Differential Privacy on Finite Computers

Victor Balcer* Salil Vadhan†
Center for Research on Computation & Society
School of Engineering & Applied Sciences
Harvard University
vbalcer@g.harvard.edu, salil_vadhan@harvard.edu

September 19, 2017

Abstract

We consider the problem of designing and analyzing differentially private algorithms that can be implemented on *discrete* models of computation in *strict* polynomial time, motivated by known attacks on floating point implementations of real-arithmetic differentially private algorithms (Mironov, CCS 2012) and the potential for timing attacks on expected polynomial-time algorithms. We use a case study the basic problem of approximating the histogram of a categorical dataset over a possibly large data universe $\mathcal{X}$. The classic Laplace Mechanism (Dwork, McSherry, Nissim, Smith, TCC 2006 and J. Privacy & Confidentiality 2017) does not satisfy our requirements, as it is based on real arithmetic, and natural discrete analogues, such as the Geometric Mechanism (Ghosh, Roughgarden, Sundarajan, STOC 2009 and SICOMP 2012), take time at least linear in $|\mathcal{X}|$, which can be exponential in the bit length of the input.

In this paper, we provide strict polynomial-time discrete algorithms for approximate histograms whose simultaneous accuracy (the maximum error over all bins) matches that of the Laplace Mechanism up to constant factors, while retaining the same (pure) differential privacy guarantee. One of our algorithms produces a sparse histogram as output. Its "per-bin accuracy" (the error on individual bins) is worse than that of the Laplace Mechanism by a factor of $\log |\mathcal{X}|$, but we prove a lower bound showing that this is necessary for any algorithm that produces a sparse histogram. A second algorithm avoids this lower bound, and matches the per-bin accuracy of the Laplace Mechanism, by producing a compact and efficiently computable representation of a dense histogram; it is based on an $(n+1)$-wise independent implementation of an appropriately clamped version of the Discrete Geometric Mechanism.

*Supported by NSF grant CNS-1237235 and CNS-1565387.

†http://seas.harvard.edu/~salil. Supported by NSF grant CNS-1237235, a Simons Investigator Award, and a grant from the Sloan Foundation.

1 Introduction

Differential Privacy [DMNS06] is by now a well-established framework for privacy-protective statistical analysis of sensitive datasets. Much work on differential privacy involves an interplay between statistics and computer science. Statistics provides many of the (non-private) analyses that we wish to approximate with differentially private algorithms, as well as probabilistic tools that are useful in analyzing such algorithms, which are necessarily randomized. From computer science, differential privacy draws upon a tradition of adversarial modeling and strong security definitions, techniques for designing and analyzing randomized algorithms, and considerations of algorithmic resource constraints (such as time and memory).

Because of its connection to statistics, it is very natural that much of the literature on differential privacy considers the estimation of real-valued functions on real-valued data (e.g. the sample mean) and introduces noise from continuous probability distributions (e.g. the Laplace distribution) to obtain privacy. However, these choices are incompatible with standard computer science models for algorithms (like the Turing machine or RAM model) as well as implementation on physical computers (which use only finite approximations to real arithmetic, e.g. via floating point numbers). This discrepancy is not just a theoretical concern; Mironov [Mir12] strikingly demonstrated that common floating-point implementations of the most basic differentially private algorithm (the Laplace Mechanism) are vulnerable to real attacks. Mironov shows how to prevent his attack with a simple modification to the implementation, but this solution is specific to a single differentially private mechanism and particular floating-point arithmetic standard. His solution increases the error by a constant factor and is most likely more efficient in practice than the algorithm we will use to replace the Laplace Mechanism. However, he provides no bounds on asymptotic running time. Gazeau, Miller and Palamidessi [GMP16] provide more general conditions for which an implementation of real numbers and a mechanism that perturbs the correct answer with noise maintains differential privacy. However, they do not provide an explicit construction with bounds on accuracy and running time.

From a theoretical point of view, a more appealing approach to resolving these issues is to avoid real or floating-point arithmetic entirely and only consider differentially private computations that involve discrete inputs and outputs, and rational probabilities. Such algorithms are realizable in standard discrete models of computation. However, some such algorithms have running times that are only bounded in expectation (e.g. due to sampling from an exponential distribution supported on the natural numbers), and this raises a potential vulnerability to timing attacks. If an adversary can observe the running time of the algorithm, it learns something about the algorithm's coin tosses, which are assumed to be secret in the definition of differential privacy. (Even if the time cannot be directly observed, in practice an adversary can determine an upper bound on the running time, which again is information that is implicitly assumed to be secret in the privacy definition.)

Because of these considerations, we advocate the following principle:

Differential Privacy for Finite Computers:

> *We should describe how to implement differentially private algorithms on* ***discrete*** *models of computation with* ***strict*** *bounds on running time (ideally polynomial in the bit length of the input) and* ***analyze*** *the effects of those constraints on both privacy and accuracy.*

Note that a strict *bound* on running time does not in itself prevent timing attacks, but once we have such a bound, we can pad all executions to take the same amount of time. Also, while standard discrete models of computation (e.g. randomized Turing machines) are defined in terms of countable

rather than finite resources (e.g. the infinite tape), if we have a strict bound on running time, then once we fix an upper bound on input length, they can indeed be implemented on a truly finite computer (e.g. like a randomized Boolean circuit).

In many cases, the above goal can be achieved by appropriate discretizations and truncations applied to a standard, real-arithmetic differentially private algorithm. However, such modifications can have a nontrivial price in accuracy or privacy, and thus we also call for a rigorous analysis of these effects.

In this paper, we carry out a case study of achieving "differential privacy for finite computers" for one of the first tasks studied in differential privacy, namely approximating a histogram of a categorical dataset. Even this basic problem turns out to require some nontrivial effort, particularly to maintain strict polynomial time, optimal accuracy and pure differential privacy when the data universe is large.

We recall the definition of differential privacy.

Definition 1.1. [DMNS06] Let $\mathcal{M} : \mathcal{X}^n \to \mathcal{R}$ be a randomized algorithm. We say $\mathcal{M}$ is (ε, δ)-**differentially private** if for every two datasets D and D' that differ on one row and every subset $S \subseteq \mathcal{R}$

$$\Pr[\mathcal{M}(D) \in S] \leq e^{\varepsilon} \cdot \Pr[\mathcal{M}(D') \in S] + \delta$$

We say an (ε, δ)-differentially private algorithm satisfies **pure differential privacy** when $\delta = 0$ and say it satisfies **approximate differential privacy** when $\delta > 0$.

In this paper, we study the problem of estimating the *histogram* of a dataset $D \in \mathcal{X}^n$, which is the vector $c = c(D) \in \mathbb{N}^{\mathcal{X}}$, where c_x is the number of rows in D that have value x. Histograms can be approximated while satisfying differential privacy using the *Laplace Mechanism*, introduced in the original paper of Dwork, McSherry, Nissim and Smith [DMNS06]. Specifically, to obtain $(\varepsilon, 0)$-differential privacy, we can add independent noise distributed according to a Laplace distribution, specifically $\mathrm{Lap}(2/\varepsilon)$, to each component of c and output the resulting vector $\tilde{c}$. Here $\mathrm{Lap}(2/\varepsilon)$ is the *continuous*, real-valued random variable with probability density function $f(z)$ that is proportional to $\exp(-\varepsilon \cdot |z|/2)$. The Laplace Mechanism also achieves very high accuracy in two respects:

Per-Query Error: For each bin x, with high probability we have $|\tilde{c}_x - c_x| \leq O(1/\varepsilon)$.

Simultaneous Error: With high probability, we have $\max_x |\tilde{c}_x - c_x| \leq O(\log(|\mathcal{X}|)/\varepsilon)$.

Note that both of the bounds are independent of the number n of rows in the dataset, and so the fractional error vanishes linearly as n grows.

Simultaneous error is the more well-studied notion in the differential privacy literature, but we consider per-query error to be an equally natural concept: if we think of the approximate histogram $\tilde{c}$ as containing approximate answers to the $|\mathcal{X}|$ different counting queries corresponding to the bins of $\mathcal{X}$, then per-query error captures the error as experienced by an analyst who may be only interested in one or a few of the bins of $\tilde{c}$. The advantage of considering per-query error is that it can be significantly smaller than the simultaneous error, as is the case in the Laplace Mechanism when the data universe $\mathcal{X}$ is very large. It is known that both of the error bounds achieved by the Laplace Mechanism are optimal up to constant factors; no $(\varepsilon, 0)$-differentially private algorithm for histograms can achieve smaller per-query error or simultaneous error [HT10, BBKN14].

Unfortunately, the Laplace Mechanism uses real arithmetic and thus cannot be implemented on a finite computer. To avoid real arithmetic, we could use the Geometric Mechanism [GRS12], which adds noise to each component of c according to the 2-sided geometric distribution, $\mathrm{Geo}(2/\varepsilon)$, which is supported on the integers and has probability mass function $f(z) \propto \exp(-\varepsilon \cdot |z|/2)$. However, this

mechanism uses integers of unbounded size and thus cannot be implemented on a finite computer. Indeed, while the algorithm can be implemented with a running time that is bounded in expectation (after reducing ε so that $e^{\varepsilon/2}$ and hence all the probabilities are rational numbers), truncating long executions or allowing an adversary to observe the actual running time can lead to a violation of differential privacy. Thus, it is better to work with the Truncated Geometric Mechanism of Ghosh, Roughgarden and Soundararajan [GRS12], where we clamp each noisy count $\tilde{c}_x$ to the interval $[0, n]$. We observe that the resulting probability distribution of $\tilde{c}_x$, supported on $\{0, 1, \ldots, n\}$, can be described explicitly in terms of c_x, ε and n, and it can be sampled in polynomial time using only integer arithmetic (after ensuring $e^{\varepsilon/2}$ is rational). Thus, we obtain:

Theorem 1.2 (Bounded Geometric Mechanism, informal statement of Thm. 3.7)**.** *For every finite $\mathcal{X}$, n and $\varepsilon \in (0, 1]$, there is an $(\varepsilon, 0)$-differentially private algorithm $\mathcal{M} : \mathcal{X}^n \to \{0, 1, \ldots, n\}^{\mathcal{X}}$ for histograms achieving:*

- *Per-query error $O(1/\varepsilon)$.*
- *Simultaneous error $O(\log |\mathcal{X}|)/\varepsilon$.*
- *Strict running time $|\mathcal{X}| \cdot \mathrm{poly}(N)$, where N is the bit length of the input (n, ε and a dataset $D \in \mathcal{X}^n$).*

We note that while we only consider our particular definition of per-query accuracy, namely that with high probability $|\tilde{c}_x - c_x| \leq O(1/\varepsilon)$, Ghosh et al. [GRS12] proved that the output of the Bounded Geometric Mechanism can be used (with post-processing) to get optimal expected loss with respect to an extremely general class of loss functions and arbitrary priors. The same result applies to each individual noisy count $\tilde{c}_x$ output by our mechanism, since each bin is distributed according to the Bounded Geometric Mechanism (up to a modification of ε to ensure rational probabilities).

The Bounded Geometric Mechanism is not polynomial time for large data universes $\mathcal{X}$. Indeed, its running time (and output length) is linear in $|\mathcal{X}|$, rather than polynomial in the bit length of data elements, which is $\log |\mathcal{X}|$. To achieve truly polynomial time, we can similarly discretize and truncate a variant of the Stability-Based Histogram of Bun, Nissim and Stemmer [BNS16]. This mechanism only adds $\mathrm{Lap}(2/\varepsilon)$ noise to the *nonzero* components of c_x and then retains only the noisy values $\tilde{c}_x$ that are larger than a threshold $t = \Theta(\log(1/\delta)/\varepsilon)$. Thus, the algorithm only outputs a partial histogram, i.e. counts $\tilde{c}_x$ for a subset of the bins x, with the rest of the counts being treated as zero. By replacing the use of the Laplace Mechanism with the (rational) Bounded Geometric Mechanism as above, we can implement this algorithm in strict polynomial time:

Theorem 1.3 (Stability-Based Histogram, informal statement of Thm. 5.2)**.** *For every finite $\mathcal{X}$, n, $\varepsilon \in (0, 1]$ and $\delta \in (0, 1/n)$, there is an (ε, δ)-differentially private algorithm $\mathcal{M} : \mathcal{X}^n \to \{0, 1, \ldots, n\}^{\subseteq \mathcal{X}}$ for histograms achieving:*

- *Per-query error $O(1/\varepsilon)$ on bins with true count at least $O(\log(1/\delta)/\varepsilon)$.*
- *Simultaneous error $O(\log(1/\delta)/\varepsilon)$.*
- *Strict running time $\mathrm{poly}(N)$, where N is the bit length of the input (n, ε and a dataset $D \in \mathcal{X}^n$).*

Notice that the simultaneous error bound of $O(\log(1/\delta)/\varepsilon)$ is better than what is achieved by the Laplace Mechanism when $\delta > 1/|\mathcal{X}|$, and is known to be optimal up to constant factors in this

range of parameters (see Theorem 6.1). The fact that this error bound is independent of the data universe size $|\mathcal{X}|$ makes it tempting to apply even for infinite data domains $\mathcal{X}$. However, we note that when $\mathcal{X}$ is infinite, it is impossible for the algorithm to have a strict bound on running time (as it needs time to read arbitrarily long data elements) and thus is vulnerable to timing attacks and is not implementable on a finite computer.

