of time steps  $t_{i,j,*}$ 
5:     for  $k \in [r-1]$  do                                            $\triangleright$  Merges two algorithms with "small" contributions
6:       if  $\hat{f}_{i,j,k-1}^{(t_{i,j,k-1}, t_{i,j,k+1})} \leq 2^{(-j-14)/p} \cdot \hat{f}_{i,j,k-1}^{(t_{i,j,k-1}, t)}$  then
7:         if  $t_{i,j,k} \notin \{t_{i,j-1,*}\}$  then
8:           Delete timestamp  $t_{i,j,k}$  and the related Strong Estimator .
9:           Push back the matrix  $\Pi$  to the queue  $q_j$ .
10:          Merge (add) the sketches for SDiffEsT( $i, j, k-1$ ) and SDiffEsT( $i, j, k$ ).
11:          Use Procedure 4.6 to merge the rounding result.
12:        end if
13:      end if
14:    end for
15:  end for
16: end function

```

4.3 Rounding Techniques

In this subsection, we present several space-saving techniques that allow us to maintain correctness guarantees while significantly reducing the storage requirements of the sketching data structures. First, we describe how rounding can be applied to the Strong Estimator sketches $f_{i,j,k}$ to reduce the number of bits required per entry. Next, we address the challenge of bounding the number of timestamps in the absence of a global union bound, introducing a rotating random set technique that probabilistically limits the number of active sketches. Finally, we apply a fine-grained rounding scheme to compress both the Indyk's p-stable Sketch and Difference Estimator sketches, ensuring that their space usage remains within our overall complexity bounds.

Storing the Sketch via Strong Estimator $f_{i,j,k}$ We now describe how to apply rounding to the estimates computed by Strong Estimator to achieve space efficiency.

Throughout the algorithm, the sketch $f_{i,j,k}$ is used to estimate the ℓ_p norm over the interval from timestamp $t_{i,j,k}$ to either another timestamp t' or the current time. While we maintain exact estimates for intervals ending at the current time, we observe that for estimates ending at previous timestamps t' , we can round the values to save space. Specifically, we round $f_{i,j,k}^{(t_{i,j,k}, t')}$ to the nearest power of $(1 + \frac{1}{16})$. The following claim shows that this rounding preserves the Strong Estimator property up to a small additive loss:

Claim 4.8. *Let u_1, \dots, u_m be a stream and $[l, r]$ an arbitrary window. Suppose f satisfies the (ϵ_1, ϵ_2) Strong Estimator property on $[l, r]$. Define $\hat{f}^{(a,b)} = \text{Round}(f^{(a,b)})$ to the nearest power of $1 + \frac{1}{16}$. Then \hat{f} satisfies the $(\epsilon_1 + \frac{1}{8}, (1 + \frac{1}{16})\epsilon_2)$ Strong Estimator property on $[l, r]$.*

Thus, in our analysis, since we use $(\frac{1}{8}, \frac{2^{-1/p}\epsilon}{2^{32}}, \delta_{\text{sub}})$ -Strong Estimator, we may safely treat the rounded sketches as $(\frac{1}{4}, \frac{2^{-1/p}\epsilon}{2^{30}}, \delta_{\text{sub}})$ -Strong Estimator without loss of generality.

However, directly storing $O(s^2)$ rounded values (where $s = O(\epsilon^{-p} \log n)$ is the number of timestamps) would still be space-inefficient. To address this, we introduce the following compression

technique:

Technique 4.9. For each timestamp $t_{i,j,k}$, we maintain an array of length $O(\log n)$. The c -th entry stores the rank of the earliest timestamp t' such that $f_{i,j,k}^{(t_{i,j,k}, t')}$ $\geq (1 + \frac{1}{16})^c$.

