

Differentially Private Continual Releases of Streaming Frequency Moment Estimations

Alessandro Epasto ✉

Google, New York, NY, USA

Jieming Mao ✉

Google, New York, NY, USA

Andres Munoz Medina ✉

Google, New York, NY, USA

Vahab Mirrokni ✉

Google, New York, NY, USA

Sergei Vassilvitskii ✉

Google, New York, NY, USA

Peilin Zhong ✉

Google, New York, NY, USA

Abstract

The streaming model of computation is a popular approach for working with large-scale data. In this setting, there is a stream of items and the goal is to compute the desired quantities (usually data statistics) while making a single pass through the stream and using as little space as possible.

Motivated by the importance of data privacy, we develop differentially private streaming algorithms under the continual release setting, where the union of outputs of the algorithm at every timestamp must be differentially private. Specifically, we study the fundamental ℓ_p ($p \in [0, +\infty)$) frequency moment estimation problem under this setting, and give an ε -DP algorithm that achieves $(1 + \eta)$ -relative approximation ($\forall \eta \in (0, 1)$) with $\text{poly} \log(Tn)$ additive error and uses $\text{poly} \log(Tn) \cdot \max(1, n^{1-2/p})$ space, where T is the length of the stream and n is the size of the universe of elements. Our space is near optimal up to poly-logarithmic factors even in the non-private setting.

To obtain our results, we first reduce several primitives under the differentially private continual release model, such as counting distinct elements, heavy hitters and counting low frequency elements, to the simpler, counting/summing problems in the same setting. Based on these primitives, we develop a differentially private continual release level set estimation approach to address the ℓ_p frequency moment estimation problem.

We also provide a simple extension of our results to the harder sliding window model, where the statistics must be maintained over the past W data items.

2012 ACM Subject Classification Theory of computation \rightarrow Streaming, sublinear and near linear time algorithms; Security and privacy

Keywords and phrases Differential Privacy, Continual Release, Sliding Window, Streaming Algorithms, Distinct Elements, Frequency Moment Estimation

Digital Object Identifier 10.4230/LIPIcs.ITCS.2023.48

Related Version *Full Version*: <https://arxiv.org/abs/2301.05605>

1 Introduction

Data privacy is a central concern in the deployment of real-world computational systems. In the vast literature on privacy in computation [27], the notion of differential privacy (DP) [18, 22] has remained the *de facto* standard for more than a decade. The classical formulation of differential privacy assumes that the data is static [18], and that the data curator is interested obtaining answers to a *predetermined* number of queries on the dataset.



© Alessandro Epasto, Jieming Mao, Andres Munoz Medina, Vahab Mirrokni, Sergei Vassilvitskii, and Peilin Zhong;

licensed under Creative Commons License CC-BY 4.0

14th Innovations in Theoretical Computer Science Conference (ITCS 2023).

Editor: Yael Tauman Kalai; Article No. 48; pp. 48:1–48:24



Leibniz International Proceedings in Informatics

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Real world applications, however, often require the analysis of user datasets that are organically and rapidly growing. This is illustrated by the popular *streaming model* of computation [47, 2], where data arrives over time, and at each update, a new solution is output by the algorithm. In such streaming applications, the *continual release* model of differential privacy [19, 28] promises a rigorous guarantee of privacy: an observer of *all* the outputs of the algorithm is information-theoretically bounded in the ability to learn about the *existence* of an individual data point in the stream.

In this paper, we focus on two fundamental challenges in the field of private streaming algorithms: the insertion only, or streaming, model and the sliding window model. In the former model, a data curator receives a stream of data a_1, a_2, \dots and at each time t releases a statistical query depending on all data received up to that point. In the latter, the computation depends only on the last W items observed by the data curator.

The sliding window model may be generally more practically relevant compared to the streaming model as it allows to account for information freshness and in some cases it can be a legal or privacy requirement as well. For instance, in some situations, data privacy laws such as the General Data Protection Regulation (GDPR)¹ do not allow unlimited retention of user data.

Our main contribution is to provide private algorithms for a series of foundational streaming problems under both the streaming and sliding window model.

Motivating example: Privacy Sandbox. We present a concrete practical application of our results. As part of the *Privacy Sandbox* initiative, Chrome has developed a series of APIs to reduce cross site tracking while supporting the digital advertising ecosystem. A key part of one of the proposals is a *k-Anonymity Server*.² The server ensures that each ad creative that is reported to advertisers has won its respective auction at least k times over a particular time window. Abstracting the specifics, this problem requires computing the number of distinct elements over a sliding window. Moreover, to further strengthen privacy protections, the computation itself should be made differentially private, which is precisely the setting we consider in this work.

The previous example elucidates a concrete motivation for the study of sliding window algorithms for counting distinct element problems with differential privacy in the continual release setting. The rest of the paper proceeds by formalizing this model and our results in this space.

In particular, we study a more general class of statistics of the input data than the problem of counting distinct elements: the ℓ_p frequency moments problem. The ℓ_p frequency moment is the sum of the p -th power of the frequencies of the elements. The number of distinct elements is a special case for $p = 0$. The ℓ_p frequency moment problem is one of the most fundamental problems in the streaming literature. The first non-trivial algorithm for $p = 1$ is [40]. Later [25] is the first to study the case for $p = 0$. [2] initiates the study for $p = 2$ and other $p \in [0, \infty)$. After the developments over several decades, the spaces of the current best ℓ_p frequency moment estimation algorithms are near optimal for all $p \in [0, \infty)$, i.e. they almost match the proven space lower bounds (see e.g., [26, 34, 35, 24, 37]).

However the landscape of DP streaming ℓ_p frequency moment is mysterious even in the non-continual release setting. Most existing work only studied for $p = 0, 1, 2$ (see more discussion in Section 1.4). The work of [50] considered general $p \in (0, 1]$. A recent independent

¹ <https://gdpr-info.eu/art-17-gdpr/>

² https://github.com/WICG/turtledove/blob/main/FLEDGE_k_anonymity_server.md

work [8] studied all $p \in [0, \infty)$. But none of [8, 50] considered continual release setting. In the DP continual release setting, [9] studied the count of distinct elements but not in the low space streaming setting. In the DP streaming continual release setting, existing work [18, 28] only studied the case for $p = 1$. No previous algorithm for $p \neq 1$ is known in the DP low space streaming continual release setting.

For ℓ_p frequency moment in the sliding window model, there are known techniques [16, 10] which can convert the streaming algorithm into sliding window algorithm using some additional small space. We show how to extend these techniques to convert our DP streaming continual release streaming algorithms to DP sliding window continual release algorithms.

1.1 Computational Model

In this paper, we consider a streaming setting with T timestamps. At each timestamp $t \in [T]$, we get an input $a_t \in \mathcal{U} \cup \{\perp\}$, where \mathcal{U} represents the universe of all possible input elements, and \perp represents empty. We sometimes also consider an input stream of integers, i.e., at each timestamp $t \in [T]$, we get an input $a_t \in \mathbb{Z}$. The goal is to compute some function $g(\cdot)$ based on the inputs. Instead of only computing g at the end of the stream, we consider the continual release model throughout the paper:

- **Continual Release Model:** At every timestamp $t \in [T]$, we want to output $g(\cdot)$ based on the data a_1, a_2, \dots, a_t .

We consider two different streaming models depending on the range of inputs that g is based on.

- **(Insertion-only) Streaming Model:** In this model, g depends on all past inputs, i.e. at timestamp $t \in [T]$, we want to compute $g(a_1, \dots, a_t)$. Unless otherwise specified, we use streaming model to refer to insertion-only streaming model throughout the paper.
- **Sliding Window Model:** In this model, we have a parameter $W \in \mathbb{Z}_{\geq 1}$ for window size. g depends on the last W inputs, i.e. at timestamp $t \in [T]$, we want to compute $g(a_{\max(t-W+1, 1)}, \dots, a_t)$.

In this paper, we are particularly interested in algorithms that use space sub-linear in T . For a (sub-)stream \mathcal{S} , we use $\|\mathcal{S}\|_p^p$ to denote the ℓ_p frequency moment of \mathcal{S} . In particular, for $p = 0$, $\|\mathcal{S}\|_0$ denotes the number of distinct elements. For $p = 1$, $\|\mathcal{S}\|_1$ denotes the number of non-empty elements. We refer readers to preliminaries section (Section 2) for detailed notation and definitions.

1.2 Our Results and Comparison to Prior Work

In this section, we give a brief overview of our results and comparison with prior work. We use n to denote the size of the universe \mathcal{U} . We use (α, γ) -approximation to specify the approximation guarantee with multiplicative factor α and additive error γ . In all of our results, we use $\varepsilon \geq 0$ for the DP parameter, and use $\eta \in (0, 0.5)$ in the relative error.

1.2.1 Differentially Private Streaming Continual Release Algorithms

We developed a series of DP algorithms for solving frequency moments estimation and its related problems in the streaming continual release model.

ℓ_p Frequency moment estimation ($p \in [0, \infty)$). Our main result is a general ℓ_p frequency moment estimation algorithm which works for all $p \in [0, \infty)$.

► **Theorem 1** (ℓ_p Frequency moment, informal version of Theorem 49). *There is an ε -DP algorithm in the streaming continual release model such that with probability at least 0.9, it always outputs an $\left(1 + \eta, \left(\frac{\log(Tn)}{\eta\varepsilon}\right)^{O(\max(1,p))}\right)$ -approximation to $\|\mathcal{S}\|_p^p$ for every timestamp t , where \mathcal{S} denotes the stream up to timestamp t . The algorithm uses space $\max(1, n^{1-2/p}) \cdot \left(\frac{\log(Tn)}{\eta\varepsilon}\right)^{O(\max(1,p))}$.*

To the best of our knowledge, we are the first to study the general ℓ_p frequency moment estimation problem in the differentially private streaming continual release setting. [19] and [28] studies the summing problem in the same setting, where the summing problem can be seen as a special case for $p = 1$. [50] studies the streaming ℓ_p frequency moment estimation for $p \in (0, 1]$ based on the p -stable distribution, and a concurrent independent work [8] studies the case for $p \in (0, 1]$, but it is not clear how to generalize their techniques to the continual release model, i.e., their approach only provides the differential privacy guarantee of the output at the end of the stream. In addition, the approach of [50] does not achieve ε -DP for an arbitrarily small $\varepsilon > 0$, and they also mention that their technique might not be easily extended to the case for $p > 1$.

Our space usage is near optimal up to poly-logarithmic factors even when comparing with the non-private streaming ℓ_p frequency moment estimation algorithms: for $p \leq 2$, the space needed for both our algorithm and previous non-private algorithm (see e.g., [35]) is poly-logarithmic, for $p > 2$, the space needed for both our algorithm and previous non-private algorithm [30] is $\tilde{O}(n^{1-2/p})^3$. Note that $\Omega(n^{1-2/p})$ space is a proven lower bound for $p > 2$ even in the non-private case [42, 5].

Summing (ℓ_1 frequency moment estimation). The easiest problem that is related to the ℓ_p frequency moment estimation problem would be the summing problem: the goal is to compute the summation of the input numbers. Note that ℓ_1 frequency moment estimation is a special case of the summing of a binary stream, i.e., we regard \perp as 0 and all other elements as 1.

► **Theorem 2** (Summing of a non-negative stream, informal version of Theorem 15). *There is an ε -DP algorithm for summing problem in the streaming continual release model. If the input numbers are guaranteed to be non-negative, with probability at least 0.9, the output is always a $(1 + \eta, O_{\varepsilon,\eta}(\log T))$ -approximation to the sum of all input numbers at any timestamp $t \in [T]$. The algorithm uses space $O(1)$.*

The summing problem was studied by [19, 28] in the differentially private streaming continual release model. Their approximation has $O(\log^{2.5} T)$ additive error. In our work, we show that if we allow $(1 + \eta)$ relative error and work on the stream with non-negative numbers only, we can reduce the additive error to $O(\log T)$. This is useful when we cannot avoid the relative error for some problem (such as the number of distinct elements) in the streaming model but we still need summing as a subroutine.