Note also that the per-query error bound only holds on bins with large enough true count (namely, those larger than our threshold t); we will discuss this point further below.

A disadvantage of the Stability-based Histogram is that it sacrifices pure differential privacy. It is natural to ask whether we can achieve polynomial running time while retaining pure differential privacy. A step in this direction was made by Cormode, Procopiuc, Srivastava and Tran [CPST11]. They observe that for an appropriate threshold $t = \Theta(\log(|\mathcal{X}|)/\varepsilon)$, if we run the Bounded Geometric Mechanism and only retain the noisy counts $\tilde{c}_x$ that are larger than t, then the expected number of bins that remain is less than $n+1$. Indeed, the expected number of bins we retain whose true count is zero ("empty bins") is less than 1. They describe a method to directly sample the distribution of the empty bins that are retained, without actually adding noise to all $|\mathcal{X}|$ bins. This yields an algorithm whose output length is polynomial in expectation. However, the output length is not strictly polynomial, as there is a nonzero probability of outputting all $|\mathcal{X}|$ bins. And it is not clear how to implement the algorithm in expected polynomial time, because even after making the probabilities rational, they have denominators of bit length linear in $|\mathcal{X}|$.

To address these issues, we consider a slightly different algorithm. Instead of trying to retain all noisy counts $\tilde{c}_x$ that are larger than some fixed threshold t, we retain the n largest noisy counts (since there are at most n nonzero true counts). This results in a mechanism whose output length is always polynomial, rather than only in expectation. However, the probabilities still have denominators of bit length linear in $|\mathcal{X}|$. Thus, we show how to approximately sample from this distribution, to within an arbitrarily small statistical distance δ, at the price of a $\mathrm{poly}(\log(1/\delta))$ increase in running time. Naively, this would result only in $(\varepsilon, O(\delta))$-differential privacy. However, when δ is significantly smaller than $1/|\mathcal{R}|$, where $\mathcal{R}$ is the range of the mechanism, we can convert an (ε, δ)-differentially private mechanism to an $(\varepsilon, 0)$-differentially private mechanism by simply outputting a uniformly random element of $\mathcal{R}$ with small probability. (A similar idea for the case that $|\mathcal{R}| = 2$ has been used in [KLN$^+$11, CDK17].) Since our range is of at most exponential size (indeed at most polynomial in bit length), the cost in our runtime for taking $\delta \ll 1/|\mathcal{R}|$ is at most polynomial. With these ideas we obtain:

Theorem 1.4 (Pure DP Histogram in Polynomial Time, informal statement of Thm. 4.14)**.** *For every finite $\mathcal{X}$, n and $\varepsilon \in (0, 2]$, there is an $(\varepsilon, 0)$-differentially private algorithm $\mathcal{M} : \mathcal{X}^n \to \{0, 1, \ldots, n\}^{\subseteq \mathcal{X}}$ for histograms achieving:*

- *Per-query error $O(1/\varepsilon)$ on bins with true count at least $O(\log(|\mathcal{X}|)/\varepsilon)$.*
- *Simultaneous error $O(\log(|\mathcal{X}|)/\varepsilon)$.*
- *Strict running time $\mathrm{poly}(N)$, where N is the bit length of the input (n, ε and a dataset $D \in \mathcal{X}^n$).*

Both Theorems 1.3 and 1.4 only retain per-query error $O(1/\varepsilon)$ on bins with a large enough true count. We also prove a lower bound showing that this limitation is inherent in any algorithm that outputs a sparse histogram (as both of these algorithms do).

Theorem 1.5 (Lower Bound on Per-Query Error for Sparse Histograms, Theorem 6.2)**.** *Suppose that there is an (ε, δ)-differentially private algorithm $\mathcal{M} : \mathcal{X}^n \to \{0, 1, \ldots, n\}^{\mathcal{X}}$ for histograms that*

always outputs histograms with at most n' nonempty bins and has per-query error at most E on all bins. Then

$$E \geq \Omega\left(\frac{\min\{\log|\mathcal{X}|, \log(1/\delta)\}}{\varepsilon}\right),$$

provided that $\varepsilon > 0$, $\varepsilon^2 > \delta > 0$ and $|\mathcal{X}| \geq (n')^2$.

This lower bound is similar in spirit to a lower bound of [BBKN14], which shows that no $(\varepsilon, 0)$-differentially private PAC learner for "point functions" (functions that are 1 on exactly one element of the domain) can produce sparse functions as hypotheses.

To bypass this lower bound, we can consider algorithms that produce succinct descriptions of dense histograms. That is, the algorithm can output a polynomial-length description of a function $\tilde{c} : \mathcal{X} \to [0, n]$ that can be evaluated in polynomial time, even though $\mathcal{X}$ may be of exponential size. We show that this relaxation allows us to regain per-query error $O(1/\varepsilon)$.

Theorem 1.6 (Polynomial-Time DP Histograms with Optimal Per-Query Accuracy, informal statement of Thm. 7.3)**.** *For every finite $\mathcal{X}$, n and $\varepsilon \in (0, 1]$, there is an $(\varepsilon, 0)$-differentially private algorithm $\mathcal{M} : \mathcal{X}^n \to \mathcal{H}$ for histograms (where $\mathcal{H}$ is an appropriate class of succinct descriptions of histograms) achieving:*

- *Per-query error $O(1/\varepsilon)$.*
- *Simultaneous error $O(\log(|\mathcal{X}|)/\varepsilon)$.*
- *Strict running time* $\mathrm{poly}(N)$*, where N is the bit length of the input (n, ε and a dataset $D \in \mathcal{X}^n$) for both producing the description of a noisy histogram $\tilde{c} \leftarrow \mathcal{M}(D)$ and for evaluating $\tilde{c}(x)$ at any point $x \in \mathcal{X}$.*

The algorithm is essentially an $(n+1)$-wise independent instantiation of the Bounded Geometric Mechanism. Specifically, we release a function $h : \mathcal{X} \to \{0,1\}^r$ selected from an $(n+1)$-wise independent family of hash functions, and for each $x \in \mathcal{X}$, we view $h(x)$ as coin tosses specifying a sample from the Bounded Geometric Distribution. That is, we let $S : \{0,1\}^r \to [0, n]$ be an efficient sampling algorithm for the Bounded Geometric Distribution, and then $\tilde{c}_x = S(h(x))$ is our noisy count for x. The hash function is chosen randomly from the family conditioned on values $\tilde{c}_x$ for the nonempty bins x, which we obtain by running the actual Bounded Geometric Mechanism on those bins. The $(n+1)$-wise independence ensures that the behavior on any two neighboring datasets (which together involve at most $n+1$ distinct elements of $\mathcal{X}$) are indistinguishable in the same way as in the ordinary Bounded Geometric Mechanism. The per-query accuracy comes from the fact that the marginal distributions of each of the noisy counts are the same as in the Bounded Geometric Mechanism. (Actually, we incur a small approximation error in matching the domain of the sampling procedure to the range of a family of hash functions.)

As far as we know, the only other use of limited independence in constructing differentially private algorithms is a use of pairwise independence by [BBKN14] in differentially private PAC learning algorithms for the class of point functions. Although that problem is related to the one we consider (releasing a histogram amounts to doing "query release" for the class of point functions, as discussed below), the design and analysis of our algorithm appears quite different. (In particular, our analysis seems to rely on $(n+1)$-wise independence in an essential way.)

Another potential interest in our technique is as another method for bypassing limitations of *synthetic data* for *query release*. Here, we have a large family of predicates $\mathcal{Q} = \{q : \mathcal{X} \to \{0,1\}\}$, and are interested in differentially private algorithms that, given a dataset $D = (x_1, \ldots, x_n) \in \mathcal{X}^n$, output a "summary" $\mathcal{M}(D)$ that allows one to approximate the answers to all of the *counting queries*

$q(D) = \sum_i q(x_i)$ associated with predicates $q \in \mathcal{Q}$. For example, if $\mathcal{Q}$ is the family of *point functions* consisting of all predicates that evaluate to 1 on exactly one point in the data universe $\mathcal{X}$, then this query release problem amounts to approximating the histogram of D. The fundamental result of Blum, Ligett, and Roth [BLR13] and successors show that this is possible even for families $\mathcal{Q}$ and data universes $\mathcal{X}$ that are of size exponential in n. Moreover, the summaries produced by these algorithms has the form of a synthetic dataset — a dataset $\hat{D} \in \mathcal{X}^{\hat{n}}$ such that for every query $q \in \mathcal{Q}$, we have $q(\hat{D}) \approx q(D)$. Unfortunately, it was shown in [UV11] that even for very simple families $\mathcal{Q}$ of queries, such correlations between pairs of binary attributes, constructing such a differentially private synthetic dataset requires time exponential in the bitlength $\log |\mathcal{X}|$ of data universe elements. Thus, it is important to find other ways of representing approximate answers to natural families $\mathcal{Q}$ of counting queries, which can bypass the inherent limitations of synthetic data, and progress along these lines was made in a variety of works [GRU12, CKKL12, HRS12, TUV12, CTUW14, DNT15]. Our algorithm, and its use of $(n+1)$-wise independence, can be seen as yet another representation that bypasses a limitation of synthetic data (albeit a statistical rather than computational one). Indeed, a sparse histogram is simply a synthetic dataset that approximates answers to all point functions, and by Theorem 1.5, our algorithm achieves provably better per-query accuracy than is possible with synthetic datasets. This raises the question of whether similar ideas can also be useful in bypassing the computational limitations of synthetic data for more complex families of counting queries.

2 Preliminaries

Throughout this paper, let $\mathbb{N}$ be the set $\{0, 1, \ldots\}$, $\mathbb{N}_+$ be the set $\{1, 2, \ldots\}$. For $n \in \mathbb{N}$, let $[n]$ be the nonstandard set $\{0, \ldots, n\}$. Notice that $|[n]| = n + 1$. Given a set A and finite set B, we define A^B to be the set of length $|B|$ vectors over A indexed by the elements of B.

2.1 Differential Privacy

We define a **dataset** $D \in \mathcal{X}^n$ to be an ordered tuple of $n \geq 1$ rows where each row is drawn from a discrete **data universe** $\mathcal{X}$ with each row corresponding to an individual. Two datasets $D, D' \in \mathcal{X}^n$ are considered **neighbors** if they differ in exactly one row.

Definition 2.1. [DMNS06] Let $\mathcal{M} : \mathcal{X}^n \to \mathcal{R}$ be a randomized algorithm. We say $\mathcal{M}$ is (ε, δ)-**differentially private** if for every pair of neighboring datasets D and D' and every subset $S \subseteq \mathcal{R}$

$$\Pr[\mathcal{M}(D) \in S] \leq e^{\varepsilon} \cdot \Pr[\mathcal{M}(D') \in S] + \delta$$

We say an (ε, δ)-differentially private algorithm satisfies **pure differential privacy** when $\delta = 0$ and say it satisfies **approximate differential privacy** when $\delta > 0$. Intuitively, the ε captures an upper bound on an adversary's ability to determine whether a particular individual is in the dataset. And the δ parameter represents an upper bound of the probability of a catastrophic privacy breach (e.g. the entire dataset is released). The common setting of parameters takes $\varepsilon \in (0, 1]$ to be a small constant and δ to be negligible in n.

The following properties of differentially private algorithms will be used in some of our proofs.