A potential issue with this scheme is that the value $f_{i,j,k}^{(t_{i,j,k}, t_u)}$ might be "overestimated" due to a later timestamp t_v with $f_{i,j,k}^{(t_{i,j,k}, t_v)} > f_{i,j,k}^{(t_{i,j,k}, t_u)}$. However, this does not violate the Strong Estimator property, since $f_{i,j,k}^{(t_{i,j,k}, t_u)} < f_{i,j,k}^{(t_{i,j,k}, t_v)} < (1 + \epsilon_1)\ell_p^{(t_{i,j,k}, t_v)} + \epsilon_2\ell_p^{(l,r)} < (1 + \epsilon_1)\ell_p^{(t_{i,j,k}, t_u)} + \epsilon_2\ell_p^{(l,r)}$.

Therefore, this approximation remains valid, and the total space required to store all the Strong Estimator sketches is reduced to $O(s \log n \log s)$ bits.

Challenges in Bounding the Number of Timestamps The analysis in [WZ22] relies on applying a union bound over all sketches and all times, which introduces an additional $\log n$ factor in the space complexity. In contrast, we leverage the Strong Estimator to eliminate the need for such a union bound. However, this also introduces new challenges.

Consider the current window $W = [r - n + 1, r]$, and let l be the earliest index such that $F_p^{(l,r)} \leq 2F_p^W$. Conditioned on the Strong Estimator satisfying its guarantee over $[l, r]$, we can show that the number of timestamps at the first level is bounded and provides reliable guidance for constructing Difference Estimator.

The complication arises because this guarantee only holds with probability $1 - \delta$, and without a global union bound, we cannot ensure that the number of timestamps is always bounded across time. For instance, if the same matrix Π is reused across all Strong Estimator instances, then in the unlikely case that these matrices fail simultaneously, the number of timestamps could grow unbounded.

To mitigate this, we introduce the following technique:

Technique 4.10 (Rotating Random Sets). For each level $j \in [\beta]$, we maintain a queue q_j that holds $C \cdot 2^j \log n$ independent random matrices Π , for a sufficiently large constant C .

Each Strong Estimator instance $f_{i,j,k}$ uses a distinct matrix Π from the queue that has not been used before. When a sketch is deleted, its corresponding matrix is returned to q_j , making it available for reuse.

Lemma 4.11. Fix any time t . With probability at least $1 - \frac{1}{\text{poly}(n)}$, the queue q_j is non-empty for all levels $j \in [\beta]$. This ensures that there are at most $O(2^j \log n)$ active timestamps at level j .

Hence, by a union bound over all levels and time, with high probability, the total number of timestamps across all levels remains bounded by $O(\epsilon^{-p} \log n)$ throughout the entire stream.

Rounding the Indyk's p-stable Sketch and Difference Estimator Our algorithm maintains only $O(\log n)$ timestamps at the top level. However, each of these timestamps stores an instance of Indyk's p-stable Sketch sketch, which uses $O(\epsilon^{-2} \log n)$ bits, and the total space for the Difference Estimator sketches at each level is also $\tilde{O}(\epsilon^{-2} \log n)$. To reduce the overall space to $\tilde{O}(\epsilon^{-p} \log^2 n + \epsilon^{-2} \log n)$, we apply the rounding technique from [JW23] to compress the sketch values.

Lemma 4.12. (Correctness of rounding Indyk's p-stable Sketch)

Let $\epsilon', \delta' \in (0, 1)$, and $[l, r]$ be a window, $\Pi \in \mathbb{R}^{d \times U}$ be the sketch matrix, $\gamma = (\frac{\epsilon' \delta'}{dm \log(Um)})^C$, $\mathcal{P}(\Pi x^{(l,r)})$ be the sketch vector after rounding procedure 4.6 with parameter γ .

Then $\text{median}(|\mathcal{P}(\Pi x^{(l,r)})_1|, \dots, |\mathcal{P}(\Pi x^{(l,r)})_d|) = \text{median}(|(\Pi x^{(l,r)})_1|, \dots, |(\Pi x^{(l,r)})_d|) \pm \epsilon' \|x^{(l,r)}\|_p$ with probability $\geq 1 - \delta'$.