Counting distinct elements (ℓ_0 frequency moment estimation). Counting distinct elements is one of the fundamental problems in the streaming literature. The goal is to estimate the number of distinct elements that appeared in the stream. We provide a DP streaming continual release algorithm for counting distinct elements.

³ We use $\tilde{O}(g)$ to denote $g \cdot \text{poly}(\log(g))$.

► **Theorem 3** (Number of distinct elements, informal version of Corollary 24). *There is an ε -DP algorithm for the number of distinct elements in the streaming continual release model. With probability at least 0.9, the output is always a $(1 + \eta, O_{\varepsilon, \eta}(\log^2(T)))$ -approximation for every timestamp $t \in [T]$. The algorithm uses $\text{poly}\left(\frac{\log(T)}{\eta \min(\varepsilon, 1)}\right)$ space.*

In the non-private streaming setting, counting distinct element can be solved via sketching algorithms of [25] and its variants e.g., [24]. Some recent work [14, 45] extends these sketching techniques for counting distinct element in a DP streaming setting. However, it is not clear how to extend these techniques to the continual release setting. Continual release of counts of distinct elements is studied by [9]. However, [9] is not in the low space streaming setting.

Estimation of frequencies and ℓ_2 frequency moments. The goal of ℓ_2 frequency moment estimation is to estimate the sum of square of frequencies of elements. We present a DP streaming continual release CountSketch [13] algorithm and use it for estimating ℓ_2 frequency moments and the frequency of each element.

► **Theorem 4** (Frequency and ℓ_2 frequency moments, informal version of Theorem 28). *There is an ε -DP algorithm in the streaming continual release model such that with probability at least 0.9, it always outputs for every timestamp $t \in [T]$:*

1. \hat{f}_a for every $a \in \mathcal{U}$ such that $|f_a - \hat{f}_a| \leq \eta \|\mathcal{S}\|_2 + \tilde{O}_{\varepsilon, \eta}(\log^{3.5}(Tn))$, where \mathcal{S} denotes the stream up to timestamp t and f_a denotes the frequency of a in \mathcal{S} ,
2. \hat{F}_2 such that $|\hat{F}_2 - \|\mathcal{S}\|_2^2| \leq \eta \|\mathcal{S}\|_2^2 + \tilde{O}_{\varepsilon, \eta}(\log^7(Tn))$

The algorithm uses $O\left(\frac{\log(Tn)}{\eta^2} \cdot \log(T)\right)$ space.

Although DP ℓ_2 frequency moment was studied by a line of work (see e.g., [7, 43, 11]), none of them considers the streaming continual release setting, and it is not clear how to extend previous techniques to the the continual release setting.

ℓ_p Heavy hitters. In the ℓ_p heavy hitters problem, we are given a parameter k , and the goal is to find elements whose frequency to the p -th power is at least $1/k$ fraction of the ℓ_p frequency moment. By extending our DP streaming continual release CountSketch algorithm, we obtain a DP streaming continual release ℓ_p heavy hitters algorithm.

► **Theorem 5** (ℓ_p Heavy hitters for all $p \in [0, \infty)$, informal version of Theorem 34). *There is an ε -DP algorithm in the streaming continual release model such that with probability at least 0.9, it always outputs a set $H \subseteq \mathcal{U}$ and a function $\hat{f} : H \rightarrow \mathbb{R}$ for every timestamp $t \in [T]$ satisfying*

1. $\forall a \in H, \hat{f}(a) \in (1 \pm \eta) \cdot f_a$ where f_a is the frequency of a in the stream \mathcal{S} up to timestamp t ,
2. $\forall a \in \mathcal{U}$, if $f_a \geq \frac{1}{\varepsilon \eta} \cdot \text{poly}\left(\log\left(\frac{T \cdot k \cdot n}{\eta}\right)\right)$ and $f_a^p \geq \|\mathcal{S}\|_p^p / k$ then $a \in H$,
3. The size of H is at most $O(\log(Tn) \cdot 2^p \cdot k)$.

The algorithm uses $\max(1, n^{1-2/p}) \cdot \frac{k^3}{\eta^2} \cdot \text{poly}(\log(T \cdot k \cdot n))$ space.

To the best of our knowledge, though DP streaming continual release ℓ_1 heavy hitters problem is studied by [12], ℓ_p (for $p \neq 1$) heavy hitters problem has not been studied in the DP streaming continual release setting before. Note that $\Omega(n^{1-2/p})$ for $p > 2$ is a lower bound of space needed for ℓ_p heavy hitters even in the non-private setting [42, 5].

1.2.2 Differentially Private Sliding Window Continual Release Algorithms

Smooth histogram [10] is a general algorithmic framework which can convert a relative-approximate streaming algorithm into a relative-approximate sliding window algorithm if the objective function that we want to compute has some nice properties.

We generalize the smooth histogram to make it support converting an approximate streaming algorithm with both relative error and additive error into an approximate sliding window algorithm with both relative error and additive error if the objective function has good properties. In addition, we show that if the streaming algorithm is DP in the continual release setting, then the converted sliding window algorithm is also DP in the continual release setting.

By applying our generalized smooth histogram approach and paying a poly $\left(\frac{\log T}{\eta}\right)$ more factor than our DP streaming continual release algorithms in both additive error and space usage, we show DP sliding window continual release algorithms for

1. ℓ_p Moment estimation (see Corollary 53),
2. Summing (see Corollary 50),
3. Counting distinct elements (see Corollary 51),
4. ℓ_2 Moment estimation (see Corollary 52).

1.3 Our Techniques

In this section, we briefly discuss the high level ideas of our algorithms. We present a set of techniques to reduce almost all problems that we considered in the DP streaming continual release setting to the summing problem in the DP streaming continual release setting.

Summing with better additive error via grouping. To illustrate the intuition of using grouping for differentially private streaming continual release algorithms, we start with the following simple problem: given a stream of numbers c_1, c_2, \dots, c_T where each c_i is at least $10 \cdot \ln(10 \cdot T)/\varepsilon$, the goal is to output a (1 ± 0.1) -approximation to $\sum_{j=1}^t c_j$ for every prefix t with probability at least 0.9, and we want the set of all outputs to be ε -DP, i.e., the continual released results to be ε -DP. A simple way to solve the above problem is that we release a stream of noisy numbers $\hat{c}_1, \hat{c}_2, \dots, \hat{c}_T$ where $\forall t \in [T], \hat{c}_t = c_t + \text{Lap}(1/\varepsilon)$, and we report $\sum_{j=1}^t \hat{c}_j$ for every prefix $t \in [T]$. It is easy to see that $(\hat{c}_1, \hat{c}_2, \dots, \hat{c}_T)$ is ε -DP. Since the reported approximate prefix sums only depend on $(\hat{c}_1, \hat{c}_2, \dots, \hat{c}_T)$, the continual released results are ε -DP. Furthermore, with probability at least 0.9, $\forall t \in [T], |c_t - \hat{c}_t| \leq \ln(10 \cdot T)/\varepsilon$. Since c_t is at least $10 \ln(10 \cdot T)/\varepsilon$, we have $0.9c_t \leq \hat{c}_t \leq 1.1c_t$ which implies that every reported approximate prefix sum is a (1 ± 0.1) -approximation.

To generalize the above idea, we propose a grouping approach in to group the consecutive numbers in the stream in a differentially private way such that the total count of each group is large enough. To implement grouping, we need to apply the sparse vector technique (see e.g., [22]) iteratively. The similar idea also appeared in [21] which shows a better additive error guarantee for the summing problem than [19] when the stream is sparse. In contrast, our additive error guarantee is always better than [19] and [21] while we allow an additional $(1 + \varepsilon)$ relative approximation.

Counting distinct elements. We explain how to reduce counting distinct elements problem to the summing problem. Suppose the element universe is small, we are able to track the set of elements that already appeared during the stream. Then, we can create a binary stream

of $\{0, 1\}$ where 1 denotes that we see a new element and 0 denotes that the input element already appeared or it is empty. Therefore, the sum of the binary stream at timestamp t is exactly the number of distinct elements. Furthermore, if we change an element in the input stream from a to b , there are only constant number of positions of the binary stream will flip: consider the change $a \rightarrow \perp \rightarrow b$. If a is not its first appearance in the input stream, changing a to \perp does not cause any change in the binary stream. If a is its first appearance in the input stream, changing a to \perp will make the corresponding 1 in the binary stream be 0 and make the 0 corresponding to the original second appearance of a in the input stream to be 1. Thus, it will affect at most 2 entries of the binary stream. Similarly, changing \perp to b will cause the change of at most 2 entries of the binary stream. Thus, the binary stream has low sensitivity which implies that a DP streaming continual release summing algorithm gives a good approximation to the number of distinct elements with a small additive error. Next, we discuss how to handle the large universe. For large universe, we can try different sampling rate $1/2, 1/4, 1/8 \dots, 1/T$. There should be a sampling rate such that (1) if we hash the sampled elements into hashing buckets, there is no collision with a good probability, (2) the number of samples is much larger than the additive error caused by the summing subroutine so we can have a good relative approximation of the number of distinct sampled elements. Then we can use the number of distinct sampled elements to estimate the number of distinct elements in the input stream.

CountSketch and ℓ_p heavy hitters. Let $h : \mathcal{U} \rightarrow [k]$ be a hash function which uniformly hash elements into k hash buckets. Let $g : \mathcal{U} \rightarrow \{-1, 1\}$ randomly map each element to -1 or 1 with equal probability. The CountSketch is a tuple of k numbers (z_1, z_2, \dots, z_k) where z_i is the sum of weighted frequencies of elements hashed to the bucket i , and the weight of the frequency of element a is $g(a)$. Changing a to b in the input stream will change at most 2 buckets: the bucket i contains a and the bucket j contains b . Since $|g(a)| \leq 1$, z_i and z_j will be changed by at most 1. We can use DP streaming continual release summing algorithm of [19, 28] to estimate (z_1, z_2, \dots, z_k) such that each estimation \hat{z}_i of z_i only has $\text{poly}(\log T)$ additive error, and $(\hat{z}_1, \hat{z}_2, \dots, \hat{z}_k)$ is DP under the streaming continual release model. Suppose the ℓ_2 frequency moment is much larger than $\text{poly}(\log T)$, then the additive error becomes the relative error, and we can use $\hat{z}_1, \hat{z}_2, \dots, \hat{z}_k$ to obtain a good relative approximation of the ℓ_2 frequency moment. Similarly, if an element has frequency much larger than $\text{poly}(\log(T))$, then $\text{poly}(\log(T))$ becomes small relative error of the frequency and we are able to check whether it is an ℓ_2 heavy hitter by the standard analysis of CountSketch. Thus, we can use this DP streaming continual release CountSketch to estimate ℓ_2 frequency moment with $(1 + \eta)$ -relative error and $\text{poly}(\log T)$ -additive error, and we can use such CountSketch to find all elements which are at least $\text{poly}(\log T)$ and are ℓ_2 heavy hitters.

Note that for $p \leq 2$, if a has the largest frequency and it is an $(1/k)$ - ℓ_p heavy hitter, then a must be an $(1/k)$ - ℓ_2 heavy hitter. For $p > 2$, if a is an $1/k$ - ℓ_p heavy hitter, then a must be an $1/(kn^{1-2/p})$ - ℓ_2 heavy hitter. Therefore, by some hashing technique, we can use ℓ_2 heavy hitters algorithm to construct ℓ_p heavy hitters algorithm. But since ℓ_2 heavy hitters can only report the elements with frequency larger than $\text{poly}(\log T)$, the obtained ℓ_p heavy hitters algorithm can only report the elements with frequency larger than $\text{poly}(\log T)$ as well.