Lemma 2.2 (post-processing [DMNS06])**.** *Let $\mathcal{M} : \mathcal{X}^n \to \mathcal{Y}$ be (ε, δ)-differentially private and $f : \mathcal{Y} \to \mathcal{Z}$ be any randomized function. Then $f \circ \mathcal{M} : \mathcal{X}^n \to \mathcal{Z}$ is (ε, δ)-differentially private.*

Lemma 2.3 (group privacy [DMNS06]). *Let $\mathcal{M} : \mathcal{X}^n \to \mathcal{Y}$ be (ε, δ)-differentially private. Let $D_1, D_2 \subseteq \mathcal{X}^n$ be datasets such that D_2 can be obtained by changing at most m rows of D_1. Then for all $S \subseteq \mathcal{Y}$*

$$\Pr[\mathcal{M}(D_1) \in S] \leq e^{m\varepsilon} \cdot \Pr[\mathcal{M}(D_2) \in S] + e^{m\varepsilon} \cdot \delta/\varepsilon$$

Lemma 2.4 (composition [DL09]). *Let $\mathcal{M}_1 : \mathcal{X}^n \to \mathcal{Y}_1$ be $(\varepsilon_1, \delta_1)$-differentially private and $\mathcal{M}_2 : \mathcal{X}^n \to \mathcal{Y}_2$ be $(\varepsilon_2, \delta_2)$-differentially private. Define $\mathcal{M} : \mathcal{X}^n \to \mathcal{Y}_1 \times \mathcal{Y}_2$ as $\mathcal{M}(D) = (\mathcal{M}_1(D), \mathcal{M}_2(D))$ for all $D \in \mathcal{X}^n$. Then $\mathcal{M}$ is $(\varepsilon_1 + \varepsilon_2, \delta_1 + \delta_2)$-differentially private.*

2.2 Histograms

For $x \in \mathcal{X}$, the **point function** $c_x : \mathcal{X}^n \to \mathbb{N}$ is defined to count the number of occurrences of x in a given dataset, i.e. for $D \in \mathcal{X}^n$

$$c_x(D) = |\{i \in \{1, \dots n\} : D_i = x\}|$$

In this paper we focus on algorithms for privately releasing approximations to the values of all point functions, also known as a **histogram**. A histogram is collection of **bins**, one for each element x in the data universe, with the x^{th} bin consisting of its **label** x and a **count** $c_x \in \mathbb{N}$.

2.2.1 Representations

The input to our algorithms is always a dataset (i.e. an element $D \in \mathcal{X}^n$) and the outputs represent approximate histograms. We consider the following histogram representations as our algorithms' outputs:

- A vector in $\mathbb{N}^{\mathcal{X}}$. We use $\{\tilde{c}_x\}_{x \in \mathcal{X}}$ to denote a histogram where $\tilde{c}_x \in \mathbb{N}$ is the approximate count for the element x.
- A partial vector $h \in (\mathcal{X} \times \mathbb{N})^*$ such that each element $x \in \mathcal{X}$ appears at most once in h with each pair $(x, \tilde{c}_x) \in \mathcal{X} \times \mathbb{N}$ interpreted as element x having approximate count $\tilde{c}_x$. Elements x not listed in the partial vector are assumed to have count $\tilde{c}_x = 0$. Implicitly an algorithm can return a partial vector by releasing bins for a subset of $\mathcal{X}$.
- A data structure, encoded as a string, which defines a function $h : \mathcal{X} \to \mathbb{N}$ where $h(x)$, denoted h_x, is the approximate count for $x \in \mathcal{X}$ and h_x is efficiently computable given this data structure (e.g. time polynomial in the length of the data structure). In Section 7, this data structure consists of the coefficients of a polynomial, along with some parameters.

Each representation is able to express any histogram over $\mathcal{X}$. The difference between them is the memory used and the efficiency of computing a count. For example, computing the approximate count for $x \in \mathcal{X}$, when using the data structure representation is bounded by the time it takes to compute the associated function. But when using partial vectors, one only needs to iterate through the vector to determine the approximate count.

We define the following class of histograms. Let $\mathcal{H}_{n,n'}(\mathcal{X}) \subseteq \mathbb{N}^{\mathcal{X}}$ be the set of all histograms over $\mathcal{X}$ with integer counts in $[0, n]$ (or $\mathbb{N}$ when $n = \infty$) and at most n' of them nonzero. By using partial vectors each element of $\mathcal{H}_{n,n'}(\mathcal{X})$ can be stored in $O(n' \cdot (\log n + \log |\mathcal{X}|))$ bits, which is shorter than the vector representation when $n' = o(|\mathcal{X}|/\log |\mathcal{X}|)$.

2.2.2 Accuracy

In order to preserve privacy, our algorithms return histograms with noise added to the counts. Therefore, it is crucial to understand their accuracy guarantees. So given a dataset $D \in \mathcal{X}^n$ we compare the **noisy count** $\tilde{c}_x = \mathcal{M}(D)_x$ of $x \in \mathcal{X}$ (the count released by algorithm $\mathcal{M}$) to its **true count**, $c_x(D)$. We focus on the following two metrics:

Definition 2.5. A histogram algorithm $\mathcal{M} : \mathcal{X}^n \to \mathbb{N}^{\mathcal{X}}$ has (a, β)**-per-query accuracy** if

$$\forall D \in \mathcal{X}^n \quad \forall x \in \mathcal{X} \qquad \Pr[|\mathcal{M}(D)_x - c_x(D)| \leq a] \geq 1 - \beta$$

Definition 2.6. A histogram algorithm $\mathcal{M} : \mathcal{X}^n \to \mathbb{N}^{\mathcal{X}}$ has (a, β)**-simultaneous accuracy** if

$$\forall D \in \mathcal{X}^n \qquad \Pr[\forall x \in \mathcal{X} \quad |\mathcal{M}(D)_x - c_x(D)| \leq a] \geq 1 - \beta$$

Respectively, these metrics capture the maximum error for any one bin and the maximum error simultaneously over all bins. Even though simultaneous accuracy is commonly used in differential privacy, per-query accuracy has several advantages:

- For histograms, one can provable achieve a smaller per-query error than is possible for simultaneous error. Indeed, the optimal simultaneous error for $(\varepsilon, 0)$-differentially private histograms is $a = \Theta\left(\log(|\mathcal{X}|/\beta)/\varepsilon\right)$ whereas the optimal per-query error is $a = \Theta\left(\log(1/\beta)/\varepsilon\right)$, which is independent of $|\mathcal{X}|$ [HT10, BBKN14].
- Per-query accuracy may be easier to convey to an end user of differential privacy. For example, it is the common interpretation of error bars shown on a graphical depiction of a histogram.

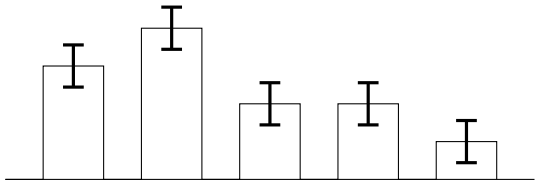

Figure 1: A histogram with error bars

- For many algorithms (such as ours), per-query accuracy is good enough to imply optimal simultaneous accuracy. Indeed, an algorithm with (a, β)-per-query accuracy also achieves $(a, \beta \cdot |\mathcal{X}|)$-simultaneous accuracy (by a union bound).

However, we may not always be able to achieve as good per-query accuracy as we want. So we will also use the following relaxation which bounds the error only on bins with large enough true count.

Definition 2.7. A histogram algorithm $\mathcal{M} : \mathcal{X}^n \to \mathbb{N}^{\mathcal{X}}$ has (a, β)**-per-query accuracy on counts larger than** t if

$$\forall D \in \mathcal{X}^n \quad \forall x \in \mathcal{X} \text{ s.t. } c_x(D) > t \qquad \Pr[|\mathcal{M}(D)_x - c_x(D)| \leq a] \geq 1 - \beta$$

2.3 Probability Terminology

Definition 2.8. Let Z be an integer-valued random variable.

1. The **probability mass function of** Z, denoted f_Z, is the function $f_Z(z) = \Pr[Z = z]$ for all $z \in \mathbb{Z}$.

2. The **cumulative distribution function of** Z, denoted F_Z, is the function $F_Z(z) = \Pr[Z \leq z]$ for all $z \in \mathbb{Z}$.

3. The **support** of Z, denoted $\text{supp}(Z)$, is the set of elements for which $f(z) \neq 0$.

Definition 2.9. Let Y and Z be random variables taking values in discrete range $\mathcal{R}$. The **total variation distance between** Y **and** Z is defined as

$$\begin{aligned}\Delta(Y,Z) &= \max_{A \subseteq \mathcal{R}} \big| \Pr[Y \in A] - \Pr[Z \in A] \big| \\ &= \frac{1}{2} \cdot \sum_{a \in \mathcal{R}} \big| \Pr[Z = a] - \Pr[Y = a] \big|\end{aligned}$$

Lemma 2.10. *Let Y and Z be random variables over discrete range $\mathcal{R}$. Total variation distance has the following properties:*

1. *Y and Z are identically distributed, denoted $Y \sim Z$, if and only if $\Delta(Y,Z) = 0$.*

2. *Let $T : \mathcal{R} \to \mathcal{R}'$ be any function with $\mathcal{R}'$ discrete. Then*

$$\Delta(T(Y), T(Z)) \leq \Delta(Y,Z)$$

3. *Let Y_1, Y_2, Z_1 and Z_2 be random variables over discrete range $\mathcal{R}$. Then*

$$\Delta((Y_1,Y_2),(Z_1,Z_2)) \leq \Delta(Y_1,Z_1) + \max_{a \in \mathcal{R}, A \subseteq \mathcal{R}} \big| \Pr[Y_2 \in A \mid Y_1 = a] - \Pr[Z_2 \in A \mid Z_1 = a] \big|$$

2.3.1 Sampling

Because we are interested in the computational efficiency of our algorithms we need to consider the efficiency of sampling from various distributions.

A standard method for sampling a random variable is via **inverse transform sampling**.

Lemma 2.11. *Let U be uniformly distributed on $(0,1]$. Then for any integer-valued random variable Z we have $F_Z^{-1}(U) \sim Z$ where $F_Z^{-1}(u)$ is defined as $\min\{z \in \text{supp}(Z) : F_Z(z) \geq u\}$.*

If Z, the random variable we wish to sample, has finite support we can compute the inverse cumulative distribution by performing binary search on $\text{supp}(Z)$ to find the minimum. This method removes the need to compute the inverse function of the cumulative distribution function and is used in some of our algorithms.

2.3.2 Order Statistics

Definition 2.12. Let $Z_1, \ldots, Z_\ell$ be integer-valued random variables. The i**-th order statistic of** $Z_1, \ldots, Z_\ell$ denoted $Z_{(i)}$ is the i-th smallest value among $Z_1, \ldots, Z_\ell$.

Lemma 2.13. *Let $Z_1, \ldots, Z_\ell$ be i.i.d. integer-valued random variables with cumulative distribution function F. Then*

$$F_{Z_{(\ell)}}(z) = (F(z))^\ell$$

and

$$F_{Z_{(i)} | Z_{(i+1)} = v_{i+1}, \ldots Z_{(\ell)} = v_\ell}(z) = F_{Z_{(i)} | Z_{(i+1)} = v_{i+1}}(z) = \begin{cases} 1 & \text{if } z > v_{i+1} \\ \big(F(z)/F(v_{i+1})\big)^i & \text{otherwise} \end{cases}$$

for all $1 \leq i < \ell$ and $v_{i+1} \leq v_{i+2} \leq \ldots \leq v_\ell$ all in the support of Z_1.

From this lemma, we can iteratively sample random variables distributed identically to $Z_{(\ell)}$, $Z_{(\ell-1)}, \ldots, Z_{(i)}$ without having to sample all ℓ of the original random variables. The inverse cumulative distributions for the order statistics are

$$F^{-1}_{Z_{(\ell)}}(u) = F^{-1}\left(u^{1/\ell}\right) \qquad F^{-1}_{Z_{(i)}|Z_{(i+1)}=v_{i+1},\ldots Z_{(\ell)}=v_\ell}(u) = F^{-1}\left(u^{1/i} \cdot F(v_{i+1})\right)$$

2.4 Two-Sided Geometric Distribution

A common technique for creating differentially private algorithms is to perturb the desired output with appropriately scaled Laplace noise. Because our algorithms' outputs are counts, we focus on a discrete analogue of the Laplace distribution as in [GRS12].