Proof. **TOPROVE 2** □

Setting the ϵ', δ' in Lemma 4.12 to be $\frac{1}{4}\epsilon, \frac{1}{4}\delta$, we can store the rounded vector in space $O(d \log(\frac{1}{\gamma} \log n)) = O(d(\log \frac{1}{\epsilon\delta} + \log \log n))$.

Next, we prove the correctness of rounding the Difference Estimator, for $p = 2$. The argument is the same as Lemma 4.12.

Lemma 4.13. (*Correctness of rounding the Difference Estimator*) Let $\epsilon_0, \delta_0 \in (0, 1), q = 3, u, v$ be 2 frequent vector, $A \in \mathbb{R}^{d \times U}$ be the sketch matrix, $\gamma = (\frac{\epsilon_0 \delta_0}{dm \log(Um)})^C$, for $p \in (0, 2)$, recall the way Difference Estimator estimate the difference between $\|u + v\|_p^p$ and $\|v\|_p^p$ is by calculate the mean of $\prod_{j=q(i-1)+1}^{qi} |(Au + Av)_j|^{p/q} - \prod_{j=q(i-1)+1}^{qi} |(Av)_j|^{p/q}$. Then the mean of $\prod_{j=q(i-1)+1}^{qi} |(\mathcal{P}(Au + Av))_j|^{p/q} - \prod_{j=q(i-1)+1}^{qi} |(\mathcal{P}(Av))_j|^{p/q}$ only deviate $O((\epsilon_0 \frac{\delta_0^2}{\delta_0^2})^{q/p}) \cdot F(v)$ from the original estimation with probability $\geq 1 - 2\delta_0$.

Thus, setting $\epsilon_0 = O(\frac{\epsilon \delta^2}{d^2})$ and $\delta_0 = O(\delta)$, we can store the each Difference Estimator in space $O(d(\log \frac{1}{\epsilon\delta} + \log \log n))$.

4.4 Nearly Optimal F_p Estimation

Now we introduce our main result for F_p estimation.

Theorem 4.14. Let $p \in [1, 2]$, for any fixed window W , with probability $\geq \frac{2}{3}$, Algorithm 2 will give an ϵ approximation of F_p^W , and use $\tilde{O}(\epsilon^{-p} \log^2 n + \epsilon^{-2} \log n)$ bits of space, more specifically, $O((\epsilon^{-p} \log^2 n + \epsilon^{-2} \log n \log^4 \frac{1}{\epsilon})(\log \frac{1}{\epsilon} + (\log \log n)^2))$.

Let the query window be $W = [r - n, r]$, and let l be the smallest index that $F_p^{(l,r)} \leq 2F_p^W$, and $W' = [l, r]$. First, we prove that our top level Strong Estimator gives a constant approximation. The proof can be easily seen from Lemma 3.7.

Lemma 4.15. (*Constant factor partitions in top level*).

With probability $\geq 1 - \delta$, $F_p^W \leq F_p^{(t_1,r)} \leq 2F_p^W$ and $\frac{1}{2}F_p^W \leq F_p^{(t_2,r)} \leq F_p^W$.

Proof. **TOPROVE 3** □

Let \mathcal{E} be the event that the top level Strong Estimator gives a constant factor partition, and for the subroutine levels, all the $O(\epsilon^{-p} \log n)$ instances of Strong Estimator satisfy the $(\frac{1}{2}, \frac{2^{-1/p} \cdot \epsilon}{2^{30}})$ Strong Estimator property on window W' . In the subroutine level, we use $(\frac{1}{2}, \frac{2^{1/p} \cdot \epsilon}{2^{30}}, \delta_{Sub} < \frac{\delta}{O(\epsilon^p \log n)})$ Strong Estimator, so by a union bound, all the $O(\epsilon^p \log n)$ instances of Strong Estimator satisfy the Strong Estimator property in window W' with probability $\geq 1 - \delta$.

So by adjusting the constant for δ , with probability $\geq 1 - \delta$, \mathcal{E} will happen.

We now show an upper bound on the moment of each substream whose contribution is estimated by the difference estimator, i.e., the difference estimators are well-defined.