ℓ_p Frequency moment estimation. In high level we want to simulate the level set estimation idea of [30] in the DP streaming continual release setting. In particular, let $\alpha = 1 + \eta$, let f_a denote the frequency of a and let $G_i = \{a \mid f_a \in (\alpha^i, \alpha^{i+1}]\}$. Then $\sum_i |G_i| \cdot (\alpha^i)^p$ is a good approximation to the ℓ_p frequency moment. We say G_i is contributing, if $|G_i| \cdot (\alpha^i)^p$ is at least

$\Omega_\alpha(1/\log(T))$ fraction of the ℓ_p moment. Since non-contributing elements only contributes a small total amount to the ℓ_p frequency moment, it is easy to see that $\sum_{\text{contributing } G_i} |G_i| \cdot (\alpha^i)^p$ is still a good approximation to the ℓ_p frequency moment. Thus, we only need to estimate the size of each contributing G_i . Due to the definition of contributing, it is easy to see that if G_i is contributing, either α^i is large or $|G_i|$ is large. In fact, as observed by [30], for each contributing level set G_i , there must be a proper sampling probability such that after sampling, there are at least $\text{poly}(\log T)$ elements from G_i sampled and all of the sampled elements from G_i are at least $1/\text{poly}(\log T)$ - ℓ_p heavy hitters among the set of all sampled elements from the universe \mathcal{U} . Ideally, we can try different sampling rate $1, 1/2, 1/4, 1/8, \dots, 1/T$ and use our ℓ_p heavy hitters algorithm to report the heavy hitters and estimate $|G_i|$ for each i . However, for i with $\alpha^i \ll \text{poly}(\log T)$, our DP streaming continual release ℓ_p heavy hitters algorithm does not report any element from G_i . We must find another way to estimate $|G_i|$ instead of using heavy hitters.

Similar to counting distinct elements, let us start with the case that the universe size is small so we can track the sets A_1, A_2, \dots, A_k for some $k = \text{poly}(\log T)$, where A_i is the set of all elements whose frequency is exactly i . We can construct streams $\mathcal{S}_1, \mathcal{S}_2, \dots, \mathcal{S}_k$ with numbers in $\{-1, 0, 1\}$. During the stream, when we see an input element a , and if a is in the set A_i , then we move a to the set A_{i+1} due to the increase of the frequency of a . At the same time, we append -1 to the stream \mathcal{S}_i , append 1 to the stream \mathcal{S}_{i+1} and append 0 to \mathcal{S}_j for $j \neq i, i+1$. It is easy to check that the sum of \mathcal{S}_l is always the same as $|A_l|$, i.e., the number of elements which have frequency exactly l . Furthermore, similar to the analysis for counting distinct elements, if we change an element in the input stream, each \mathcal{S}_l might be affected by at most 4 entries. Thus, the total sensitivity of $(\mathcal{S}_1, \mathcal{S}_2, \dots, \mathcal{S}_k)$ is at most $O(k)$. Therefore, we can use the DP continual release summing to estimate the sum of each \mathcal{S}_l with additive error $\text{poly}(\log T)$. Thus, we can estimate $|G_i|$ for each i with $\alpha^i \ll \text{poly}(\log T)$ with additive error $\text{poly}(\log T)$ and will only introduce at most $\text{poly}(\log T)$ additive error in approximating the ℓ_p frequency moment.

Now let us go back to the case that the size of the universe is large. In this case, we can use the similar hashing and subsampling technique discussed for counting distinct elements to estimate $|A_l|$ for each $l \in [k]$.

1.4 Related Work

[19] and [28] initiated the study of differential privacy in the continual release model, and proposed the binary tree mechanism for computing summations. [9] and [41] generalized their results to decayed summations, counting distinct elements without space constraints and summations with real-valued data. [46, 23] studied graph problems under the differentially private continual release model. [32, 44, 1] studied differentially private online learning. [31] gave the first polynomial separation in terms of error between the continual release model and the batch model under differential privacy. [49] studied heavy hitters in the differentially private sliding window model.

Differentially private frequency moment estimation for $p = 0, 1, 2$ (without continual releases) has been well-studied [38, 20, 7, 43, 14, 45, 11]. [50] studied frequency moment estimation (without continual releases) for $p \in (0, 1]$ with low space complexity. Recent concurrent independent work [8] studies $p \in [0, \infty)$ with low space complexity but not in continual release setting as well. The differentially private ℓ_1 heavy hitters problem is studied by [38, 20] in the low space streaming setting but not in the continual release setting. [12] studied differentially private ℓ_1 heavy hitters problem in the low space continual release streaming setting. But it is not clear how to extend their techniques to ℓ_p case for $p \neq 1$.

ℓ_p Frequency moment estimation and ℓ_p heavy hitters are heavily studied in the non-private streaming literature. For ℓ_p frequency moment estimation, the problem can be solved by e.g. [25, 24, 17] for $p = 0$, [2, 13, 48] for $p = 2$, [35, 29, 36, 34] for $p \in (0, 2)$ and [30, 4, 3] for $p > 2$. For ℓ_p heavy hitters, the problem can be solved by e.g., [15, 39] for $p = 1$, [13] for $p = 2$, [33] for $p \in (0, 2)$, and [30, 4] for $p > 2$.

2 Preliminaries

2.1 Notation

In this paper, for $n \geq 1$, we use $[n]$ to denote the set $\{1, 2, \dots, n\}$. If there is no ambiguity, for $i \leq j \in \mathbb{Z}$, we sometimes use $[i, j]$ to denote the set of integers $\{i, i + 1, \dots, j\}$ instead of the set of real numbers $\{a \in \mathbb{R} \mid i \leq a \leq j\}$. We use $f_a(a_1, a_2, \dots, a_k)$ to denote the frequency of a in the sequence (a_1, a_2, \dots, a_k) , i.e., $f_a(a_1, a_2, \dots, a_k) = |\{i \in [k] \mid a_i = a\}|$. If the sequence (a_1, a_2, \dots, a_k) is clear in the context, we use f_a to denote the frequency of a for short. We use $\mathbf{1}_{\mathcal{E}}$ to denote a indicator, i.e., $\mathbf{1}_{\mathcal{E}} = 1$ if condition \mathcal{E} holds and $\mathbf{1}_{\mathcal{E}} = 0$ if condition \mathcal{E} does not hold.

For $\alpha \geq 1, \gamma \geq 0$, if $\frac{1}{\alpha} \cdot x - \gamma \leq y \leq \alpha \cdot x + \gamma$, then y is an (α, γ) -approximation to x . If y is a $(1, \gamma)$ -approximation to x , we say y is an approximation to x with additive error γ . If y is an $(\alpha, 0)$ -approximation to x , we say y is an α -approximation to x . We use $a \pm b$ to denote the real number interval $[a - |b|, a + |b|]$. For a set S of real numbers, we use $S \pm b$ to denote the set $\bigcup_{a \in S} a \pm b$, and use $S \cdot c$ to denote the set $\bigcup_{a \in S} a \cdot c$. We use $\text{Lap}(b)$ to denote the Laplace distribution with scale b , i.e., $\text{Lap}(b)$ has density function given by $\frac{1}{2b} \exp(|x|/b)$

2.2 Functions to Compute

We study several fundamental functions in the streaming literature. When the inputs are integers, we consider the summing problem over a (sub-)stream (a_i, \dots, a_j) :

- Sum of numbers: $\mathbf{Sum}(a_i, \dots, a_j) := \sum_{k=i}^j a_k$

When the inputs are from $\mathcal{U} \cup \{\perp\}$, we consider the functions $g(a_i, \dots, a_j)$ that are based on the frequencies of the elements in a (sub-)stream (a_i, \dots, a_j) .

- Count of non-empty elements: $\|(a_i, \dots, a_j)\|_1 := \sum_{a \in \mathcal{U}} f_a(a_i, \dots, a_j)$.
- The number of distinct elements: $\|(a_i, \dots, a_j)\|_0 := \sum_{a \in \mathcal{U}} \mathbf{1}_{f_a(a_i, \dots, a_j) > 0}$.
- ℓ_p -Frequency moment: $\|(a_i, \dots, a_j)\|_p^p := \sum_{a \in \mathcal{U}} f_a(a_i, \dots, a_j)^p$.
- ℓ_p -Heavy hitters: $(1/k)$ - ℓ_p - $\mathbf{HH}(a_i, \dots, a_j) := \{a \in \mathcal{U} \mid f_a(a_i, \dots, a_j)^p \geq \|(a_i, \dots, a_j)\|_p^p / k\}$.

Note that $\|(a_i, \dots, a_j)\|_1$ is a special case of $\mathbf{Sum}(a_i, \dots, a_j)$ with binary inputs.

2.3 Differential Privacy

Neighboring streams: Consider two streams $\mathcal{S} = (a_1, a_2, \dots, a_T)$ and $\mathcal{S}' = (a'_1, a'_2, \dots, a'_T)$. If there is at most one timestamp $t \in [T]$ such that (1). $|a_t - a'_t| \leq 1$ (only required when the inputs are treated as integers) (2). $\forall i \neq t, a_i = a'_i$, then we say \mathcal{S} and \mathcal{S}' are neighboring streams.

► **Definition 6** (Differential Privacy). *We say algorithm A is ϵ -DP, if for any two neighboring streams $\mathcal{S}, \mathcal{S}'$, and any output set \mathcal{O} ,*

$$\Pr[A(\mathcal{S}) \in \mathcal{O}] \leq e^\epsilon \cdot \Pr[A(\mathcal{S}') \in \mathcal{O}].$$

Note that in the continual release model, the output $A(\mathcal{S})$ mentioned in Definition 6 is the entire output history of the algorithm A over stream \mathcal{S} at every timestamp.

► **Definition 7** (Distance between streams). *Consider two streams \mathcal{S} and \mathcal{S}' . If d is the minimum number such that there exists a sequence of streams $\mathcal{S}_0, \mathcal{S}_1, \dots, \mathcal{S}_d$ where $\mathcal{S}_0 = \mathcal{S}, \mathcal{S}_d = \mathcal{S}'$ and $\forall i \in [d], \mathcal{S}_i$ and \mathcal{S}_{i-1} are neighboring streams, then the distance between \mathcal{S} and \mathcal{S}' is $\text{dist}(\mathcal{S}, \mathcal{S}') = d$.*

► **Definition 8** (Sensitivity of a stream mapping). *Let \mathcal{F} be a mapping which maps a given input stream \mathcal{S} to a tuple of streams $(\mathcal{F}_1(\mathcal{S}), \mathcal{F}_2(\mathcal{S}), \dots, \mathcal{F}_k(\mathcal{S}))$. The sensitivity of \mathcal{F} is the minimum value s such that for any two neighboring streams \mathcal{S} and \mathcal{S}' , $\sum_{i \in [k]} \text{dist}(\mathcal{F}_i(\mathcal{S}), \mathcal{F}_i(\mathcal{S}')) \leq s$.*

► **Theorem 9** (Composition [22]). *Let \mathcal{F} be a mapping which maps a given input stream \mathcal{S} to a tuple of streams $(\mathcal{F}_1(\mathcal{S}), \mathcal{F}_2(\mathcal{S}), \dots, \mathcal{F}_k(\mathcal{S}))$. Let A_1, A_2, \dots, A_k be k ε -DP algorithms. Let A be an algorithm such that $A(\mathcal{S}) = M(A_1(\mathcal{F}_1(\mathcal{S})), A_2(\mathcal{F}_2(\mathcal{S})), \dots, A_k(\mathcal{F}_k(\mathcal{S})))$ for some function $M(\cdot)$. Then the algorithm A is $(s\varepsilon)$ -DP.*

Example usage of Theorem 9: Some example usage of Theorem 9 in our paper are presented as the following:

- **Composition of multiple algorithms over the input stream:** Suppose each of A_1, A_2, \dots, A_k is ε -DP, then for any function $M(\cdot)$, $M(A_1(\mathcal{S}), A_2(\mathcal{S}), \dots, A_k(\mathcal{S}))$ is $(k\varepsilon)$ -DP.
- **Composition of algorithms on disjoint sub-streams:** For an input stream $\mathcal{S} = (a_1, a_2, \dots, a_T)$, we partition \mathcal{S} into $\mathcal{S}_1 = (a_{1,1}, a_{1,2}, \dots, a_{1,T}), \mathcal{S}_2 = (a_{2,1}, a_{2,2}, \dots, a_{2,T}), \dots, \mathcal{S}_k = (a_{k,1}, a_{k,2}, \dots, a_{k,T})$, i.e., $\forall t \in [T]$, there is only one $i \in [k]$ such that $a_{i,t} = a_t$, and $\forall i' \neq i, a_{i',t} = \perp$ (or 0). It is clear that such partitioning has sensitivity 1. Thus, if each of A_1, A_2, \dots, A_k is an ε -DP algorithms, then for any function $M(\cdot)$, $M(A_1(\mathcal{S}_1), A_2(\mathcal{S}_2), \dots, A_k(\mathcal{S}_k))$ is ε -DP.