We say an integer-valued random variable Z follows a **two-sided geometric distribution with scale parameter s centered at $c \in \mathbb{Z}$** (denoted $Z \sim c + \mathrm{Geo}(s)$) if its probability mass function $f_Z(z)$ is proportional to $e^{-|z-c|/s}$. It can be verified that f_Z and its cumulative distribution function F_Z are

$$f_Z(z) = \left(\frac{e^{1/s}-1}{e^{1/s}+1}\right) \cdot e^{-|z-c|/s} \qquad F_Z(z) = \begin{cases} \frac{e^{1/s}}{e^{1/s}+1} \cdot e^{-(c-z)/s} & \text{if } z \le c \\ 1 - \frac{1}{e^{1/s}+1} \cdot e^{-(z-c)/s} & \text{otherwise} \end{cases}$$

for all $z \in \mathbb{Z}$. When c is not specified, it is assumed to be 0. The inverse cumulative distribution of Z is

$$F^{-1}_Z(u) = c + \begin{cases} \left\lceil s\ln(u) + s\ln\left(e^{1/s}+1\right)\right\rceil - 1 & \text{if } u \le 1/2 \\ \left\lceil -s\ln(1-u) - s\ln\left(e^{1/s}+1\right)\right\rceil & \text{otherwise} \end{cases}$$

or, equivalently,

$$F^{-1}_Z(u) = c + \left\lceil s \cdot \mathrm{sign}(1/2-u)\left(\ln(1-|2u-1|) + \ln\left(\frac{e^{1/s}+1}{2}\right)\right)\right\rceil + \lfloor 2u \rfloor - 1$$

2.5 Model of Computation

We analyze the running time of our algorithms with respect to the **w-bit word RAM model** taking $w = O(\log N)$ where N is the bit length of our algorithms' inputs ($D \in \mathcal{X}^n$, ε and possibly some additional parameters). In this model, memory accesses and basic operations (arithmetic, comparisons and logical) on w-bit words are constant time. In addition, we assume the data universe $\mathcal{X} = [m]$ for some $m \in \mathbb{N}$. Some parameters to our algorithms are rational. We represent rationals by pairs of integers.

Because our algorithms require randomness, we assume that they have access to an oracle that when given a number $d \in \mathbb{N}_+$ returns a uniformly random integer between 1 and d inclusively.

3 The Geometric Mechanism

In this section we show how to construct a differentially private histogram using the Laplace Mechanism only requiring integer computations of bounded length.

As shown by Dwork, McSherry, Nissim and Smith [DMNS06], we can privately release a histogram by adding independent and appropriately scaled Laplace noise to each bin. Below we state a variant that uses discrete noise, formally studied in [GRS12].

Algorithm 3.1. `GeometricMechanism`(D, ε) for $D \in \mathcal{X}^n$ and $\varepsilon > 0$

1. For each $x \in \mathcal{X}$, do the following:

   (a) Set $\tilde{c}_x$ to $c_x(D) + \text{Geo}(2/\varepsilon)$ clamped to the interval $[0, n]$. i.e.

$$\tilde{c}_x = \begin{cases} 0 & \text{if } Z_x \leq 0 \\ n & \text{if } Z_x \geq n \\ Z_x & \text{otherwise} \end{cases} \qquad \text{where } Z_x = c_x(D) + \text{Geo}(2/\varepsilon).$$

   (b) Release $(x, \tilde{c}_x)$.

Note that the output of this algorithm is a collection of bins $(x, \tilde{c}_x)$ which represents a partial vector, but in this case we have a count for each $x \in \mathcal{X}$ so it defines a complete vector in $\mathcal{H}_{n,|\mathcal{X}|}(\mathcal{X}) \subseteq \mathbb{N}^{\mathcal{X}}$. The privacy and accuracy properties of the algorithm are similar to those of the Laplace Mechanism.

Theorem 3.2. `GeometricMechanism`(D, ε) *has the following properties:*

i. `GeometricMechanism`(D, ε) *is* $(\varepsilon, 0)$*-differentially private [GRS12].*

ii. `GeometricMechanism`(D, ε) *has* (a, β)*-per-query accuracy for*

$$a = \left\lceil \frac{2}{\varepsilon} \ln \frac{1}{\beta} \right\rceil$$

iii. `GeometricMechanism`(D, ε) *has* (a, β)*-simultaneous accuracy for*

$$a = \left\lceil \frac{2}{\varepsilon} \ln \left(\frac{1}{1 - (1-\beta)^{1/|\mathcal{X}|}} \right) \right\rceil \leq \left\lceil \frac{2}{\varepsilon} \ln \frac{|\mathcal{X}|}{\beta} \right\rceil$$

Proof of ii-iii. Let $Z \sim \text{Geo}(2/\varepsilon)$. For any $x \in \mathcal{X}$

$$\begin{aligned} \Pr[|\tilde{c}_x - c_x(D)| \leq a] &\geq \Pr\left[|Z| \leq \lfloor a \rfloor\right] \\ &= 1 - 2 \cdot \Pr[Z \leq -\lfloor a \rfloor - 1] \\ &= 1 - 2 \cdot \frac{e^{-\lfloor a \rfloor \cdot \varepsilon/2}}{e^{\varepsilon/2} + 1} \end{aligned}$$

Now, for $a = \left\lceil \frac{2}{\varepsilon} \ln \frac{1}{\beta} \right\rceil$

$$\Pr[|\tilde{c}_x - c_x(D)| \leq a] \geq 1 - \frac{2 \cdot \beta}{e^{\varepsilon/2} + 1} \geq 1 - \beta$$

Part *iii* follows similarly after noting by independence of the counts that

$$\Pr[\forall x \in \mathcal{X} \;\; |\tilde{c}_x - c_x(D)| \leq a] \geq \Pr[|Z| \leq \lfloor a \rfloor]^{|\mathcal{X}|}$$

□

The accuracy bounds up to constant factors match the lower bounds for releasing a differentially private histogram [HT10, BBKN14].

As presented above, this algorithm needs to store integers of unbounded size since $\mathrm{Geo}(2/\varepsilon)$ is unbounded in magnitude. As noted in [GRS12], by restricting the generated noise to a fixed range we can avoid this problem. However, even when the generated noise is restricted to a fixed range, generating this noise via inverse transform sampling may require infinite precision. By appropriately choosing ε, the probabilities of this noise's cumulative distribution function can be represented with finite precision, and therefore generating this noise via inverse transform sampling only requires finite precision.

Proposition 3.3. *For $k \in \mathbb{N}$, $n \in \mathbb{N}_+$ and $c \in [n]$, the algorithm* `GeoSample`(k, n, c) *has output identically distributed to a two-sided geometric random variable with scale parameter $2/\tilde{\varepsilon}$ centered at c clamped to the range $[0, n]$ where we define $\tilde{\varepsilon} = 2 \cdot \ln\left(1 + 2^{-k}\right)$. Morever,* `GeoSample`$(k, n, c)$ *has running time* $\mathrm{poly}(k, n)$.

We have chosen $\tilde{\varepsilon}$ so that the cumulative distribution function of a two-sided geometric random variable with scale parameter $2/\tilde{\varepsilon}$ clamped to $[0, n]$ takes on only rational values with a common denominator d. Therefore, to implement inverse transform sampling on this distribution we only need to choose a uniformly random integer in $\{1, \ldots, d\}$ rather than a uniformly random variable over $(0, 1]$.

Algorithm 3.4. `GeoSample`(k, n, c) for $k \in \mathbb{N}$, $n \in \mathbb{N}_+$ and $c \in [n]$

1. Let $d = (2^{k+1} + 1)(2^k + 1)^{n-1}$.
2. Define the function
$$F(z) = \begin{cases} 2^{k(c-z)} \left(2^k + 1\right)^{n+z-c} & \text{if } 0 \le z \le c \\ d - 2^{k(z-c+1)} \left(2^k + 1\right)^{n-1-z+c} & \text{if } c < z < n \\ d & \text{if } z = n \end{cases}$$
3. Sample U uniformly at random from $\{1, \ldots, d\}$.
4. Using binary search find the smallest $z \in [n]$ such that $F(z) \ge U$.
5. Return z.

The function F is obtained by clearing denominators in the cumulative distribution function of $c + \mathrm{Geo}(2/\tilde{\varepsilon})$ clamped to $[0, n]$.

Lemma 3.5. *Let $F(z)$ be defined as in Algorithm 3.4. Then for $z \in [n]$, $F(z) \in [d]$ and $F(z)/d$ equals the cumulative distribution function of $c + Geo(2/\tilde{\varepsilon})$ clamped to $[0, n]$.*

We prove this lemma after seeing how it implies Proposition 3.3.

Proof of Proposition 3.3. Let U be drawn uniformly at random from $\{1, \ldots, d\}$. By construction, for all $z \in [n]$

$$\Pr[\texttt{GeoSample}(k, n, c) \le z] = \Pr[U \le F(z)] = F(z)/d$$

implying $\texttt{GeoSample}(k, n, c) \sim c + \text{Geo}(2/\tilde{\varepsilon})$ by Lemma 3.5.

We now bound the running time. Binary search takes $O(\log n)$ rounds. The largest number used is d with bit length $O(nk)$ and all operations are polynomial in the bit length of these numbers. Therefore, $\texttt{GeoSample}(k, n, c)$ has running time $\text{poly}(k, n)$. □

Proof of Lemma 3.5. The cumulative distribution function of $Z \sim c + \text{Geo}(2/\tilde{\varepsilon})$ is

$$F_Z(z) = \begin{cases} 0 & \text{if } z < 0 \\ \frac{e^{\tilde{\varepsilon}/2}}{e^{\tilde{\varepsilon}/2}+1} \cdot e^{-(c-z)\cdot\tilde{\varepsilon}/2} & \text{if } 0 \le z \le c \\ 1 - \frac{1}{e^{\tilde{\varepsilon}/2}+1} \cdot e^{(c-z)\cdot\tilde{\varepsilon}/2} & \text{if } c < z < n \\ 1 & \text{if } z \ge n \end{cases}$$

Consider the case when $0 \le z \le c$.

$$\begin{aligned} F_Z(z) = \frac{e^{\tilde{\varepsilon}/2}}{e^{\tilde{\varepsilon}/2}+1} \cdot e^{-(c-z)\tilde{\varepsilon}/2} &= \frac{1+2^{-k}}{2+2^{-k}} \cdot \left(1+2^{-k}\right)^{-(c-z)} \\ &= \frac{2^k+1}{2^{k+1}+1} \cdot \left(\frac{2^k}{2^k+1}\right)^{c-z} \\ &= \frac{2^{k(c-z)}}{\left(2^{k+1}+1\right)\left(2^k+1\right)^{c-z-1}} \cdot \left(\frac{2^k+1}{2^k+1}\right)^{n-(c-z)} \\ &= \frac{2^{k(c-z)}\left(2^k+1\right)^{n+z-c}}{d} \\ &= \frac{F(z)}{d} \end{aligned}$$

A similar argument holds when $c < z < n$ and $F_Z(n) = 1 = F(n)/d$. So $F_Z(z) = F(z)/d$ for all $z \in [n]$. □

Using this algorithm we are ready to construct a private histogram algorithm with bounded time complexity whose accuracy is identical to that of $\texttt{GeometricMechanism}$ up to constant factors.

Algorithm 3.6. $\texttt{BoundedGeometricMechanism}(D, \varepsilon)$ for $D \in \mathcal{X}^n$ and rational $\varepsilon \in (0, 1]$

1. Let $k = \lceil \log(2/\varepsilon) \rceil$.
2. For each $x \in \mathcal{X}$, do the following:
   (a) Let $\tilde{c}_x = \texttt{GeoSample}(k, n, c_x(D))$.
   (b) Release $(x, \tilde{c}_x)$.