Lemma 4.16. (*Upper bounds on splitting times*) Let $p \in [1, 2]$. Conditioned on \mathcal{E} , we have that for each $j \in [\beta]$ and all k , either $t_{1,j,k+1} = t_{1,j,k} + 1$ or $F_p^{(t_{1,j,k}, t_{1,j,k+1})} \leq 2^{-j-7} \cdot F_p^W$.

Lemma 4.17. (*Accuracy of difference estimators*) (Lemma A.3 [WZ22]) Let $p \in [1, 2]$. Conditioned on the event in Lemma 4.16 holding, we have that for each $j \in [\beta]$ and all k ,

SDiffEsT $(t_{i,j,k-1}, t_{i,j,k}, t, \gamma_j, \eta, \delta)$ gives an additive $\eta \cdot F_p(t_{i,j,k}, t)$ approximation to $F_p^{(t_{i,j,k-1}, t)} - F_p^{(t_{i,j,k}, t)}$ with probability at least $1 - \delta$.

Lemma 4.18. (Geometric upper bounds on splitting times) (Lemma A.4 [WZ22]) Let $p \in [1, 2]$. Conditioned on the event in Lemma 4.16 holding, we have that for each $j \in [\beta]$ and each k that either $t_{i,j,k+1} = t_{i,j,k} + 1$ or $F_p^{(t_{i,j,k}, m)} - F_p^{(t_{i,j,k+1}, m)} \leq 2^{-j/p-2} p \cdot F_p^{(t_{i,j,k+1}, m)}$.

Lemma 4.19. (Geometric Lower bounds on splitting times). Let $p \in [1, 2]$, conditioned on \mathcal{E} , we have that for each $j \in [\beta]$ and each k that

$$F_p^{(t_{i,j,k-1}, t_{i,j,k+1})} > 2^{-j-21} \cdot F_p^W.$$

```

1: function STITCHSW( $t$ )
2:    $c_0 \leftarrow t_1$ 
3:    $X \leftarrow g^{(t_1, t)}$  ▷ By Indyk's p-stable Sketch , after rounding
4:   for  $j \in [\beta]$  do ▷ Stitch sketches
5:     Let  $a$  be the smallest index such that  $t_{1,j,a} \geq c_{j-1}$ 
6:     Let  $b$  be the largest index such that  $t_{1,j,b} \leq t - n + 1$ 
7:      $c_j \leftarrow t_{1,j,b}$ 
8:      $Y_j \leftarrow \sum_{k=a}^{b-1} \text{SDiffEst}(1, j, k)$ 
9:   end for
10:  return  $Z := X - \sum_{j=1}^{\beta} Y_j$ 
11: end function

```

Lemma 4.20. (Number of level j difference estimators) (Lemma A.6 [WZ22])

Let $p \in [1, 2]$, conditioned on the Geometric Lower bounds on splitting times to hold (Lemma 4.19), we have that for each $j \in [\beta]$, that $k \leq 2^{j+23}$ for any $t_{1,j,k}$.

We next bound the number of level j difference estimators that can occur from the end of the previous level $j - 1$ difference estimator to the time when the sliding window begins. We say a difference estimator $\mathcal{C}_{1,j,k}$ is active if $k \in [a_j, b_j]$ for the indices a_j and b_j defined in **StitchSW**. The active difference estimators in each level will be the algorithms whose output are subtracted from the initial rough estimate to form the final estimate of F_p^W .

Lemma 4.21. (Number of active level j difference estimators) (Lemma A.7 [WZ22]) Conditioned on the Upper bounds and Geometric lower bounds on splitting times to hold (Lemma 4.16 and 4.19), for $p \in [1, 2]$ and each $j \in [\beta]$, let a_j be the smallest index such that $t_{1,j,a_j} \geq c_{j-1}$ and let b_j be the largest index such that $t_{1,j,b_j} \leq r - n$. Then conditioned on \mathcal{E} , we have $b_j - a_j \leq 2^{24}$.