2.4 Streaming Continual Release Summing and Counting

For summing problem in the streaming continual release model, the binary tree mechanism was proposed in [19, 28]. It gets poly-logarithmic additive error and uses logarithmic space. Furthermore, it can handle negative numbers in the stream.

► **Theorem 10** ([19, 28]). *Let $\varepsilon \geq 0, \xi \in (0, 0.5)$, there is an ε -DP algorithm for summing in the streaming continual release model. With probability $1 - \xi$, the additive error of the output for every timestamp $t \in [T]$ is always at most $O\left(\frac{1}{\varepsilon} \log^{2.5}(T) \log\left(\frac{1}{\xi}\right)\right)$. The algorithm uses $O(\log(T))$ space.*

2.5 Probability Tools

► **Lemma 11** ([6]). *Let $\lambda \geq 4$ be an even integer. Let X be the sum of n λ -wise independent random variables which take values in $[0, 1]$. Let $\mu = E[X]$ and $A > 0$. Then we have*

$$\Pr[|X - \mu| > A] \leq 8 \cdot \left(\frac{\lambda\mu + \lambda^2}{A^2}\right)^{\lambda/2}$$

► **Lemma 12** (Median trick to boost success probability). *Suppose X is a random estimator of value v such that with probability at least $2/3$, X is an (α, γ) -approximation to v . Then, for $\xi \in (0, 0.5)$, if we draw $k = \lceil 50 \log(1/\xi) \rceil$ independent copies X_1, X_2, \dots, X_k of X , with probability at least $1 - \xi$, the median of X_1, X_2, \dots, X_k is an (α, γ) -approximation to v .*

Proof. We say X_i is good if X_i is an (α, γ) -approximation to v . If the median is not good, then $\sum_{i \in [k]} \mathbf{1}_{X_i \text{ is good}} < \frac{k}{2}$. By Chernoff bound, $\Pr[\text{median is not good}] \leq \Pr[\sum_{i \in [k]} \mathbf{1}_{X_i \text{ is good}} < \frac{k}{2}] \leq \Pr[E[\sum_{i \in [k]} \mathbf{1}_{X_i \text{ is good}}] - \frac{k}{2} > \frac{k}{6}] \leq e^{-k/48} \leq \xi$. \blacktriangleleft

3 Continual Released Summing with Better Additive Error

In this section, we show that if we allow a relative approximation and the input stream only contains non-negative numbers, then we can have a continual released summing algorithm with additive error better than Theorem 10.

■ **Algorithm 1** Grouping Stream of Counts.

Input: A stream of non-negative numbers c_1, c_2, \dots, c_T DP parameter $\varepsilon > 0$, approximation parameter $\eta \in (0, 0.5)$, and failure probability $\xi \in (0, 1)$.

Output: A stream of groups with grouped noisy counts $(\hat{c}_1, \hat{c}_2, \dots, \hat{c}_T)$

Let $\varepsilon_0 \leftarrow \varepsilon/2$.

Initialize a group index $i \leftarrow 1$, current group $G_1 \leftarrow \emptyset$, and threshold $\tau_1 \leftarrow \left(\frac{1}{\eta} + 1\right) \cdot \frac{7}{\varepsilon_0} \cdot \ln(3 \cdot T/\gamma) + \text{Lap}(2/\varepsilon_0)$. // G_i is used for analysis only.

for $t = 1$ **to** T **do**

$G_i \leftarrow G_i \cup \{t\}$.

Let $\nu_t \leftarrow \text{Lap}(4/\varepsilon_0)$.

if $\nu_t + \sum_{j \in G_i} c_j \geq \tau_i$ **then**

$\hat{c}_i \leftarrow \text{Lap}(1/\varepsilon_0) + \sum_{j \in G_i} c_j$.

$i \leftarrow i + 1$.

$\tau_i \leftarrow \left(\frac{1}{\eta} + 1\right) \cdot \frac{7}{\varepsilon_0} \cdot \ln(3 \cdot T/\gamma) + \text{Lap}(2/\varepsilon_0)$.

$G_i \leftarrow \emptyset$.

else

$\hat{c}_t \leftarrow 0$.

end

end

► **Lemma 13.** *The output stream $\hat{c}_1, \hat{c}_2, \dots, \hat{c}_T$ of Algorithm 1 is ε -DP.*

The proof idea is to iteratively apply sparse vector technique. See full version for the detailed proof.

► **Lemma 14.** *Let $\hat{c}_1, \hat{c}_2, \dots, \hat{c}_T$ be the output stream of Algorithm 1. Then with probability at least $1 - \xi$, $\forall l, r$ satisfying $1 \leq l \leq r \leq T$,*

$$(1 - \xi) \sum_{j=l}^r c_j - \left(\frac{1}{\eta} + 4\right) \cdot \frac{7}{\varepsilon_0} \cdot \ln(3 \cdot T/\xi) \leq \sum_{j=l}^r \hat{c}_j \leq (1 + \eta) \sum_{j=l}^r c_j + \left(\frac{1}{\eta} + 4\right) \cdot \frac{7}{\varepsilon_0} \cdot \ln(3 \cdot T/\xi).$$

See full version for the detailed proof.

► **Theorem 15** (Summing of a non-negative stream). *Let $\varepsilon \geq 0, \xi \in (0, 0.5)$, there is an ε -DP algorithm for summing in the streaming continual release model. If the input numbers are guaranteed to be non-negative, with probability at least $1 - \xi$, the output is always a $\left(1 + \eta, O\left(\frac{\log(T/\xi)}{\varepsilon\eta}\right)\right)$ -approximation to the summing problem at any timestamp $t \in [T]$. The algorithm uses space $O(1)$.*

Proof. According to Lemma 13, the output stream of Algorithm 1 is ε -DP, thus, we only need to solve non-private summing over $(\hat{c}_1, \hat{c}_2, \dots, \hat{c}_T)$. The approximation guarantee is given by Lemma 14. Note that we do not need to store G_i , we only need to maintain the sum of numbers in G_i at any timestamp. Thus, the total space needed is $O(1)$. ◀

4 Continual Released Number of Distinct Elements

In this section, we show how to use ε -DP streaming continual release summing to solve ε -DP streaming continual release number of distinct elements. In Section 4.1, we show how to estimate the number of distinct elements if the universe is small. In Section 4.2, we reduce the number of distinct elements of a large universe to the number of distinct elements of a small universe via subsampling. Due to the space limit, we put all missing details in the full version.

4.1 Number of Distinct Elements for Small Universe

■ **Algorithm 2** Number of Distinct Elements for Small Universe.

Input: A stream \mathcal{S} of elements $a_1, a_2, \dots, a_T \in \mathcal{U} \cup \{\perp\}$ with guarantee that $|\mathcal{U}| \leq m$.
Parameters : Relative approximation factor $\alpha \geq 1$ and additive approximation factor $\gamma \geq 0$ depending on the streaming continual release summing algorithm. //See Theorem 10.

Output: Estimation of the number of distinct elements at every timestamp t .

Initialize an empty stream \mathcal{C} .

Let $S \leftarrow \emptyset$.

for each a_t in the stream \mathcal{S} **do**

if $a_t \notin S$ and $a_t \neq \perp$ **then**

$S \leftarrow S \cup \{a_t\}$.

 Append 1 to the end of the stream \mathcal{C} .

end

else

 Append 0 to the end of the stream \mathcal{C} .

end

 Output an (α, γ) -approximation to the total counts of \mathcal{C} .

end

► **Lemma 16** (Approximation). *At the end of any time $t \in [T]$, the output of Algorithm 2 is an (α, γ) -approximation to the number of distinct elements.*

► **Lemma 17** (Privacy). *If the algorithm to continually release the approximate total counts of \mathcal{C} in Algorithm 2 is ε -DP, Algorithm 2 is 5ε -DP in the continual release model.*

► **Theorem 18** (Distinct elements for small universe). *Let $\varepsilon \geq 0, \xi \in (0, 0.5)$, suppose there is an ε -DP streaming continual release summing algorithm (for stream of non-negative numbers) which uses space J and with probability at least $1 - \xi$ always outputs an (α, γ) -approximation for every timestamp. There is a (5ε) -DP algorithm for the number of distinct elements of streams with universe size at most m in the streaming continual release model. With probability at least $1 - \xi$, the algorithm always outputs an (α, γ) -approximation for every timestamp $t \in [T]$. The algorithm uses $O(m + J)$ space.*

By combining the above theorem with Theorem 10, we obtain the following corollary.

► **Corollary 19** (Streaming continual release distinct elements for small universe). *There is an ε -DP algorithm for the number of distinct elements of streams with universe size at most m in the streaming continual release model. With probability at least $1 - \xi$, the additive error of the output is always at most $O\left(\frac{1}{\varepsilon} \log^{2.5}(T) \log\left(\frac{1}{\xi}\right)\right)$ for every timestamp $t \in [T]$. The algorithm uses $O(m + \log(T))$ space.*

By combining Theorem 18 with Theorem 15, we obtain the following corollary:

► **Corollary 20** (Streaming continual release distinct elements for small universe, better additive error). *There is an ε -DP algorithm for the number of distinct elements of streams with universe size at most m in the streaming continual release model. With probability at least $1 - \xi$, the additive error of the output is always an $\left(1 + \eta, O\left(\frac{\log(T/\xi)}{\varepsilon\eta}\right)\right)$ -approximation to the number of distinct elements for every timestamp $t \in [T]$. The algorithm uses $O(m)$ space.*

4.2 Number of Distinct Elements for General Universe

■ **Algorithm 3** Number of Distinct Elements via Subsampling.

Input: A stream \mathcal{S} of elements $a_1, a_2, \dots, a_T \in \mathcal{U} \cup \{\perp\}$, and a error parameter $\eta \in (0, 0.5)$.

Parameters : Relative approximation factor $\alpha \geq 1$ and additive approximation factor $\gamma \geq 0$ depending on the streaming continual release algorithm for number of distinct elements of streams with small universe of elements. //See Theorem 18.

Output: Estimation of the number of distinct elements $\|\mathcal{S}\|_0$.

$L \leftarrow \lceil \log \min(|\mathcal{U}|, T) \rceil, \lambda \leftarrow 2 \log(1000L), m \leftarrow 100L \cdot (16\alpha \max(\gamma/\eta, 32\alpha\lambda/\eta^2))^2$.

Let $h : \mathcal{U} \rightarrow [m]$ be a pairwise independent hash function.

//Here we treat $[m]$ as a universe of elements with size m instead of a set of integers.

Let $g : \mathcal{U} \rightarrow [L] \cup \{\perp\}$ be a λ -wise independent hash function and

$\forall a \in \mathcal{U}, i \in [L], \Pr[g(a) = i] = 2^{-i}, \Pr[g(a) = \perp] = 2^{-L}$.