Theorem 3.7. *Let rational $\varepsilon \in (0, 1]$ and $\tilde{\varepsilon} = 2 \cdot \ln\left(1 + 2^{-\lceil \log(2/\varepsilon) \rceil}\right) \in (4/9 \cdot \varepsilon, \varepsilon]$. Then* $\texttt{BoundedGeometricMechanism}(D, \varepsilon)$ *has the following properties:*

i. $\texttt{BoundedGeometricMechanism}(D, \varepsilon)$ *is $(\varepsilon, 0)$-differentially private.*

ii. $\texttt{BoundedGeometricMechanism}(D, \varepsilon)$ *has (a, β)-per-query accuracy for*

$$a = \left\lceil \frac{2}{\tilde{\varepsilon}} \ln \frac{1}{\beta} \right\rceil \le \left\lceil \frac{9}{2\varepsilon} \ln \frac{1}{\beta} \right\rceil$$

iii. $\texttt{BoundedGeometricMechanism}(D, \varepsilon)$ *has* (a, β)*-simultaneous accuracy for*

$$a = \left\lceil \frac{2}{\tilde{\varepsilon}} \ln \frac{|\mathcal{X}|}{\beta} \right\rceil \leq \left\lceil \frac{9}{2\varepsilon} \ln \frac{|\mathcal{X}|}{\beta} \right\rceil$$

iv. $\texttt{BoundedGeometricMechanism}(D, \varepsilon)$ *has running time*

$$|\mathcal{X}| \cdot \text{poly}(N)$$

where N *is the bit length of the algorithm's input* ($D \in \mathcal{X}^n$ *and* ε*).*

Proof of i-iii. By Proposition 3.3, $\texttt{GeoSample}(k, n, c_x(D))$ generates a two-sided geometric random variable with scale parameter $2/\tilde{\varepsilon}$ centered at $c_x(D)$ clamped to $[0, n]$. So this algorithm is identically distributed to $\texttt{GeometricMechanism}(D, \tilde{\varepsilon})$. As $\tilde{\varepsilon} \in (4/9 \cdot \varepsilon, \varepsilon]$, parts *i-iii* follows from Theorem 3.2. □

Proof of iv. For each $x \in \mathcal{X}$, computing $c_x(D)$ takes time $O(n \log |\mathcal{X}|)$ and by Proposition 3.3 $\texttt{GeoSample}(k, n, c_x(D))$ takes times $\text{poly}(n, \log(1/\varepsilon))$ as $k = O(\log(1/\varepsilon))$. □

4 Improving the Running Time

For datasets over large domains $\mathcal{X}$, the linear in $|\mathcal{X}|$ running time of Algorithm 3.6 can be prohibitive. We present an algorithm that reduces the running time's dependence on the universe size from nearly linear to poly-logarithmic based on the observation that most counts are 0 when $n \ll |\mathcal{X}|$; this is the same observation made by Cormode, Procopiuc, Srivastava and Tran [CPST11] to output sparse histograms.

4.1 Sparse Histograms

We start by reducing the output length of $\texttt{GeometricMechanism}$ to release only the bins with the heaviest (or largest) counts (interpreted as a partial vector).

Algorithm 4.1. $\texttt{KeepHeavy}(D, \varepsilon)$ for $D \in \mathcal{X}^n$ and $\varepsilon > 0$

1. For each $x \in \mathcal{X}$, set $\tilde{c}_x$ to $c_x(D) + \text{Geo}(2/\varepsilon)$ clamped to $[0, n]$.
2. Let $x_1, \ldots, x_{n+1}$ be the elements of $\mathcal{X}$ with the largest counts in sorted order, i.e.

$$\tilde{c}_{x_1} \geq \tilde{c}_{x_2} \geq \ldots \geq \tilde{c}_{x_{n+1}} \geq \max_{x \in \mathcal{X} \setminus \{x_1, \ldots, x_{n+1}\}} \tilde{c}_x$$

3. Release $h = \{(x, \tilde{c}_x) : \tilde{c}_x > \tilde{c}_{x_{n+1}}\} \in \mathcal{H}_{n,n}(\mathcal{X})$.

Observe that the output length has been improved to $O(n \cdot (\log n + \log |\mathcal{X}|))$ bits compared to the $O(|\mathcal{X}| \cdot \log n)$ bits needed to represent the outputs of $\texttt{GeometricMechanism}$.

Theorem 4.2. $\texttt{KeepHeavy}(D, \varepsilon)$ *has the following properties:*

i. $\texttt{KeepHeavy}(D, \varepsilon)$ *is* $(\varepsilon, 0)$*-differentially private.*

ii. $\texttt{KeepHeavy}(D, \varepsilon)$ *has* (a, β)*-per-query accuracy on counts larger than* t *for*

$$t = 2\left\lceil \frac{2}{\varepsilon} \ln \frac{2|\mathcal{X}|}{\beta} \right\rceil \quad \textit{and} \quad a = \left\lceil \frac{2}{\varepsilon} \ln \frac{2}{\beta} \right\rceil$$

iii. $\texttt{KeepHeavy}(D, \varepsilon)$ *has* (a, β)*-simultaneous accuracy for*

$$a = 2\left\lceil \frac{2}{\varepsilon} \ln \frac{|\mathcal{X}|}{\beta} \right\rceil$$

Note that, unlike `GeometricMechanism`, this algorithm only has $(O(\log(1/\beta)/\varepsilon), \beta)$-per-query accuracy on counts larger than $t = O(\log(|\mathcal{X}|/\beta)/\varepsilon)$. This loss is necessary for any algorithm that outputs a sparse histogram as we will show in Theorem 6.2.

Proof of i. Privacy follows from the $(\varepsilon, 0)$-differential privacy of `GeometricMechanism` (part i of Theorem 3.2) along with differential privacy's closure under post-processing (Lemma 2.2). □

To prove the remaining parts, we start with the following lemma.

Lemma 4.3. *For any* $\beta' \in (0, 1)$*, let* $t' = 2\left\lceil \frac{2}{\varepsilon} \ln \frac{|\mathcal{X}|}{\beta'} \right\rceil$ *and define the event*

$$E_{t'} = \{\forall x \in \mathcal{X} \;\; |\tilde{c}_x - c_x(D)| \leq t'/2\}$$

$\Pr[E_{t'}] \geq 1 - \beta'$ *and* $E_{t'}$ *implies that for all* $x \in \mathcal{X}$ *such that* $c_x(D) > t'$ *we have* $\tilde{c}_x > \tilde{c}_{x_{n+1}}$.

Proof. The probability of $E_{t'}$ occurring follows from part *iii* of Theorem 3.2 as the $\{\tilde{c}_x\}_{x \in \mathcal{X}}$ are identically distributed to the output of $\texttt{GeometricMechanism}(D, \varepsilon)$.

Assume the event $E_{t'}$. Then for all $x \in \mathcal{X}$ such that $c_x(D) > t'$, we have $\tilde{c}_x > t'/2$ and for all $x \in \mathcal{X}$ such that $c_x(D) = 0$, we have $\tilde{c}_x \leq t'/2$. Because there are at most n distinct elements in D, we have $\tilde{c}_x > \tilde{c}_{x_{n+1}}$ for all $x \in \mathcal{X}$ such that $c_x(D) > t'$. □

Proof of part ii of Theorem 4.2. Let $x \in \mathcal{X}$ such that $c_x(D) > t$. We have

$$\begin{aligned} \Pr\left(|h_x - c_x(D)| > a\right) &\leq \Pr\left(\tilde{c}_x \leq \tilde{c}_{x_{n+1}}\right) + \Pr\left(|\tilde{c}_x - c_x(D)| > a\right) \\ &\leq \beta/2 + \beta/2 \end{aligned}$$

by Lemma 4.3 with $\beta' = \beta/2$ and $t' = t$, and part *ii* of Theorem 3.2. □

Proof of part iii of Theorem 4.2. Let $t = 2\lceil (2/\varepsilon) \cdot \ln(|\mathcal{X}|/\beta) \rceil$. The event E_t in Lemma 4.3 occurs with probability at least $1 - \beta$. Assume E_t. By Lemma 4.3, for all $x \in \mathcal{X}$ such that $c_x(D) > t$ we have $\tilde{c}_x > \tilde{c}_{x_{n+1}}$. This implies $|h_x - c_x(D)| \leq t/2$. For the remaining $x \in \mathcal{X}$ we trivially have $|h_x - c_x(D)| \leq t$ as $h_x = 0$. □

However, as described `KeepHeavy` still requires adding noise to the count of every bin. The following algorithm $\mathcal{M}_1 : \mathcal{X}^n \times (0, 1] \to \mathcal{H}_{n,n}(\mathcal{X})$ simulates `KeepHeavy` by generating a candidate set of heavy bins from which only the heaviest are released. This candidate set is constructed from all bins with nonzero true count and a sample representing the bins with a true count of 0 that have the heaviest noisy counts.

Algorithm 4.4. $\mathcal{M}_1(D, \varepsilon)$ for $D \in \mathcal{X}^n$, $\varepsilon > 0$ and $|\mathcal{X}| \geq 2n + 1$ [1]

1. Let $A = \{x \in \mathcal{X} : c_x(D) > 0\}$ and $w = |\mathcal{X} \setminus A|$.
2. For each $x \in A$, set $\tilde{c}_x$ to $c_x(D) + \text{Geo}(2/\varepsilon)$ clamped to $[0, n]$.
3. Pick a uniformly random sequence $(q_0, \ldots, q_n)$ of distinct elements from $\mathcal{X} \setminus A$.
4. Sample $(\tilde{c}_{q_0}, \ldots, \tilde{c}_{q_n})$ from the joint distribution of the order statistics $(Z_{(w)}, \ldots, Z_{(w-n)})$ where $Z_1, \ldots, Z_w$ are i.i.d. and distributed as $\text{Geo}(2/\varepsilon)$ clamped to $[0, n]$.
5. Sort the elements of $A \cup \{q_0, \ldots, q_n\}$ as $x_1, \ldots, x_{|A|+n+1}$ such that $\tilde{c}_{x_1} \geq \ldots \geq \tilde{c}_{x_{|A|+n+1}}$.
6. Release $h = \{(x, \tilde{c}_x) : x \in \{x_1, \ldots, x_n\} \text{ and } \tilde{c}_x > \tilde{c}_{x_{n+1}}\} \in \mathcal{H}_{n,n}(\mathcal{X})$.[2]

Proposition 4.5. *$\mathcal{M}_1(D, \varepsilon)$ is identically distributed to* `KeepHeavy`(D, ε).

Proof. Let $\{\hat{c}_x\}_{x \in \mathcal{X}}$ be the noisy counts set by `KeepHeavy`(D, ε) and let $\hat{x}_1, \ldots, \hat{x}_{n+1}$ be the sorted ordering defined by these counts. We have $\tilde{c}_x \sim \hat{c}_x$ for all $x \in A$ and the Z_i's are identically distributed to $\{\hat{c}_x\}_{x \in \mathcal{X} \setminus A}$.

$\{(q_i, \tilde{c}_{q_i})\}_{i=0}^n$ is identically distributed to the $n + 1$ bins with heaviest counts of $\{(x, \hat{c}_x)\}_{x \in \mathcal{X} \setminus A}$. Let the random variable B be the set of the labels of the $n + 1$ bins with heaviest counts of $\{(x, \hat{c}_x)\}_{x \in \mathcal{X} \setminus A}$. Therefore,

$$\begin{aligned} h &= \{(x, \tilde{c}_x) : x \in \{x_1, \ldots, x_n\} \text{ and } \tilde{c}_x > \tilde{c}_{x_{n+1}}\} \\ &= \{(x, \tilde{c}_x) : x \in A \cup \{q_0, \ldots, q_n\} \text{ and } \tilde{c}_x \geq \tilde{c}_{x_{n+1}}\} \\ &\sim \{(x, \hat{c}_x) : x \in A \cup B \text{ and } \hat{c}_x \geq \hat{c}_{\hat{x}_{n+1}}\} \\ &= \{(x, \hat{c}_x) : x \in \mathcal{X} \text{ and } \hat{c}_x \geq \hat{c}_{\hat{x}_{n+1}}\} \end{aligned}$$

which shows that $\mathcal{M}_1(D, \varepsilon)$ is identically distributed to `KeepHeavy`(D, ε). □

In order to sample from the order statistics used by $\mathcal{M}_1$ we construct the following algorithm similar to `GeoSample` (Algorithm 3.4) from the previous section.

Proposition 4.6. *Let $k \in \mathbb{N}$ and $n, w \in \mathbb{N}_+$. Let $v \in [n]$ and $i \in \{1, \ldots, w\}$. Define the i.i.d. random variables $Z_1, \ldots, Z_w$ with each identically distributed to $\text{Geo}(2/\tilde{\varepsilon})$ clamped to $[0, n]$ where $\tilde{\varepsilon} = 2 \cdot \ln(1 + 2^{-k})$. The following subroutine* `OrdSample`(k, n, v, i) *is identically distributed to the i-th order statistic $Z_{(i)}$ conditioned on $Z_{(i+1)} = v$. Also,* `OrdSample`(k, n, v, i) *has running time* $\text{poly}(n, k, i)$.