Lemma 4.22. (Correctness of sliding window algorithm) For $p \in [1, 2]$, Algorithm 2 outputs a $(1 \pm \epsilon)$ -approximation to F_p^W with probability $\geq 1 - \delta$.

Now we will prove the space bound.

Lemma 4.23. Let δ be a constant $\frac{1}{3}$, Algorithm 2 uses space $O((\epsilon^{-p} \log^2 n + \epsilon^{-2} \log n \log^4 \frac{1}{\epsilon})(\log \frac{1}{\epsilon} + (\log \log n)^2))$ bits.

Combining Lemma 4.22, which establishes the approximation guarantee, with the space bound given in Lemma 4.23, we conclude the proof of Theorem 4.14.

5 Lower bounds

We consider the following communication problem, which is a generalization of the classic GreaterThan communication problem.

Problem 5.1. *In the $\text{IndexGreater}(N, k)$ problem, Alice receives a sequence $(x_1, \dots, x_k) \in [2^N]^k$. Bob receives an index j and a number $y \in [2^N]$. Alice sends a one-way message to Bob, and Bob must decide if $y > x_j$ or if $y \leq x_j$.*

Lemma 5.2. *Problem 5.1 requires $\Omega(Nk)$ communication to solve with $2/3$ probability.*

This lower bound is known. For instance [BGL⁺18] applies it in their lower bounds as well. We supply a short proof in the appendix from the somewhat more standard Augmented Index problem.

Lemma 5.3. *Let $p \in (1, 2]$, and let W, N, k, r be parameters satisfying the following:*

- $n \leq U$
- $2^r N k^{1-1/p} \leq n^{1-1/p}$
- $2^r N k \leq n$

Suppose that \mathcal{A} is a streaming algorithm that operates on sliding windows of size n , and gives a $1 \pm \frac{1}{12k^{1/p}}$ multiplicative approximation to F_p on any fixed window with $9/10$ probability, over the universe $[U]$.

Then \mathcal{A} uses at least $\Omega(kr \log N)$ space.

Proof. TOPROVE 4 □

Theorem 5.4. *Fix $p \in (1, 2]$. Suppose that \mathcal{A} is a streaming algorithm that operates on a window of size n , and outputs a $(1 \pm \varepsilon)$ approximation of F_p on any fixed window with $9/10$ probability. Suppose that $\varepsilon \geq n^{-1/p}$. Then \mathcal{A} must use at least*

$$\Omega\left(\frac{(p-1)^2}{\varepsilon^p} \log^2(\varepsilon \min(n, U)^{1/p}) + \frac{1}{\varepsilon^2} \log(\varepsilon^{1/p} \min(U, n))\right)$$

space. In particular for constant p and $n \geq U$, \mathcal{A} must use at least

$$\Omega\left(\frac{1}{\varepsilon^p} \log^2(\varepsilon U) + \frac{1}{\varepsilon^2} \log(\varepsilon^{1/p} U)\right)$$

space.

Proof. TOPROVE 5 □

References

- [AMS96] Noga Alon, Yossi Matias, and Mario Szegedy. The space complexity of approximating the frequency moments. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 20–29, 1996.

- [BCI⁺17] Vladimir Braverman, Stephen R Chestnut, Nikita Ivkin, Jelani Nelson, Zhengyu Wang, and David P Woodruff. Bptree: An ℓ_2 heavy hitters algorithm using constant memory. In *Proceedings of the 36th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, pages 361–376, 2017.
- [BDN17] Jarosław Błasiok, Jian Ding, and Jelani Nelson. Continuous monitoring of ℓ_p norms in data streams. *arXiv preprint arXiv:1704.06710*, 2017.
- [BGL⁺18] Vladimir Braverman, Elena Grigorescu, Harry Lang, David P Woodruff, and Samson Zhou. Nearly optimal distinct elements and heavy hitters on sliding windows. *arXiv preprint arXiv:1805.00212*, 2018.
- [BO07a] Vladimir Braverman and Rafail Ostrovsky. Smooth histograms for sliding windows. In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS'07)*, pages 283–293, 2007.