Initialize empty streams $\mathcal{S}_1, \mathcal{S}_2, \dots, \mathcal{S}_L$.

for each a_t in the stream \mathcal{S} **do**

for $i \in [L]$ **do**

if $a_t \neq \perp$ and $g(a_t) = i$ **then**

 Append $h(a_t)$ to the end of the stream \mathcal{S}_i .

end

else

 Append \perp to the end of the stream \mathcal{S}_i .

end

end

$\forall i \in [L]$, compute \hat{s}_i which is an (α, γ) -approximation to $\|\mathcal{S}_i\|_0$.

 Find the largest $i \in [L]$ such that $\hat{s}_i \geq \max(\gamma/\eta, 32\alpha\lambda/\eta^2)$, and output $\hat{s}_i \cdot 2^i$.

 If such i does not exist, output 0.

end

► **Lemma 21** (Approximation). *Consider any timestamp $t \in [T]$. Let v be the output of Algorithm 3. With probability at least 0.9, v is a $((1 + O(\eta))\alpha, O(\alpha^2 \max(\gamma/\eta, \alpha \log(L)/\eta^2)))$ -approximation to $\|(a_1, a_2, \dots, a_t)\|_0$.*

► **Theorem 22** (Distinct elements). *Let $\varepsilon \geq 0, \xi, \xi' \in (0, 0.5), \eta \in (0, 0.5)$, suppose there is an ε -DP algorithm for the number of distinct elements of streams with element universe size at most $100 \log(\min(|\mathcal{U}|, T)) \cdot (16\alpha \max(\gamma/\eta, 32\alpha \cdot 2 \log(1000 \log(\min(|\mathcal{U}|, T)))/\eta^2))^2$ in the streaming continual release model which uses space J and with probability at least $1 - \xi$ always outputs an*

(α, γ) -approximation for every timestamp. There is an $(\varepsilon' = \lceil 50 \log(T/\xi') \rceil \varepsilon)$ -DP algorithm for the number of distinct elements of streams with element universe \mathcal{U} in the streaming continual release model. With probability at least $1 - \xi' - \log(\min(|\mathcal{U}|, T)) \cdot \lceil 50 \log(T/\xi') \rceil \cdot \xi$, the algorithm always outputs an $((1 + O(\eta))\alpha, O(\alpha^2 \max(\gamma/\eta, \alpha \log \log(\min(|\mathcal{U}|, T))/\eta^2)))$ -approximation for every timestamp $t \in [T]$. The algorithm uses $O(J \cdot \log(\min(|\mathcal{U}|, T)) \cdot \log(T/\xi'))$ space.

By plugging Corollary 19 into above theorem with $\xi = \frac{\xi'/2}{\log(\min(|\mathcal{U}|, T)) \cdot \lceil 50 \log(2T/\xi') \rceil}$, $\varepsilon = \frac{\varepsilon'}{\lceil 50 \log(2T/\xi') \rceil}$, $\alpha = 1$, $\gamma = O(\frac{1}{\varepsilon} \log^{2.5}(T) \log(1/\xi))$ and $J = O(\log(T) + 100 \log(\min(|\mathcal{U}|, T))) \cdot (16\alpha \max(\gamma/\eta, 32\alpha \cdot 2 \log(1000 \log(\min(|\mathcal{U}|, T)))/\eta^2))^2$, we get the following corollary.

► **Corollary 23** (Streaming continual release distinct elements). For $\eta \in (0, 0.5)$, there is an ε -DP algorithm for the number of distinct elements of streams with element universe \mathcal{U} in the streaming continual release model. With probability at least $1 - \xi$, the output is always a $(1 + \eta, O(\max(\frac{\log(T/\xi) \log^{2.5}(T) \log(1/\xi) \log \log(T/\xi)}{\eta \varepsilon}, \frac{\log \log T}{\eta^2})))$ -approximation for every timestamp $t \in [T]$. The algorithm uses $\text{poly}(\frac{\log(T/\xi)}{\eta \min(\varepsilon, 1)})$ space.

By plugging Corollary 20 into Theorem 22 with $\xi = \frac{\xi'/2}{\log(\min(|\mathcal{U}|, T)) \cdot \lceil 50 \log(2T/\xi') \rceil}$, $\varepsilon = \frac{\varepsilon'}{\lceil 50 \log(2T/\xi') \rceil}$, $\alpha = 1 + \eta$, $\gamma = O(\frac{\log(T/\xi)}{\varepsilon \eta})$ and

$$J = O(100 \log(\min(|\mathcal{U}|, T)) \cdot (16\alpha \max(\gamma/\eta, 32\alpha \cdot 2 \log(1000 \log(\min(|\mathcal{U}|, T)))/\eta^2))^2),$$

we get the following corollary.

► **Corollary 24** (Streaming continual release distinct elements, better dependence in $\log(T)$). For $\eta \in (0, 0.5)$, there is an ε -DP algorithm for the number of distinct elements of streams with element universe \mathcal{U} in the streaming continual release model. With probability at least $1 - \xi$, the output is always a $(1 + O(\eta), O(\frac{\log^2(T/\xi)}{\eta^2 \varepsilon}))$ -approximation for every timestamp $t \in [T]$. The algorithm uses $\text{poly}(\frac{\log(T/\xi)}{\eta \min(\varepsilon, 1)})$ space.

5 Continual Released ℓ_p Heavy Hitters and Frequency Moment Estimation

In this section, we present ε -DP streaming continual release algorithms for ℓ_p heavy hitters and frequency moment estimation. In Section 5.1, we present an algorithm for ε -DP CountSketch [13] in the streaming continual release model. The CountSketch is used for ℓ_2 heavy hitters and ℓ_2 moment estimation. In Section 5.2, we show how to use ℓ_2 heavy hitters to solve ℓ_p heavy hitters. In Section 5.3, we show how to estimate the number of elements which have low frequencies. In Section 5.4, we show how to use ℓ_p heavy hitters and the estimator of low frequency elements to estimate the ℓ_p frequency moment. Due to space limit, we put all missing proofs in the full version.

5.1 Continual Released CountSketch

► **Lemma 25** (DP guarantee). If the subroutine of continually releasing the approximate total counts of \mathcal{S}_i for every $i \in [k]$ in Algorithm 4 is ε -DP, Algorithm 4 is 2ε -DP.

► **Lemma 26** (Good approximation for frequent elements). Consider any $a \in \mathcal{U}$ and any timestamp $t \in [T]$. Let f_a be the frequency of a in a_1, a_2, \dots, a_t . Let (z_1, z_2, \dots, z_k) be the output of Algorithm 4 at timestamp t . Then $\forall \eta \in (0, 0.5)$, with probability at least $1 - 1/(k\eta^2)$, $|f_a - g(a) \cdot z_{h(a)}| \leq \eta \cdot \sqrt{\sum_{b \in \mathcal{U}} f_b^2} + \gamma$.

■ **Algorithm 4** Continual Released CountSketch.

Input: A stream \mathcal{S} of elements $a_1, a_2, \dots, a_T \in \mathcal{U} \cup \{\perp\}$, a parameter $k \in \mathbb{Z}_{\geq 1}$.
Parameters : Relative approximation factor $\alpha \geq 1$ and additive approximation factor $\gamma \geq 0$ depending on the streaming continual release summing algorithm.
//See Theorem 10.

Output: A tuple (z_1, z_2, \dots, z_k) at every timestamp t .
 Let $h: \mathcal{U} \rightarrow [k]$ be a 4-wise independent hash function, s.t., $\forall a \in \mathcal{U}, i \in [k], \Pr[h(a) = i] = \frac{1}{k}$.
 Let $g: \mathcal{U} \rightarrow \{-1, 1\}$ be a 4-wise independent hash function, s.t., $\forall a \in \mathcal{U}, \Pr[g(a) = 1] = \frac{1}{2}$.
 Initialize empty streams $\mathcal{S}_1, \mathcal{S}_2, \dots, \mathcal{S}_k$.

for each a_t in the stream \mathcal{S} **do**
 if $a_t = \perp$ **then**
 Append 0 to the end of every stream $\mathcal{S}_1, \mathcal{S}_2, \dots, \mathcal{S}_k$.
 end
 else
 Append $g(a_t)$ to the end of the stream $\mathcal{S}_{h(a_t)}$ and append 0 to the end of every stream \mathcal{S}_i for $i \neq h(a_t)$.
 end
 Output a tuple (z_1, z_2, \dots, z_k) where z_i is an estimation of the total counts of \mathcal{S}_i with additive error at most γ .
end

► **Lemma 27** (ℓ_2 Frequency moment estimation). Consider any timestamp $t \in [T]$. For $a \in \mathcal{U}$, let f_a be the frequency of a in a_1, a_2, \dots, a_t . Let (z_1, z_2, \dots, z_k) be the output of Algorithm 4 at timestamp t . Then $\forall \eta \in (0, 0.5)$, with probability at least $1 - 100/(k\eta^2)$, $|\sum_{i=1}^k z_i^2 - \sum_{a \in \mathcal{U}} f_a^2| \leq \eta \sum_{a \in \mathcal{U}} f_a^2 + 4k\gamma^2/\eta$

► **Theorem 28** (Streaming continual release ℓ_2 frequency estimators). Let $\varepsilon > 0, \eta \in (0, 0.5), \xi \in (0, 0.5)$. There is an ε -DP algorithm in the streaming continual release model such that with probability at least $1 - \xi$, it always outputs for every timestamp $t \in [T]$:

1. \hat{f}_a for every $a \in \mathcal{U}$ such that $|f_a - \hat{f}_a| \leq \eta \|\mathcal{S}\|_2 + O\left(\frac{\log(T/\xi) + \log(|\mathcal{U}|)}{\xi} \cdot \log^{2.5}(T) \cdot \log\left(\frac{\log(T/\xi) + \log(|\mathcal{U}|)}{\xi\eta}\right)\right)$, where \mathcal{S} denotes the stream (a_1, a_2, \dots, a_t) and f_a denotes the frequency of a in \mathcal{S} ,
 2. \hat{F}_2 such that $|\hat{F}_2 - \|\mathcal{S}\|_2^2| \leq \eta \|\mathcal{S}\|_2^2 + O\left(\frac{(\log(T/\xi) + \log(|\mathcal{U}|))^2}{\varepsilon^2 \eta^3} \cdot \log^5(T) \cdot \log^2\left(\frac{\log(T/\xi) + \log(|\mathcal{U}|)}{\xi\eta}\right)\right)$
- The algorithm uses $O\left(\frac{\log(T/\xi) + \log(|\mathcal{U}|)}{\eta^2} \cdot \log(T)\right)$ space.

5.2 Continual Released ℓ_p Heavy Hitters

By applying the CountSketch, we are able to develop ℓ_p heavy hitters.

► **Lemma 29** (DP guarantee). If $\forall i \in [m]$ the subroutine in Algorithm 5 of continually releasing $\hat{F}_{2,i}$ and \hat{f}_a for all a satisfying $h(a) = i$ is ε -DP, Algorithm 5 is 2ε -DP.

► **Lemma 30** (Approximation of frequencies). At any timestamp $t \in [T]$, $\forall a \in \mathcal{U}$, if $a \in \hat{H}$, $(1 - \eta)f_a^2 \leq \hat{f}_a^2 \leq (1 + \eta)f_a^2$ where f_a is the frequency of a in a_1, a_2, \dots, a_t .

► **Lemma 31** (Output size). At any timestamp t , the output H of Algorithm 5 has size at most $\left(\frac{1+\eta}{1-\eta}\right)^p \cdot k$.

► **Lemma 32** (Heavy hitters are in \hat{H}). At any timestamp t , consider any $a \in \mathcal{U}$. Let f_a be the frequency of a in a_1, a_2, \dots, a_t . If $f_a \geq 4\sqrt{\gamma_1/(\phi k)} + 512\gamma_2^2/\eta^2$ and $f_a^p \geq \|\mathcal{S}\|_p^p/k$, with probability at least 0.9, $a \in \hat{H}$.