[1] $|\mathcal{X}| \geq 2n + 1$ ensures that $|\mathcal{X} \setminus A| \geq n + 1$. One can use `GeometricMechanism`$(D, \varepsilon)$ when $|\mathcal{X}| \leq 2n$.

[2] If instead we used continuous noise this last step is equivalent to releasing the n heaviest bins. However, in the discrete case, where ties can occur, from the set $A \cup \{x_1, \ldots, x_n\}$ we cannot determine all bins with a count tied for the n-th heaviest as there may be many other noisy counts tied with $\tilde{c}_{x_n}$. As a result, we only output the bins with a strictly heavier count than $\tilde{c}_{x_{n+1}}$.

As in `BoundedGeometricMechanism` (Algorithm 3.6), we have chosen $\tilde{\varepsilon}$ so that the cumulative distribution function of $\mathrm{Geo}(2/\tilde{\varepsilon})$ takes rational values with a common denominator of d. Therefore, the cumulative distribution function of $Z_{(i)}$ conditioned on $Z_{(i+1)} = v$ is also rational and we can sample from it with finite precision.

Algorithm 4.7. `OrdSample`(k, n, v, i) for $k \in \mathbb{N}$; $n, i \in \mathbb{N}_+$ and $v \in [n]$

1. Let $d = (2^{k+1} + 1)(2^k + 1)^{n-1}$.
2. Define the function
$$F(z) = \begin{cases} d - 2^{k(z+1)} \left(2^k + 1\right)^{n-1-z} & \text{if } 0 \le z < n \\ d & \text{if } z = n \end{cases}$$
3. Sample U uniformly at random from $\{1, \ldots, F(v)^i\}$.
4. Using binary search find the smallest $z \in [v]$ such that $F(z)^i \ge U$.
5. Return z.

Proof. By Lemma 3.5, $F(z)/d$ is the cumulative distribution of $\mathrm{Geo}(2/\tilde{\varepsilon})$ clamped to $[0, n]$. Therefore, by Lemma 2.13,
$$F_{Z_{(i)} | Z_{(i+1)} = v}(z) = \left(\frac{F(z)/d}{F(v)/d}\right)^i = \left(\frac{F(z)}{F(v)}\right)^i$$
for $z \in [v]$. Let U be drawn uniformly at random from $\{1, \ldots, F(v)^i\}$. Then, by construction,
$$\Pr[\texttt{OrdSample}(k, n, v, i) \le z] = \Pr[U \le F(z)^i] = \left(\frac{F(z)}{F(v)}\right)^i$$
implying $\texttt{OrdSample}(k, n, v, i) \sim Z_{(i)} \mid Z_{(i+1)} = v$.

The binary search takes $O(\log n)$ iterations. Each iteration has running time polynomial in the bit length of the numbers used. Therefore, this algorithm has running time $\mathrm{poly}(\log d, i) = \mathrm{poly}(n, k, i)$. □

Now from $\mathcal{M}_1$ we replace sampling from the joint distribution of the order statistics with iterative calls to `OrdSample` to get the following algorithm.

Algorithm 4.8. $\mathcal{M}_2(D, \varepsilon)$ for $D \in \mathcal{X}^n$, rational $\varepsilon \in (0, 1]$ and $|\mathcal{X}| \geq 2n + 1$

1. Let $k = \lceil \log(2/\varepsilon) \rceil$.
2. Let $A = \{x \in \mathcal{X} : c_x(D) > 0\}$ and $w = |\mathcal{X} \setminus A|$.
3. For each $x \in A$, let $\tilde{c}_x = \texttt{GeoSample}(k, n, c_x(D))$.
4. Pick a uniformly random sequence $(q_0, \ldots, q_n)$ of distinct elements from $\mathcal{X} \setminus A$.
5. Let $\tilde{c}_{q_0} = \texttt{OrdSample}(k, n, n, w)$.
6. For each $i \in \{1, \ldots, n\}$, let $\tilde{c}_{q_i} = \texttt{OrdSample}(k, n, \tilde{c}_{q_{i-1}}, w - i)$.
7. Sort the elements of $A \cup \{q_0, \ldots, q_n\}$ as $x_1, \ldots, x_{|A|+n+1}$ such that $\tilde{c}_{x_1} \geq \ldots \geq \tilde{c}_{x_{|A|+n+1}}$.
8. Release $h = \{(x, \tilde{c}_x) : x \in \{x_1, \ldots, x_n\} \text{ and } \tilde{c}_x > \tilde{c}_{x_{n+1}}\} \in \mathcal{H}_{n,n}(\mathcal{X})$.

Theorem 4.9. *Let rational $\varepsilon \in (0, 1]$ and $\tilde{\varepsilon} = 2 \cdot \ln(1 + 2^{-\lceil \log(2/\varepsilon) \rceil}) \in (4/9 \cdot \varepsilon, \varepsilon]$. $\mathcal{M}_2(D, \varepsilon)$ is identically distributed to $\texttt{KeepHeavy}(D, \tilde{\varepsilon})$. Therefore,*

i. $\mathcal{M}_2(D, \varepsilon)$ is $(\varepsilon, 0)$-differentially private.

ii. $\mathcal{M}_2(D, \varepsilon)$ has (a, β)-per-query accuracy on counts larger than t for

$$t = 2\left\lceil \frac{2}{\tilde{\varepsilon}} \ln \frac{2|\mathcal{X}|}{\beta} \right\rceil \quad \text{and} \quad a = \left\lceil \frac{2}{\tilde{\varepsilon}} \ln \frac{2}{\beta} \right\rceil$$

iii. $\mathcal{M}_2(D, \varepsilon)$ has (a, β)-simultaneous accuracy for

$$a = 2\left\lceil \frac{2}{\tilde{\varepsilon}} \ln \frac{|\mathcal{X}|}{\beta} \right\rceil$$

Proof. By Proposition 4.6, we have $(\tilde{c}_{q_0}, \ldots, \tilde{c}_{q_n})$ is identically distributed to the joint distribution $(Z_{(w)}, \ldots, Z_{(w-n)})$ used by $\mathcal{M}_1$. Therefore, this algorithm is identically distributed to $\mathcal{M}_1(D, \tilde{\varepsilon})$ and, then by Proposition 4.5, identically distributed to $\texttt{KeepHeavy}(D, \tilde{\varepsilon})$. Parts *i* to *iii* follow from Theorem 4.2. □

This algorithm only has an output of length $O(n \cdot (\log n + \log |\mathcal{X}|))$. However, its running time depends polynomially on $|\mathcal{X}|$ since sampling the w^{th} order statistic, $\tilde{c}_{q_0}$, using $\texttt{OrdSample}$ takes time polynomial in $w \geq |\mathcal{X}| - n$. Indeed, this is necessary since the distribution of the order statistic $Z_{(w)}$ has probabilities that are exponentially small in w. [3]

4.2 An Efficient Approximation

To remedy the inefficiency of $\mathcal{M}_2$ we consider an efficient algorithm that approximates the output distribution of $\mathcal{M}_2$.

[3] Notice $\Pr[Z_{(w)} = 0] = \Pr[Z = 0]^w$ where $Z \sim \text{Geo}(2/\tilde{\varepsilon})$. And $\Pr[Z = 0]^w = \left(\frac{e^{\tilde{\varepsilon}/2} - 1}{e^{\tilde{\varepsilon}/2} + 1}\right)^w$. So we need to toss $\Omega(w)$ coins to sample from this distribution.

Theorem 4.10. *There exists an algorithm* $\texttt{NonzeroGeometric} : \mathcal{X}^n \times (0,1] \times (0,1) \to \mathcal{H}_{n,n}(\mathcal{X})$ *such that for all input datasets* $D \in \mathcal{X}^n$*, rational* $\varepsilon \in (0,1]$ *and* $\delta \in (0,1)$ *such that* $1/\delta \in \mathbb{N}$

i. $\Delta\left(\mathcal{M}_2(D,\varepsilon), \texttt{NonzeroGeometric}(D,\varepsilon,\delta)\right) \leq \delta$.

ii. $\texttt{NonzeroGeometric}(D,\varepsilon,\delta)$ *is* $(\varepsilon, (e^\varepsilon+1)\cdot\delta)$*-differentially private.*

iii. Moreover, the running time of $\texttt{NonzeroGeometric}(D,\varepsilon,\delta)$ *is*

$$\text{poly}(N)$$

where N *is the bit length of this algorithm's input (*D*,* ε *and* δ*).*

Note that this algorithm only achieves (ε,δ)-differential privacy. By reducing δ, the algorithm better approximates $\mathcal{M}_2$, improving accuracy, at the cost of increasing running time (polynomial in $\log(1/\delta)$). This is in contrast to most (ε,δ)-differentially private algorithms such as the stability based algorithm of Section 5, where one needs $n \geq \Omega(\log(1/\delta)/\varepsilon)$ to get any meaningful accuracy.

Now, we convert $\texttt{NonzeroGeometric}$ to a pure differentially private algorithm by mixing it with a uniformly random output inspired by a similar technique in [KLN+11, CDK17].

Algorithm 4.11. $\mathcal{M}^*(D,\gamma)$ for $D \in \mathcal{X}^n$ and rational $\gamma \in (0,1)$

1. With probability $1-\gamma$ release $\mathcal{M}'(D)$.
2. Otherwise release a uniformly random element of $\mathcal{R}$.

Lemma 4.12. *Let* $\mathcal{M} : \mathcal{X}^n \to \mathcal{R}$ *be* $(\varepsilon,0)$*-differentially private with discrete range* $\mathcal{R}$*. Suppose algorithm* $\mathcal{M}' : \mathcal{X}^n \to \mathcal{R}$ *satisfies* $\Delta\left(\mathcal{M}(D), \mathcal{M}'(D)\right) \leq \delta$ *for all input datasets* $D \in \mathcal{X}^n$ *with parameter* $\delta \in [0,1)$*. Then the algorithm* $\mathcal{M}^*$ *has the following properties:*

i. $\mathcal{M}^*(D,\gamma)$ *is* $(\varepsilon,0)$*-differentially private whenever*

$$\delta \leq \frac{e^\varepsilon - 1}{e^\varepsilon + 1} \cdot \frac{\gamma}{1-\gamma} \cdot \frac{1}{|\mathcal{R}|}$$

ii. $\mathcal{M}^*(D,\gamma)$ *has running time upper bounded by the sum of the bit length to represent* γ*, the running time of* $\mathcal{M}'(D)$ *and the time required to sample a uniformly random element of* $\mathcal{R}$*.*

By taking γ and δ small enough and satisfying the constraint in part *i*, the algorithm $\mathcal{M}^*$ satisfies pure differential privacy and has nearly the same utility as $\mathcal{M}$ (due to having a statistical distance at most $\gamma+\delta$ from $\mathcal{M}$) while allowing for a possibly more efficient implementation since we only need to approximately sample from the output distribution of $\mathcal{M}$.

Proof of i. For all neighboring datasets $D, D' \in \mathcal{X}^n$ and all $r \in \mathcal{R}$

$$\begin{aligned}
\Pr[\mathcal{M}^*(D,\gamma) = r] &= \gamma \cdot \frac{1}{|\mathcal{R}|} + (1-\gamma)\cdot \Pr[\mathcal{M}'(D) = r] \\
&\leq \gamma \cdot \frac{1}{|\mathcal{R}|} + (1-\gamma)\cdot \left(\Pr[\mathcal{M}(D) = r] + \delta\right) \\
&\leq \gamma \cdot \frac{1}{|\mathcal{R}|} + (1-\gamma)\left(e^\varepsilon \cdot \Pr[\mathcal{M}(D') = r] + \delta\right) \\
&\leq \gamma \cdot \frac{1}{|\mathcal{R}|} + (1-\gamma)\left(e^\varepsilon \cdot \left(\Pr[\mathcal{M}'(D') = r] + \delta\right) + \delta\right)
\end{aligned}$$

Rearranging terms and using the upper bound on δ yields

$$\begin{aligned}\Pr[\mathcal{M}^*(D,\gamma)=r] &= e^{\varepsilon}(1-\gamma)\cdot\Pr[\mathcal{M}'(D')=r]+\gamma\cdot\frac{1}{|\mathcal{R}|}+(e^{\varepsilon}+1)(1-\gamma)\cdot\delta\\ &\le e^{\varepsilon}(1-\gamma)\cdot\Pr[\mathcal{M}'(D')=r]+\gamma\cdot\frac{1}{|\mathcal{R}|}+(e^{\varepsilon}-1)\left(\gamma\cdot\frac{1}{|\mathcal{R}|}\right)\\ &= e^{\varepsilon}\left((1-\gamma)\cdot\Pr[\mathcal{M}'(D')=r]+\gamma\cdot\frac{1}{|\mathcal{R}|}\right)\\ &= e^{\varepsilon}\cdot\Pr[\mathcal{M}^*(D',\gamma)=r]\end{aligned}$$

□

Proof of ii. This follows directly from the construction of $\mathcal{M}^*$. □

We can apply this lemma to `NonzeroGeometric` and under reasonable settings of parameters get accuracy bounds identical to $\mathcal{M}_2$ up to constant factors.