- [BO07b] Vladimir Braverman and Rafail Ostrovsky. Smooth histograms for sliding windows. In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS'07)*, pages 283–293. IEEE, 2007.
- [BO10] Vladimir Braverman and Rafail Ostrovsky. Effective computations on sliding windows. *SIAM Journal on Computing*, 39(6):2113–2131, 2010.
- [BR94] Mihir Bellare and John Rompel. Randomness-efficient oblivious sampling. In *Proceedings 35th Annual Symposium on Foundations of Computer Science*, pages 276–287. IEEE, 1994.
- [BYJKS04] 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.
- [BZ24] Mark Braverman and Or Zamir. Optimality of frequency moment estimation. *arXiv preprint arXiv:2411.02148*, 2024.
- [CCFC02] Moses Charikar, Kevin Chen, and Martin Farach-Colton. Finding frequent items in data streams. In *International Colloquium on Automata, Languages, and Programming*, pages 693–703. Springer, 2002.
- [DGIM02] Mayur Datar, Aristides Gionis, Piotr Indyk, and Rajeev Motwani. Maintaining stream statistics over sliding windows. *SIAM journal on computing*, 31(6):1794–1813, 2002.
- [DLOM02] Erik D Demaine, Alejandro López-Ortiz, and J Ian Munro. Frequency estimation of internet packet streams with limited space. In *European Symposium on Algorithms*, pages 348–360. Springer, 2002.
- [Ind06] Piotr Indyk. Stable distributions, pseudorandom generators, embeddings, and data stream computation. *Journal of the ACM (JACM)*, 53(3):307–323, 2006.
- [IW05] Piotr Indyk and David Woodruff. Optimal approximations of the frequency moments of data streams. In *Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*, pages 202–208, 2005.

- [JST11] Hossein Jowhari, Mert Sağlam, and Gábor Tardos. Tight bounds for L_p samplers, finding duplicates in streams, and related problems. In *Proceedings of the Thirtieth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '11, pages 49–58, New York, NY, USA, 2011. Association for Computing Machinery.
- [JW23] Rajesh Jayaram and David P. Woodruff. Towards optimal moment estimation in streaming and distributed models. *ACM Trans. Algorithms*, 19(3), June 2023.
- [KNW10] 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. SIAM, 2010.
- [Nol01] James L. Jr. Nolan. Stable distributions. models for heavy tailed data. 2001.
- [SW02] Subhabrata Sen and Jia Wang. Analyzing peer-to-peer traffic across large networks. In *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurement*, pages 137–150, 2002.
- [WZ22] David P Woodruff and Samson Zhou. Tight bounds for adversarially robust streams and sliding windows via difference estimators. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1183–1196. IEEE, 2022.
- [Zol86] Vladimir M Zolotarev. *One-dimensional stable distributions*, volume 65. American Mathematical Soc., 1986.

6 Appendix

6.1 Upper Bound Lemmas

Proof of Lemma 3.2.

Proof. [TOPROVE 6](#) □

Proof of Lemma 3.3.

Proof. [TOPROVE 7](#) □

Proof of Lemma 3.4.

Proof. [TOPROVE 8](#) □

Proof of Lemma 3.7.

Proof. [TOPROVE 9](#) □

Proof of Lemma 4.7

Proof. [TOPROVE 10](#) □

Proof of Lemma 4.11.

Proof. [TOPROVE 11](#) □

Proof of Lemma 4.13.

Proof. [TOPROVE 12](#) □

Proof of Lemma 4.16.

Proof. [TOPROVE 13](#) □

Proof of Lemma 4.19.

Proof. [TOPROVE 14](#) □

Proof of Lemma 4.22.

Proof. [TOPROVE 15](#) □

Proof of Lemma 4.23.

Proof. [TOPROVE 16](#) □

6.2 Lower Bounds

Proof of Lemma 5.2

Proof. [TOPROVE 17](#) □

Proof of Lemma 5.3.

Proof. **TOPROVE 18**

□