► **Lemma 33**. At any timestamp t , if $f_a^p \geq \|\mathcal{S}\|_p^p/k$ and $a \in \hat{H}$, then $a \in H$.

■ **Algorithm 5** Continual Released ℓ_p Heavy Hitters ($p \in [0, \infty)$).

Input: A stream \mathcal{S} of elements $a_1, a_2, \dots, a_T \in \mathcal{U} \cup \{\perp\}$, a parameter $k \in \mathbb{Z}_{\geq 1}$, an error parameter $\eta \in (0, 0.5)$.

Parameters : Additive error parameters $\gamma_1, \gamma_2 \geq 0$ depending on the streaming continual release CountSketch algorithm. //See Theorem 28.

Output: A set $H \subseteq \mathcal{U}$ of elements and their estimated frequencies $\hat{f} : H \rightarrow \mathbb{R}_{\geq 0}$ at every timestamp t .

Let $\phi \geq \max(|\mathcal{U}|^{1-2/p}, 1)$. Let $m = 10k^2$.

Let $h : \mathcal{U} \rightarrow [m]$ be a pairwise independent hash function where $\forall a \in \mathcal{U}, i \in [m], \Pr[h(a) = i] = 1/m$.

Initialize empty streams $\mathcal{S}_1, \mathcal{S}_2, \dots, \mathcal{S}_m$. **for** each a_t in the stream \mathcal{S} **do**

if $a_t = \perp$ **then**

Append \perp to the end of every stream $\mathcal{S}_1, \mathcal{S}_2, \dots, \mathcal{S}_m$.

end

else

Append a_t to the end of the stream $\mathcal{S}_{h(a_t)}$ and append \perp to the end of every stream \mathcal{S}_i for $i \neq h(a_t)$.

end

For $i \in [m]$, compute $\hat{F}_{2,i}$ which is a $(1.1, \gamma_1)$ -approximation to $\|\mathcal{S}_i\|_2^2$.

For $a \in \mathcal{U}$, compute \hat{f}_a which is a $(1, (\eta/16)/(10\sqrt{\phi k}) \cdot \|\mathcal{S}_{h(a)}\|_2 + \gamma_2)$ -approximation to f_a , the frequency of a in \mathcal{S} (or equivalently in $\mathcal{S}_{h(a)}$).

For $a \in \mathcal{U}$, if $\hat{f}_a^2 \geq \frac{\hat{F}_{2,h(a)} + \gamma_1}{25\phi k} + \frac{512\gamma_2^2}{\eta^2}$, add a into \hat{H} .

Let $H \subseteq \hat{H}$ only keep the elements a such that \hat{f}_a is one the top- $(\frac{1+\eta}{1-\eta})^p \cdot k$ values among $\{\hat{f}_b \mid b \in \hat{H}\}$. For each $a \in H$, report $\hat{f}(a) \leftarrow \hat{f}_a$.

end

► **Theorem 34** (ℓ_p Heavy hitters for all $p \in [0, \infty)$). Let $\varepsilon > 0, \eta \in (0, 0.5), k \geq 1, \xi \in (0, 0.5)$. Let $\phi = \max(1, |\mathcal{U}|^{1-2/p})$. There is an ε -DP algorithm in the streaming continual release model such that with probability at least $1 - \xi$, it always outputs a set $H \subseteq \mathcal{U}$ and a function $\hat{f} : H \rightarrow \mathbb{R}$ for every timestamp $t \in [T]$: such that

1. $\forall a \in H, \hat{f}(a) \in (1 \pm \eta) \cdot f_a$ where f_a is the frequency of a in the stream $\mathcal{S} = (a_1, a_2, \dots, a_t)$,
2. $\forall a \in \mathcal{U}$, if $f_a \geq \frac{1}{\varepsilon \eta} \cdot \log^C \left(\frac{T \cdot k \cdot |\mathcal{U}|}{\xi \eta} \right)$ for some sufficiently large constant $C > 0$ and $f_a^p \geq \|\mathcal{S}\|_p^p / k$ then $a \in H$,
3. The size of H is at most $O \left((\log(T/\xi) + \log(|\mathcal{U}|)) \cdot \left(\frac{1+\eta}{1-\eta} \right)^p \cdot k \right)$.

The algorithm uses $\frac{\phi k^3}{\eta^2} \cdot \text{poly} \left(\log \left(\frac{T \cdot k \cdot |\mathcal{U}|}{\xi} \right) \right)$ space.

5.3 Differentially Private Continual Released Counting of Low Frequency Elements

In this section, we show a differentially private continual released algorithm for counting the number of elements that have a certain (low) frequency. Similar to our counting distinct elements algorithm, we first consider the case where the universe of the elements is small.

5.3.1 Number of Low Frequency Elements for Small Universe

► **Lemma 35** (Approximation). At the end of any time $t \in [T], \forall i \in [k]$, the output \hat{s}_i of Algorithm 6 is an (α, γ) -approximation to $|\{a \in \mathcal{U} \mid f_a = i\}|$, the size of the set of elements of which the frequency is exact i .

■ **Algorithm 6** Number of Low Frequency Elements for Small Universe.

Input: A stream \mathcal{S} of elements $a_1, a_2, \dots, a_T \in \mathcal{U} \cup \{\perp\}$ with guarantee that $|\mathcal{U}| \leq m$, and a target frequency k .

Parameters : Relative approximation factor $\alpha \geq 1$ and additive approximation factor $\gamma \geq 0$ depending on the streaming continual release summing algorithm.

//See Theorem 10.

Output: Estimation of the number of elements with frequency exactly i for each $i \in [k]$ at every timestamp t .

Initialize empty streams $\mathcal{C}_1, \mathcal{C}_2, \dots, \mathcal{C}_k$.

For each $a \in \mathcal{U}$, initialize frequency $f(a) \leftarrow 0$

for each a_t in the stream \mathcal{S} **do**

 If $a_t \neq \perp$, $f(a_t) \leftarrow f(a_t) + 1$.

for each $i \in [k]$ **do**

if $a_t \neq \perp$ and $f(a_t) = i + 1$ **then**
 | Append -1 at the end of \mathcal{C}_i .

end

else if $a_t \neq \perp$ and $f(a_t) = i$ **then**
 | Append 1 at the end of \mathcal{C}_i .

end

else

 | Append 0 at the end of \mathcal{C}_i .

end

end

 For each $i \in [k]$, output \hat{s}_i which is an (α, γ) -approximation to the total counts of \mathcal{C}_i .

end

► **Lemma 36** (Privacy). *If the algorithm that continually release the approximate total counts of \mathcal{C}_i for every $i \in [k]$ is ε -DP, Algorithm 6 is $(8k\varepsilon)$ -DP in the continual release model.*

► **Theorem 37** (Streaming continual release count of low frequency elements for small universe). *Let $k \geq 1, \varepsilon \geq 0, \xi \in (0, 0.5)$. Suppose the universe \mathcal{U} has size at most m . There is an ε -DP algorithm in the streaming continual release model such that with probability at least $1 - \xi$, it always outputs k numbers $\hat{s}_1, \hat{s}_2, \dots, \hat{s}_k$ for every timestamp t , such that $\forall i \in [k], \hat{s}_i$ is an approximation to $|\{a \in \mathcal{U} \mid f_a = i\}|$ with additive error $O\left(\frac{k}{\varepsilon} \cdot \log^{2.5}(T) \log\left(\frac{k}{\xi}\right)\right)$. The algorithm uses $O(m + k \log(T))$ space.*

5.3.2 Number of Low Frequency Elements for General Universe

► **Lemma 38** (Privacy). *If the subroutine of continually releasing \hat{d} in Algorithm 7 is ε -DP and the subroutine of continually releasing $\{\hat{s}_{i,1}, \hat{s}_{i,2}, \dots, \hat{s}_{i,k}\}$ for each $i \in [k]$ is also ε -DP, Algorithm 7 is 3ε -DP.*

► **Lemma 39** (Approximation). *Consider an arbitrary timestamp $t \in [T]$. $\forall a \in \mathcal{U}$, let f_a denote the frequency of a in a_1, a_2, \dots, a_t . With probability at least 0.9 , $\forall j \in [k]$, the output \hat{s}_j of Algorithm 7 is an $\left(\alpha_2, \left(\alpha_2 \eta + \frac{\eta^2 \gamma_2}{32\lambda}\right) \cdot \|\mathcal{S}\|_0 + 6\gamma_1 + 128\lambda/\eta^2\right)$ -approximation to $|\{a \in \mathcal{U} \mid f_a = j\}|$, where $\|\mathcal{S}\|_0$ is the number of distinct elements in a_1, a_2, \dots, a_t .*

► **Theorem 40** (Streaming continual release of count of low frequency elements). *Let $k \geq 1, \varepsilon \geq 0, \xi \in (0, 0.5), \eta \in (0, 0.5)$. There is an ε -DP algorithm in the streaming continual release model such that with probability at least $1 - \xi$, it always outputs k numbers $\hat{s}_1, \hat{s}_2, \dots, \hat{s}_k$*

Algorithm 7 Number of Low Frequency Elements via Subsampling.

Input: A stream \mathcal{S} of elements $a_1, a_2, \dots, a_T \in \mathcal{U} \cup \{\perp\}$, an error parameter $\eta \in (0, 0.5)$, and a parameter $k \geq 1$.

Parameters : Additive approximation factor γ_1 depending on the streaming continual release number of distinct elements, relative approximation factor $\alpha_2 \geq 1$ and additive approximation factor $\gamma_2 \geq 0$ depending on the streaming continual release count of low frequency elements for small universe.
//See Corollary 23 and Theorem 37.

Output: Estimation of the number of elements with frequency exactly i for each $i \in [k]$ at every timestamp t .

$L \leftarrow \lceil \log \min(|\mathcal{U}|, T) \rceil, \lambda \leftarrow 2 \log(1000k), m \leftarrow 100 \cdot (25600\lambda/\eta^2)^2$.

Let $h : \mathcal{U} \rightarrow [m]$ be a pairwise independent hash function.

Let $g : \mathcal{U} \rightarrow [L] \cup \{\perp\}$ be a λ -wise independent hash function and $\forall a \in \mathcal{U}, i \in [L], \Pr[g(a) = i] = 2^{-i}, \Pr[g(a) = \perp] = 2^{-L}$.

Initialize empty streams $\mathcal{S}_1, \mathcal{S}_2, \dots, \mathcal{S}_L$.

for each a_t in the stream \mathcal{S} **do**

for $i \in L$ **do**

if $a_t \neq \perp$ and $g(a_t) = i$ **then**

| Append $h(a_t)$ to the end of the stream \mathcal{S}_i .

end

else

| Append \perp to the end of the stream \mathcal{S}_i .

end

end

Compute \hat{d} which is a $(1.1, \gamma_1)$ -approximation to the number of distinct elements in \mathcal{S} .

For $i \in [L]$, compute $\hat{s}_{i,1}, \hat{s}_{i,2}, \dots, \hat{s}_{i,k}$ where $\hat{s}_{i,j}$ is an (α_2, γ_2) -approximation to the number of elements in \mathcal{S}_i where each element has frequency exact j .

If $\hat{d} \leq \max(3\gamma_1, 64\lambda/\eta^2)$, set $\hat{s}_1 = \hat{s}_2 = \dots = \hat{s}_k = 0$.

Otherwise, find the largest $i^* \in [L]$ such that $2^{i^*} \cdot (64\lambda/\eta^2) \leq \hat{d}$, and $\forall j \in [k]$, set $\hat{s}_j = \hat{s}_{i^*,j} \cdot 2^i$.