Algorithm 4.13. `PureNonzeroGeometric`(D,ε,β) for $D\in\mathcal{X}^n$, rational $\varepsilon\in(0,1]$ and $1/\beta\in\mathbb{N}$

1. With probability $1-\beta/3$ release `NonzeroGeometric`(D,ε,δ) with

$$\delta=\frac{2^{-\lceil\log(1/\varepsilon)\rceil}}{3}\cdot\frac{\beta}{3}\cdot\frac{1}{|\mathcal{H}_{n,n}(\mathcal{X})|}$$

2. Otherwise release a uniformly random element of $\mathcal{H}_{n,n}(\mathcal{X})$.

Theorem 4.14. *Let rational* $\varepsilon\in(0,1]$, $\tilde{\varepsilon}=2\cdot\ln\left(1+2^{-\lceil\log(2/\varepsilon)\rceil}\right)\in(4/9\cdot\varepsilon,\varepsilon]$ *and* $1/\beta\in\mathbb{N}$. `PureNonzeroGeometric`(D,ε,β) *has the following properties:*

i. `PureNonzeroGeometric`(D,ε,β) *is* $(\varepsilon,0)$*-differentially private.*

ii. `PureNonzeroGeometric`(D,ε,β) *has* (a,β)*-per-query accuracy on counts larger than* t *for*

$$t=2\left\lceil\frac{2}{\tilde{\varepsilon}}\ln\frac{6|\mathcal{X}|}{\beta}\right\rceil\quad\text{and}\quad a=\left\lceil\frac{2}{\tilde{\varepsilon}}\ln\frac{6}{\beta}\right\rceil$$

iii. `PureNonzeroGeometric`(D,ε,β) *has* (a,β)*-simultaneous accuracy for*

$$a=2\left\lceil\frac{2}{\tilde{\varepsilon}}\ln\frac{3|\mathcal{X}|}{\beta}\right\rceil$$

iv. `PureNonzeroGeometric`(D,ε,β) *has running time*

$$\mathrm{poly}(N)$$

where N *is the bit length of this algorithm's input (*$D\in\mathcal{X}^n$*,* ε *and* β*).*

Proof of i. Notice that $1/\delta\in\mathbb{N}$ (a constraint needed for `NonzeroGeometric`). Privacy follows from Lemma 4.12 by taking $\mathcal{M}=\mathcal{M}_2$, $\mathcal{M}'=$ `NonzeroGeometric`, $\gamma=\beta/3$ and $\mathcal{R}=\mathcal{H}_{n,n}(\mathcal{X})$ as

$$\delta=\frac{2^{-\lceil\log(1/\varepsilon)\rceil}}{3}\cdot\frac{\beta}{3}\cdot\frac{1}{|\mathcal{H}_{n,n}(\mathcal{X})|}\le\frac{e^{\varepsilon}-1}{e^{\varepsilon}+1}\cdot\frac{\beta/3}{1-\beta/3}\cdot\frac{1}{|\mathcal{H}_{n,n}(\mathcal{X})|}$$

□

Proof of ii-iii. For any $D \in \mathcal{X}^n$ and $x \in \mathcal{X}$ such that $c_x(D) > t$ define the set $G = \{h \in \mathcal{H}_{n,n}(\mathcal{X}) : |h_x - c_x(D)| \leq a\}$. By construction,

$$\Pr[\texttt{PureNonzeroGeometric}(D, \varepsilon, \beta) \in G] \geq \Pr[\texttt{NonzeroGeometric}(D, \varepsilon, \delta) \in G] - \beta/3$$

where δ is defined in Algorithm 4.13. Notice that $\delta \leq \beta/3$. So by Theorem 4.10,

$$\Pr[\texttt{NonzeroGeometric}(D, \varepsilon, \delta) \in G] - \beta/3 \geq \Pr[\mathcal{M}_2(D, \varepsilon) \in G] - 2 \cdot \beta/3$$

And by part *ii* of Theorem 4.9, we have

$$\Pr[\mathcal{M}_2(D, \varepsilon) \in G] - 2 \cdot \beta/3 \geq 1 - \beta$$

Similarly, we can bound the simultaneous accuracy by using part *iii* of Theorem 4.9. □

Proof of iv. By Lemma A1.1, it takes $\text{poly}(n, \log|\mathcal{X}|)$ time to sample a uniformly random element of $\mathcal{H}_{n,n}(\mathcal{X})$ and compute $|\mathcal{H}_{n,n}(\mathcal{X})|$. And by Theorem 4.10, $\texttt{NonzeroGeometric}(D, \varepsilon, \delta)$ with

$$\delta = \frac{2^{-\lceil \log(1/\varepsilon) \rceil}}{3} \cdot \frac{\beta}{3} \cdot \frac{1}{|\mathcal{H}_{n,n}(\mathcal{X})|}$$

has running time $\text{poly}(N)$. So by Lemma 4.12, `PureNonzeroGeometric` has the desired running time. □

4.3 Construction of `NonzeroGeometric`

We finish this section with the construction of `NonzeroGeometric`. Notice that $\mathcal{M}_2$ passes arguments to `OrdSample` that result in `OrdSample` exponentiating an integer, which represents the numerator of a fraction $a/b = F(z)/F(v)$, to a power $i \geq |\mathcal{X}| - n$. We want to ensure that numbers used by `OrdSample` do not exceed some maximum s.

The following algorithm will approximate $s \cdot (a/b)^i$ by using repeated squaring and truncating each intermediate result to keep the bit length manageable. The following lemma provides a bound on the error and on the running time.

Proposition 4.15. *There is an algorithm* $\texttt{ExpApprox}(a, b, i, s)$ *such that for all* $b, i, s \in \mathbb{N}_+$ *and* $a \in [b]$*:*

i. $\texttt{ExpApprox}(a, b, i, s)$ *is a nondecreasing function in* a *and* $\texttt{ExpApprox}(a, b, i, s) = s$ *when* $a = b$*.*

ii. $\texttt{ExpApprox}(a, b, i, s)$ *satisfies the accuracy bound of*

$$\left| \frac{\texttt{ExpApprox}(a, b, i, s)}{s} - \left(\frac{a}{b}\right)^i \right| \leq 2 \cdot \left(\frac{i}{s}\right)$$

iii. $\texttt{ExpApprox}(a, b, i, s)$ *has running time* $\text{poly}(\log b, \log i, \log s)$*.*

Proof. The algorithm is defined as follows.

Algorithm 4.16. $\texttt{ExpApprox}(a, b, i, s)$ for $b, i, s \in \mathbb{N}_+$ and $a \in [b]$

1. If $i = 1$ return $\left\lfloor \frac{as}{b} \right\rfloor$.
2. Otherwise let $r = \texttt{ExpApprox}(a, b, \lfloor i/2 \rfloor, s)$ and return $\begin{cases} \lfloor r^2/s \rfloor & \text{if } i \text{ is even} \\ \lfloor (a/b) \cdot (r^2/s) \rfloor & \text{if } i \text{ is odd} \end{cases}$

Proof of i. We proceed by induction. The case when $i = 1$ is trivial. Let $i > 1$ and assume that $\texttt{ExpApprox}(a, b, j, s)$ is a nondecreasing function in a for all $j < i$. Consider $a \le a' \le b$. If i is odd, then

$$\begin{aligned}\texttt{ExpApprox}(a, b, i, s) &= \left\lfloor \frac{a}{b} \cdot \frac{\texttt{ExpApprox}(a, b, \lfloor i/2 \rfloor, s)^2}{s} \right\rfloor \\ &\le \left\lfloor \frac{a'}{b} \cdot \frac{\texttt{ExpApprox}(a', b, \lfloor i/2 \rfloor, s)^2}{s} \right\rfloor = \texttt{ExpApprox}(a', b, i, s)\end{aligned}$$

Likewise, the result holds when i is even. Therefore, $\texttt{ExpApprox}(a, b, i, s)$ is a nondecreasing function in a. It is trivial that $\texttt{ExpApprox}\,(a, b, i, s) = s$ when $a = b$. □

Proof of ii. For ease of notation define $\texttt{EA}(i) = \texttt{ExpApprox}\,(a, b, i, s)$ as the other parameters do not change in our analysis. By construction

$$\texttt{EA}\,(i) = \frac{m_i \cdot \texttt{EA}\,(\lfloor i/2 \rfloor)^2}{s} + \omega_i$$

where $m_i = \begin{cases} 1 & \text{if } i \text{ is even} \\ a/b & \text{if } i \text{ is odd} \end{cases}$ and $|\omega_i| \le 1$. We can now bound the error as

$$\begin{aligned}\left| \frac{\texttt{EA}\,(i)}{s} - \left(\frac{a}{b}\right)^i \right| &\le \left| \frac{\texttt{EA}\,(i)}{s} - \frac{m_i \cdot \texttt{EA}\,(\lfloor i/2 \rfloor)^2}{s^2} \right| + \left| \frac{m_i \cdot \texttt{EA}\,(\lfloor i/2 \rfloor)^2}{s^2} - \left(\frac{a}{b}\right)^i \right| \\ &\le \frac{|\omega_i|}{s} + m_i \cdot \left| \frac{\texttt{EA}\,(\lfloor i/2 \rfloor)}{s} + \left(\frac{a}{b}\right)^{\lfloor i/2 \rfloor} \right| \cdot \left| \frac{\texttt{EA}\,(\lfloor i/2 \rfloor)}{s} - \left(\frac{a}{b}\right)^{\lfloor i/2 \rfloor} \right| \\ &\le \frac{1}{s} + 2 \cdot \left| \frac{\texttt{EA}\,(\lfloor i/2 \rfloor)}{s} - \left(\frac{a}{b}\right)^{\lfloor i/2 \rfloor} \right|\end{aligned}$$

When $i = 1$, we have

$$\left| \frac{\texttt{EA}(1)}{s} - \frac{a}{b} \right| = \left| \frac{\frac{as}{b} + \omega_1}{s} - \frac{a}{b} \right| \le \frac{1}{s}$$

So solving the recurrence gives

$$\left| \frac{\texttt{EA}\,(i)}{s} - \left(\frac{a}{b}\right)^i \right| \le 2 \cdot \left(\frac{i}{s}\right)$$ □

Proof of iii. The running time follows from the observation that a call to $\texttt{ExpApprox}\,(a, b, i, s)$ makes at most $O(\log i)$ recursive calls with the remaining operations polynomial in the bit lengths of the numbers used. □

Now we can modify `OrdSample` using `ExpApprox` to keep the bit lengths of its numbers from becoming too large, yielding an efficient algorithm whose output distribution is close to that of `OrdSample`.

Algorithm 4.17. `ApproxOrdSample`(k, n, v, i, s) for $k, n, i, s \in \mathbb{N}_+$ and $v \in [n]$

1. Let $d = (2^{k+1} + 1)(2^k + 1)^{n-1}$.
2. Define the function
$$F(z) = \begin{cases} d - 2^{k(z+1)} \left(2^k + 1\right)^{n-1-z} & \text{if } 0 \le z < n \\ d & \text{if } z = n \end{cases}$$
3. Sample U uniformly from $\{1, \ldots, s\}$.
4. Using binary search find the smallest $z \in [v]$ such that
$$\texttt{ExpApprox}\left(F(z), F(v), i, s\right) \ge U$$
5. Return z.