Output $\hat{s}_1, \hat{s}_2, \dots, \hat{s}_k$.

end

for every timestamp t such that $\forall i \in [k], \hat{s}_i$ is an approximation to $|\{a \in \mathcal{U} \mid f_a = i\}|$ with additive error:

$$\left(\eta + \eta^2 \cdot \frac{k}{\varepsilon} \cdot \text{poly} \left(\log \left(\frac{Tk}{\xi} \right) \right) \right) \cdot \|\mathcal{S}\|_0 + \frac{1}{\varepsilon} \cdot \text{poly} \left(\log \left(\frac{T}{\xi} \right) \right) + O \left(\frac{\log k}{\eta^2} \right)$$

The algorithm uses $\frac{1}{\eta^4} \cdot \text{poly} \left(\frac{\log(T \cdot k/\xi)}{\min(\varepsilon, 1)} \right)$ space.

5.4 ℓ_p Moment Estimation

In this section, we show how to use our ℓ_p heavy hitters and the estimator of number of low frequency elements to solve ℓ_p moment estimation problem.

► **Lemma 41** (Privacy). *Consider the subroutines in Algorithm 8. If the algorithm that continually release $\hat{s}_1, \hat{s}_2, \dots, \hat{s}_k$ is ε -DP and for every $i \in [L] \cup \{0\}$ the algorithm that continually release (H_i, \hat{f}_i) is ε -DP. Algorithm 8 is 4ε -DP in the continual release model.*

In the remaining of the section, let us analyze the approximation guarantee of Algorithm 8. Let us consider a timestamp $t \in [T]$ and $\forall a \in \mathcal{U}$, let f_a denote the frequency of a in a_1, a_2, \dots, a_t . Let $\mathcal{S}, \mathcal{S}_0, \mathcal{S}_1, \dots, \mathcal{S}_L$ denote the streams up to timestamp t . Let $\mathcal{I} = \{[1, 1], [2, 2], \dots, [k, k], I_{q_1^*}, I_{q_1^*+1}, \dots, I_{q_2^*}\}$. By our choice of k, q_1^*, q_2^* , we have the following observation:

■ **Algorithm 8** ℓ_p Frequency Moment Estimation.

Input: A stream \mathcal{S} of elements $a_1, a_2, \dots, a_T \in \mathcal{U} \cup \{\perp\}$, an error parameter $\eta \in (0, 0.5)$.
Parameters : A threshold parameter τ depending on the heavy hitter algorithm, a relative approximation factor α and an additive approximation factor γ depending on the algorithm of estimating count of low frequency elements. //See Theorem 34 and Theorem 40.

Output: Estimation of the ℓ_p frequency moment at every timestamp t .

Let β' be drawn uniformly at random from $[1/2, 1]$ and let $\beta \in \beta' \pm (\eta/T)^C$ for some sufficiently large constant $C > 0$. //Thus β can be represented by $\Theta(\log(T/\eta))$ bits.
Let q_1^* be the smallest integer that $\beta(1 + \eta)^{q_1^*} > \tau$ and let q_2^* be the smallest integer that $\beta(1 + \eta)^{q_2^*+1} \geq T$, and for any $q \in [q_1^*, q_2^*]$, define the interval $I_q = (\beta(1 + \eta)^q, \beta(1 + \eta)^{q+1}]$.
 $[L \leftarrow \log(|\mathcal{U}|)], \lambda \leftarrow 2 \cdot \log(1000(L + 1) \log(4T)/\eta)$.
Let k be the largest integer such that $k \leq \beta(1 + \eta)^{q_1^*}$. Let
 $B \leftarrow \left(\frac{\log(4T)}{\eta}\right) \cdot 100(L + 1) \cdot \frac{32\lambda}{\eta^3} \cdot (1 + \eta)^p$.
Let $g : \mathcal{U} \rightarrow [L] \cup \{\perp\}$ be a λ -wise independent hash function and
 $\forall a \in \mathcal{U}, i \in [L], \Pr[g(a) = i] = 2^{-i}, \Pr[g(a) = \perp] = 2^{-L}$.
Initialize empty streams $\mathcal{S}_0, \mathcal{S}_1, \mathcal{S}_2, \dots, \mathcal{S}_L$.

for each a_t in the stream \mathcal{S} **do**
 if $a_t \neq \perp$ **then**
 Append a_t at the end of \mathcal{S}_0 .
 For $i \in [L]$, if $g(a_t) = i$, append a_t to \mathcal{S}_i , otherwise append \perp to \mathcal{S}_i .
 end
 else
 For $i \in [L] \cup \{0\}$, append \perp at the end of \mathcal{S}_i .
 end

For each $i \in [L] \cup \{0\}$, compute a set $H_i \subseteq \mathcal{U}$ together with a function $\hat{f}_i : H_i \rightarrow \mathbb{R}_{\geq 0}$ satisfying:

1. $\forall a \in H_i, (1 - \eta') \cdot f_a \leq \hat{f}_i(a) \leq (1 + \eta') \cdot f_a$, where f_a is the frequency of a in a_1, a_2, \dots, a_t , and η' satisfies $\eta' \leq \frac{\eta}{10000(L+1)|H_i|}$.
2. $\forall a \in \mathcal{U}$ that appears in \mathcal{S}_i , if $f_a \geq \tau$ and $f_a^p \geq \|\mathcal{S}_i\|_p^p / B$, $a \in H_i$.

Compute $\hat{s}_1, \hat{s}_2, \dots, \hat{s}_k$ where $\forall l \in [k], \hat{s}_l$ is an (α, γ) -approximation to $|\{a \in \mathcal{U} \mid f_a = l\}|$.

for $q \in [q_1^*, q_2^*]$ **do**
 Initialize $\hat{z}_q = 0$.
 for $i \in [L] \cup \{0\}$ **do**
 if $|\{a \in H_i \mid \hat{f}_i(a) \in I_q\}| \geq 8\lambda/\eta^2$ or $i = 0$ **then**
 $\hat{z}_q \leftarrow \max(\hat{z}_q, |\{a \in H_i \mid \hat{f}_i(a) \in I_q\}| \cdot 2^i)$.
 end
 end

end
Output $\hat{F}_p = \sum_{l \in [k]} \hat{s}_l \cdot l^p + \sum_{q \in [q_1^*, q_2^*]} \hat{z}_q \cdot (\beta(1 + \eta)^q)^p$

end

► **Observation 42.** $\sum_{I \in \mathcal{I}} \sum_{a \in \mathcal{U}: f_a \in I} f_a^p = \|\mathcal{S}\|_p^p$.

► **Definition 43.** For any $I \in \mathcal{I}$, if $\sum_{a \in \mathcal{U}: f_a \in I} f_a^p \geq \eta \|\mathcal{S}\|_p^p / (q_2^* - q_1^* + 1)$ or I is $\{i\}$ for some $i \in [k]$, then interval I is contributing.

5.4.1 Analysis of High Frequency Elements

We show that $\sum_{q \in [q_1^*, q_2^*]} \hat{z}_q \cdot (\beta(1 + \eta)^q)^p$ is a good approximation to $\sum_{q \in [q_1^*, q_2^*]} \sum_{a \in \mathcal{U}: f_a \in I_q} f_a^p$ and $\sum_{q \in [q_1^*, q_2^*]: I_q \text{ is contributing}} \sum_{a \in \mathcal{U}: f_a \in I_q} f_a^p$. See full version for more details.

► **Lemma 44** (Upper bound of estimation of high frequency moment). *With probability at least 0.98, $\sum_{q \in [q_1^*, q_2^*]} \hat{z}_q \cdot (\beta(1 + \eta)^q)^p \leq (1 + \eta) \cdot \sum_{q \in [q_1^*, q_2^*]} \sum_{a \in \mathcal{U}, f_a \in I_q} f_a^p$*

► **Lemma 45** (Lower bound of estimation of contributing high frequency moment). *With probability at least 0.97, $\sum_{q \in [q_1^*, q_2^*]} \hat{z}_q \cdot (\beta(1 + \eta)^q)^p \geq (1 - \eta)^{p+1} \cdot \sum_{q \in [q_1^*, q_2^*]: I_q \text{ is contributing}} \sum_{a \in \mathcal{U}, f_a \in I_q} f_a^p$*

5.4.2 Analysis of Low Frequency Elements

In this section, we show that $\sum_{l \in [k]} \hat{s}_l \cdot l^p$ is a good approximation to $\sum_{l \in [k]} \sum_{a \in \mathcal{U}: f_a = l} f_a^p$.

► **Lemma 46** (Approximation of low frequency moments). *$\sum_{l \in [k]} \hat{s}_l \cdot l^p$ is a $(\alpha, \gamma \cdot (2\tau)^{p+1})$ -approximation to $\sum_{l \in [k]} \sum_{a \in \mathcal{U}: f_a = l} f_a^p$.*

5.4.3 Putting All Together

► **Lemma 47.** $\sum_{\text{contributing } I \in \mathcal{I}} \sum_{a \in \mathcal{U}: f_a \in I} f_a^p \geq (1 - \eta) \cdot \|\mathcal{S}\|_p^p$.

► **Lemma 48.** *Consider any timestamp $t \in [T]$. $\forall a \in \mathcal{U}$, let f_a denote the frequency of a in a_1, a_2, \dots, a_t . With probability at least 0.9, the output \hat{F}_p of Algorithm 8 is a $\left(\max\left(\frac{\alpha}{1-\eta}, (1+2\eta)^{p+2}\right), \gamma \cdot (2\tau)^{p+1}\right)$ -approximation to $\|\mathcal{S}\|_p^p$.*

► **Theorem 49** (Streaming continual release ℓ_p frequency moment estimation). *Let $p > 0, \varepsilon \geq 0, \xi \in (0, 0.5), \eta \in (0, 0.5)$. There is an ε -DP algorithm in the streaming continual release model such that with probability at least $1 - \xi$, it always outputs an $\left(1 + \eta, \left(\frac{\log(T|\mathcal{U}|/\xi)}{\eta^\varepsilon}\right)^{O(\max(1,p))}\right)$ -approximation to $\|\mathcal{S}\|_p^p$. The algorithm uses space at most $\phi \cdot \left(\frac{\log(T|\mathcal{U}|/\xi)}{\eta^\varepsilon}\right)^{O(\max(1,p))}$, where $\phi = \max(1, |\mathcal{U}|^{1-2/p})$.*

6 Extension to Sliding Window Continual Release Algorithms

In this section, we show how to extend our results to the sliding window setting.

► **Corollary 50** (Sliding window summing of a non-negative numbers). *Let $\eta \in (0, 0.5), \varepsilon \geq 0, \xi \in (0, 0.5)$, there is an ε -DP algorithm for summing in the sliding window continual release model. If the input numbers are guaranteed to be non-negative, with probability at least $1 - \xi$, the output is always a $\left(1 + \eta, O\left(\frac{\log(T/(\eta\xi)) \log(T)}{\varepsilon \eta^3}\right)\right)$ -approximation to the summing problem at any timestamp $t \in [T]$. The algorithm uses space $O(\log(T)/\eta)$.*

► **Corollary 51** (Sliding window continual release distinct elements). *For $\eta \in (0, 0.5), \varepsilon \geq 0, \xi \in (0, 0.5)$ there is an ε -DP algorithm for the number of distinct elements of streams with element universe \mathcal{U} in the sliding window continual release model. With probability at least $1 - \xi$, the output is always a $\left(1 + \eta, O\left(\frac{\log^2(T/(\eta\xi)) \log(T)}{\eta^4 \varepsilon}\right)\right)$ -approximation for every timestamp $t \in [T]$. The algorithm uses $\text{poly}\left(\frac{\log(T/\xi)}{\eta \min(\varepsilon, 1)}\right)$ space.*

► **Corollary 52** (Sliding window continual release ℓ_2 frequency moments). *Let $\varepsilon > 0, \eta \in (0, 0.5), \xi \in (0, 0.5)$. There is an ε -DP algorithm in the sliding window continual release model such that with probability at least $1 - \xi$, it always outputs \hat{F}_2 for every timestamp $t \in [T]$ such that $|\hat{F}_2 - \|\mathcal{S}\|_2^2| \leq \eta \|\mathcal{S}\|_2^2 + O\left(\frac{(\log(T/(\xi\eta)) + \log(|\mathcal{U}|))^2 \log^2(T)}{\varepsilon^2 \eta^3} \cdot \log^5(T) \cdot \log^2\left(\frac{\log(T/\xi) + \log(|\mathcal{U}|)}{\xi \eta}\right)\right)$, where \mathcal{S} denotes the sub-stream corresponding to the latest W elements at timestamp t . The algorithm uses $O\left(\frac{\log(T/(\xi\eta)) + \log(|\mathcal{U}|)}{\eta^4} \cdot \log^2(T)\right)$ space.*

► **Corollary 53** (Sliding window continual release ℓ_p frequency moments). *Let $p > 0, \varepsilon \geq 0, \xi \in (0, 0.5), \eta \in (0, 0.5)$. There is an ε -DP algorithm in the sliding window continual release model such that with probability at least $1 - \xi$, it always outputs an $\left(1 + \eta, \left(\frac{\log(T|\mathcal{U}|/\xi)}{\eta\varepsilon}\right)^{O(\max(1,p))}\right)$ -approximation to $\|\mathcal{S}\|_p^p$, where \mathcal{S} denotes the sub-stream corresponding to the latest W elements at timestamp t . The algorithm uses space at most $\phi \cdot \left(\frac{\log(T|\mathcal{U}|/\xi)}{\eta\varepsilon}\right)^{O(\max(1,p))}$, where $\phi = \max(1, |\mathcal{U}|^{1-2/p})$.*

References

- 1 Naman Agarwal and Karan Singh. The price of differential privacy for online learning. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 32–40. PMLR, 06–11 August 2017. URL: <https://proceedings.mlr.press/v70/agarwal17a.html>.
- 2 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.
- 3 Alexandr Andoni. High frequency moments via max-stability. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6364–6368. IEEE, 2017.
- 4 Alexandr Andoni, Robert Krauthgamer, and Krzysztof Onak. Streaming algorithms via precision sampling. In *2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*, pages 363–372. IEEE, 2011.
- 5 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.
- 6 Mihir Bellare and John Rompel. Randomness-efficient oblivious sampling. In *Proceedings 35th Annual Symposium on Foundations of Computer Science*, pages 276–287. IEEE, 1994.
- 7 Jeremiah Blocki, Avrim Blum, Anupam Datta, and Or Sheffet. The johnson-lindenstrauss transform itself preserves differential privacy. In *53rd Annual IEEE Symposium on Foundations of Computer Science, FOCS 2012, New Brunswick, NJ, USA, October 20-23, 2012*, pages 410–419. IEEE Computer Society, 2012. doi:10.1109/FOCS.2012.67.
- 8 Jeremiah Blocki, Elena Grigorescu, Tamalika Mukherjee, and Samson Zhou. How to make your approximation algorithm private: A black-box differentially-private transformation for tunable approximation algorithms of functions with low sensitivity. *arXiv preprint*, 2022. arXiv:2210.03831.
- 9 Jean Bolot, Nadia Fawaz, S. Muthukrishnan, Aleksandar Nikolov, and Nina Taft. Private decayed predicate sums on streams. In *Proceedings of the 16th International Conference on Database Theory, ICDT '13*, pages 284–295, New York, NY, USA, 2013. Association for Computing Machinery. doi:10.1145/2448496.2448530.
- 10 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. doi:10.1109/FOCS.2007.55.
- 11 Zhiqi Bu, Sivakanth Gopi, Janardhan Kulkarni, Yin Tat Lee, Judy Hanwen Shen, and Uthai-pon Tantipongpipat. Fast and memory efficient differentially private-sgd via JL projections. In Marc’Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 19680–19691, 2021. URL: <https://proceedings.neurips.cc/paper/2021/hash/a3842ed7b3d0fe3ac263bcabd2999790-Abstract.html>.

- 12 T-H Hubert Chan, Mingfei Li, Elaine Shi, and Wenchang Xu. Differentially private continual monitoring of heavy hitters from distributed streams. In *International Symposium on Privacy Enhancing Technologies Symposium*, pages 140–159. Springer, 2012.
- 13 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.
- 14 Seung Geol Choi, Dana Dachman-Soled, Mukul Kulkarni, and Arkady Yerukhimovich. Differentially-private multi-party sketching for large-scale statistics. Cryptology ePrint Archive, Paper 2020/029, 2020. URL: <https://eprint.iacr.org/2020/029>.
- 15 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.
- 16 Mayur Datar, Aristides Gionis, Piotr Indyk, and Rajeev Motwani. Maintaining stream statistics over sliding windows. *SIAM journal on computing*, 31(6):1794–1813, 2002.
- 17 Marianne Durand and Philippe Flajolet. Loglog counting of large cardinalities. In *European Symposium on Algorithms*, pages 605–617. Springer, 2003.
- 18 Cynthia Dwork. Differential privacy: A survey of results. In *International conference on theory and applications of models of computation*, pages 1–19. Springer, 2008.
- 19 Cynthia Dwork, Moni Naor, Toniann Pitassi, and Guy N. Rothblum. Differential privacy under continual observation. In Leonard J. Schulman, editor, *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC*, pages 715–724. ACM, 2010.
- 20 Cynthia Dwork, Moni Naor, Toniann Pitassi, Guy N Rothblum, and Sergey Yekhanin. Pan-private streaming algorithms. In *ics*, pages 66–80, 2010.
- 21 Cynthia Dwork, Moni Naor, Omer Reingold, and Guy N. Rothblum. Pure differential privacy for rectangle queries via private partitions. In *Proceedings, Part II, of the 21st International Conference on Advances in Cryptology – ASIACRYPT 2015 - Volume 9453*, pages 735–751, Berlin, Heidelberg, 2015. Springer-Verlag. doi:10.1007/978-3-662-48800-3_30.
- 22 Cynthia Dwork, Aaron Roth, et al. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3–4):211–407, 2014.
- 23 Hendrik Fichtenberger, Monika Henzinger, and Wolfgang Ost. Differentially private algorithms for graphs under continual observation. In Petra Mutzel, Rasmus Pagh, and Grzegorz Herman, editors, *29th Annual European Symposium on Algorithms, ESA 2021, September 6–8, 2021, Lisbon, Portugal (Virtual Conference)*, volume 204 of *LIPICs*, pages 42:1–42:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPICs.ESA.2021.42.
- 24 Philippe Flajolet, Éric Fusy, Olivier Gandouet, and Frédéric Meunier. Hyperloglog: the analysis of a near-optimal cardinality estimation algorithm. In *Discrete Mathematics and Theoretical Computer Science*, pages 137–156. Discrete Mathematics and Theoretical Computer Science, 2007.
- 25 Philippe Flajolet and G Nigel Martin. Probabilistic counting algorithms for data base applications. *Journal of computer and system sciences*, 31(2):182–209, 1985.
- 26 Sumit Ganguly. Polynomial estimators for high frequency moments. *arXiv preprint*, 2011. arXiv:1104.4552.
- 27 Hsiang Hsu, Natalia Martinez, Martin Bertran, Guillermo Sapiro, and Flavio P. Calmon. A survey on statistical, information, and estimation—theoretic views on privacy. *IEEE BITS the Information Theory Magazine*, 1(1):45–56, 2021.
- 28 T-H. Hubert Chan, Elaine Shi, and Dawn Song. Private and continual release of statistics. In Samson Abramsky, Cyril Gavoille, Claude Kirchner, Friedhelm Meyer auf der Heide, and Paul G. Spirakis, editors, *Automata, Languages and Programming*, pages 405–417, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.
- 29 Piotr Indyk. Stable distributions, pseudorandom generators, embeddings, and data stream computation. *Journal of the ACM (JACM)*, 53(3):307–323, 2006.

- 30 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.
- 31 Palak Jain, Sofya Raskhodnikova, Satchit Sivakumar, and Adam D. Smith. The price of differential privacy under continual observation. *CoRR*, abs/2112.00828, 2021. [arXiv: 2112.00828](https://arxiv.org/abs/2112.00828).
- 32 Prateek Jain, Pravesh Kothari, and Abhradeep Thakurta. Differentially private online learning. In Shie Mannor, Nathan Srebro, and Robert C. Williamson, editors, *Proceedings of the 25th Annual Conference on Learning Theory*, volume 23 of *Proceedings of Machine Learning Research*, pages 24.1–24.34, Edinburgh, Scotland, 25–27 June 2012. PMLR. URL: <https://proceedings.mlr.press/v23/jain12.html>.
- 33 Hossein Jowhari, Mert Sağlam, and Gábor Tardos. Tight bounds for lp samplers, finding duplicates in streams, and related problems. In *Proceedings of the thirtieth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 49–58, 2011.
- 34 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.
- 35 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.
- 36 Ping Li. Estimators and tail bounds for dimension reduction in l_α ($0 < \alpha \leq 2$) using stable random projections. In *Proceedings of the nineteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 10–19, 2008.
- 37 Yi Li and David P Woodruff. A tight lower bound for high frequency moment estimation with small error. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 623–638. Springer, 2013.
- 38 Darakhshan Mir, Shan Muthukrishnan, Aleksandar Nikolov, and Rebecca N Wright. Pan-private algorithms via statistics on sketches. In *Proceedings of the thirtieth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 37–48, 2011.
- 39 Jayadev Misra and David Gries. Finding repeated elements. *Science of computer programming*, 2(2):143–152, 1982.
- 40 Robert Morris. Counting large numbers of events in small registers. *Communications of the ACM*, 21(10):840–842, 1978.
- 41 Victor Perrier, Hassan Jameel Asghar, and Dali Kaafar. Private continual release of real-valued data streams. In *26th Annual Network and Distributed System Security Symposium, NDSS 2019, San Diego, California, USA, February 24-27, 2019*. The Internet Society, 2019. URL: <https://www.ndss-symposium.org/ndss-paper/private-continual-release-of-real-valued-data-streams/>.
- 42 Michael Saks and Xiaodong Sun. Space lower bounds for distance approximation in the data stream model. In *Proceedings of the thirty-fourth annual ACM symposium on Theory of computing*, pages 360–369, 2002.
- 43 Or Sheffet. Differentially private ordinary least squares. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 3105–3114. PMLR, 2017. URL: <http://proceedings.mlr.press/v70/sheffet17a.html>.
- 44 Adam Smith and Abhradeep Thakurta. (nearly) optimal algorithms for private online learning in full-information and bandit settings. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2, NIPS'13*, pages 2733–2741, Red Hook, NY, USA, 2013. Curran Associates Inc.

- 45 Adam D. Smith, Shuang Song, and Abhradeep Thakurta. The flajolet-martin sketch itself preserves differential privacy: Private counting with minimal space. In Hugo Larochelle, Marc’Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL: <https://proceedings.neurips.cc/paper/2020/hash/e3019767b1b23f82883c9850356b71d6-Abstract.html>.
- 46 Shuang Song, Susan Little, Sanjay Mehta, Staal Vinterbo, and Kamalika Chaudhuri. Differentially private continual release of graph statistics, 2018. doi:10.48550/ARXIV.1809.02575.
- 47 Robert H. Morris Sr. Counting large numbers of events in small registers. *Commun. ACM*, 21(10):840–842, 1978.
- 48 Mikkel Thorup and Yin Zhang. Tabulation based 4-universal hashing with applications to second moment estimation. In *SODA*, volume 4, pages 615–624, 2004.
- 49 Jalaj Upadhyay. Sublinear space private algorithms under the sliding window model. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 6363–6372. PMLR, 09–15 June 2019. URL: <https://proceedings.mlr.press/v97/upadhyay19a.html>.
- 50 Lun Wang, Iosif Pinelis, and Dawn Song. Differentially private fractional frequency moments estimation with polylogarithmic space. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL: <https://openreview.net/forum?id=7I8LPkcx8V>.