Proposition 4.18. *Let $k, n, i, s \in \mathbb{N}_+$ and $v \in [n]$. Then*

$$\Delta(\texttt{OrdSample}(k, n, v, i), \texttt{ApproxOrdSample}(k, n, v, i, s)) \le 2 \cdot (n + 1) \cdot \left(\frac{i}{s}\right)$$

In addition, `ApproxOrdSample`(k, n, v, i, s) *has a running time of* $\mathrm{poly}(k, n, \log i, \log s)$.

Proof. Let $Y \sim$ `OrdSample`(k, n, v, i) and $\widetilde{Y} \sim$ `ApproxOrdSample`(k, n, v, i, s). Let $z \in [v]$. Then

$$\Pr[Y = z] = \left(\frac{F(z)}{F(v)}\right)^i - \left(\frac{F(z-1)}{F(v)}\right)^i$$

where we define $F(-1) = 0$. By part i of Proposition 4.15, `ExpApprox`$(F(z), F(v), i, s)$ is an increasing function in z. Therefore,

$$\Pr[\widetilde{Y} = z] = \frac{\texttt{ExpApprox}(F(z), F(v), i, s)}{s} - \frac{\texttt{ExpApprox}(F(z-1), F(v), i, s)}{s}$$

Therefore, by the triangle inequality and part *ii* of Proposition 4.15

$$|\Pr[Y = z] - \Pr[\widetilde{Y} = z]| \le \sum_{z'=z-1}^{z} \left| \frac{\texttt{ExpApprox}(F(z'), F(v), i, s)}{s} - \left(\frac{F(z')}{F(v)}\right)^i \right| \le 4 \cdot \left(\frac{i}{s}\right)$$

Summing over $z \in [v]$ yields the desired bound on total variation distance (Definition 2.9).

The running time is dominated by the $O(\log n)$ calls to `ExpApprox`$(F(z), F(v), i, s)$. By part *iii* of Proposition 4.15, each call takes time

$$\mathrm{poly}(\log F(v), \log i, \log s) = \mathrm{poly}(k, n, \log i, \log s)$$

as $\log F(v) \le \log d = O(nk)$. So overall `ApproxOrdSample` has the desired running time. □

Because $\mathcal{M}_2$ samples from a joint distribution obtained by iterated calls to `OrdSample` we must also consider the accumulated distance between iterated calls to `OrdSample` and iterated calls to `ApproxOrdSample`.

Corollary 4.19. *Let $k, n, w, s \in \mathbb{N}_+$ with $w > n$. Consider the following random variables:*

- $Y_0 \sim \texttt{OrdSample}(k, n, n, w)$
- $Y_j \sim \texttt{OrdSample}(k, n, Y_{j-1}, w - j)$ *for each* $j \in \{1, \ldots, n\}$
- $\widetilde{Y}_0 \sim \texttt{ApproxOrdSample}(k, n, n, w, s)$
- $\widetilde{Y}_j \sim \texttt{ApproxOrdSample}(k, n, \widetilde{Y}_{j-1}, w - j, s)$ *for each* $j \in \{1, \ldots, n\}$

Then

$$\Delta\big((Y_0, \ldots, Y_n), (\widetilde{Y}_0, \ldots, \widetilde{Y}_n)\big) \le 2 \cdot (n+1)^2 \cdot \Big(\frac{w}{s}\Big)$$

Proof. By part *iii* of Lemma 2.10 and Proposition 4.18,

$$\Delta\big((Y_0, \ldots, Y_n), (\widetilde{Y}_0, \ldots, \widetilde{Y}_n)\big) \le \Delta\big(Y_0, \widetilde{Y}_0\big) + \sum_{j=1}^{n} \Delta\big(Y_j | Y_{j-1}, \widetilde{Y}_j | \widetilde{Y}_{j-1}\big) \le 2 \cdot (n+1)^2 \cdot \Big(\frac{w}{s}\Big)$$

where

$$\Delta\big(Y_j | Y_{j-1}, \widetilde{Y}_j | \widetilde{Y}_{j-1}\big) = \max_{x \in [n], S \subseteq [n]} \Big\{ \Big| \Pr[Y_j \in S | Y_{j-1} = x] - \Pr[\widetilde{Y}_j \in S | \widetilde{Y}_{j-1} = x] \} \Big| \Big\} \qquad \square$$

We are ready to state the mechanism `NonzeroGeometric` and show it satisfies Theorem 4.10. It is identical to $\mathcal{M}_2$ except we replace calls to `OrdSample` with calls to `ApproxOrdSample`.

Algorithm 4.20. $\texttt{NonzeroGeometric}(D, \varepsilon, \delta)$ for $D \in \mathcal{X}^n$, rational $\varepsilon \in (0, 1]$, $1/\delta \in \mathbb{N}$ and $|\mathcal{X}| \ge 2n + 1$

1. Let $k = \lceil \log(2/\varepsilon) \rceil$ and $s = 2 \cdot (n+1)^2 \cdot |\mathcal{X}|/\delta$.
2. Let $A = \{x \in \mathcal{X} : c_x(D) > 0\}$ and let $w = |\mathcal{X} \setminus A|$.
3. For each $x \in A$, let $\tilde{c}_x = \texttt{GeoSample}(k, n, c_x(D))$.
4. Pick a uniformly random sequence $(q_0, \ldots, q_n)$ of distinct elements from $\mathcal{X} \setminus A$.
5. Let $\tilde{c}_{q_0} = \texttt{ApproxOrdSample}(k, n, n, w, s)$.
6. For each $i \in \{1, \ldots, n\}$, let $\tilde{c}_{q_i} = \texttt{ApproxOrdSample}(k, n, \tilde{c}_{q_{i-1}}, w - i, s)$.
7. Sort the elements of $A \cup \{q_0, \ldots, q_n\}$ as $x_1, \ldots, x_{|A|+n+1}$ such that $\tilde{c}_{x_1} \ge \ldots \ge \tilde{c}_{x_{|A|+n+1}}$.
8. Release $h = \{(x, \tilde{c}_x) : x \in \{x_1, \ldots, x_n\} \text{ and } \tilde{c}_x > \tilde{c}_{x_{n+1}}\} \in \mathcal{H}_{n,n}(\mathcal{X})$.

Theorem 4.10 (restated)**.** *The algorithm* $\texttt{NonzeroGeometric}(D, \varepsilon, \delta)$ *satisfies for all input datasets* $D \in \mathcal{X}^n$*, rational* $\varepsilon \in (0, 1]$ *and* $\delta \in (0, 1)$ *such that* $1/\delta \in \mathbb{N}$

i. $\Delta\left(\mathcal{M}_2(D, \varepsilon), \texttt{NonzeroGeometric}(D, \varepsilon, \delta)\right) \le \delta$.

ii. $\texttt{NonzeroGeometric}(D, \varepsilon, \delta)$ *is* $(\varepsilon, (e^\varepsilon + 1) \cdot \delta)$*-differentially private.*

iii. Moreover, the running time of $\texttt{NonzeroGeometric}(D, \varepsilon, \delta)$ *is*

$$\text{poly}(N)$$

where N is the bit length of this algorithm's input (D, ε and δ).

Proof of i. Let $\mathcal{M}_2^* : \mathcal{X}^n \times (0,1] \to \mathcal{H}_{n,2n+1}(\mathcal{X})$ be the algorithm $\mathcal{M}_2$ except, instead of releasing the heaviest bins, $\mathcal{M}_2^*$ releases the bins for all elements of $A \cup \{x_0, \ldots, x_n\}$ (i.e. $\mathcal{M}_2^*$ releases $(y, \tilde{c}_y)$ for all $y \in A$ and $(x_i, \tilde{c}_{x_i})$ for all $i \in [n]$). Similarly, we define $\texttt{NonzeroGeometric}^*$ with respect to $\texttt{NonzeroGeometric}$.

Notice that $\mathcal{M}_2^*$ and $\texttt{NonzeroGeometric}^*$ have the same distribution overs the bins with nonzero true count. Only on the bins with counts sampled using $\texttt{OrdSample}$ and $\texttt{ApproxOrdSample}$ respectively do their output distributions differ. As a result, we can apply Corollary 4.19 to the output distributions of $\mathcal{M}_2^*$ and $\texttt{NonzeroGeometric}^*$. So for all $D \in \mathcal{X}^n$

$$\Delta\left(\mathcal{M}_2^*(D, \varepsilon), \texttt{NonzeroGeometric}^*(D, \varepsilon, \delta)\right) \le 2 \cdot (n+1)^2 \cdot \left(\frac{w}{s}\right) \le \delta$$

Now we consider the effect of keeping the heaviest counts. Define $T : \mathcal{H}_{n,2n+1} \to \mathcal{H}_{n,n}(\mathcal{X})$ to be the function that sets counts not strictly larger than the $(n+1)$-heaviest count of its input to 0. Notice that $T \circ \mathcal{M}_2^* \sim \mathcal{M}_2$ and $T \circ \texttt{NonzeroGeometric}^* \sim \texttt{NonzeroGeometric}$. So for all $D \in \mathcal{X}^n$, by part *ii* of Lemma 2.10,

$$\begin{aligned}
\Delta(\mathcal{M}_2(D, \varepsilon), \texttt{NonzeroGeometric}(D, \varepsilon, \delta)) &= \Delta(T(\mathcal{M}_2^*(D, \varepsilon)), T(\texttt{NonzeroGeometric}^*(D, \varepsilon, \delta))) \\
&\le \Delta(\mathcal{M}_2^*(D, \varepsilon), \texttt{NonzeroGeometric}^*(D, \varepsilon, \delta)) \\
&\le \delta
\end{aligned}$$

□

Proof of ii. Let D and D' be neighboring datasets. Let $S \subseteq \mathcal{H}_{n,n}(\mathcal{X})$. By the previous part and part i of Theorem 4.9,

$$\begin{aligned}
\Pr[\texttt{NonzeroGeometric}(D, \varepsilon, \delta) \in S] &\le \Pr[\mathcal{M}_2(D, \varepsilon) \in S] + \delta \\
&\le e^{\varepsilon} \cdot \Pr[\mathcal{M}_2(D', \varepsilon) \in S] + \delta \\
&\le e^{\varepsilon} \cdot \left(\Pr[\texttt{NonzeroGeometric}(D', \varepsilon, \delta) \in S] + \delta\right) + \delta
\end{aligned}$$

Therefore, $\texttt{NonzeroGeometric}(D, \varepsilon, \delta)$ is $(\varepsilon, (e^{\varepsilon} + 1) \cdot \delta)$-differentially private. □

Proof of iii. We consider the running time at each step. Construction of the true histogram takes $O(n \log |\mathcal{X}|)$ time. And by Proposition 3.3, the at most n calls to $\texttt{GeoSample}$ take $\text{poly}(\log(1/\varepsilon), n)$ time. Sampling n random bin labels from $\mathcal{X}$ takes $O(n \log |\mathcal{X}|)$ time.

Now, $\texttt{NonzeroGeometric}$ makes $n+1$ calls to $\texttt{ApproxOrdSample}$ with no argument exceeding each term of $(k, n, n, |\mathcal{X}|, s)$ respectively. So by Proposition 4.18, these calls take time

$$\text{poly}(\log(1/\varepsilon), n, \log |\mathcal{X}|, \log s) \le \text{poly}(\log(1/\varepsilon), n, \log |\mathcal{X}|, \log(1/\delta))$$

Sorting the at most $2n+1$ elements of $A \cup \{q_0, \ldots, q_n\}$ and then releasing the heaviest counts takes $O(n \log n + n \log |\mathcal{X}|)$ time.

Therefore, overall $\texttt{NonzeroGeometric}$ has the desired running time. □

We have constructed algorithms for releasing a differentially private histogram, both pure and approximate, with running time polynomial in $\log |\mathcal{X}|$ and simultaneous accuracy matching that of $\texttt{GeometricMechanism}$ up to constant factors.

5 Removing the Dependence on Universe Size

When we would like to have accuracy independent of $|\mathcal{X}|$, we can use an approximate differentially private algorithm based on stability techniques [BNS16] (Proposition 2.20). We present a reformulation of their algorithm using two-sided geometric noise instead of Laplace noise.

Algorithm 5.1. `StabilityHistogram`(D, ε, b) for $D \in \mathcal{X}^n$, $\varepsilon > 0$ and $b \in [n]$

1. Let $A = \{x \in \mathcal{X} : c_x(D) > 0\}$.
2. For each $x \in A$, set $\tilde{c}_x$ to $c_x(D) + \mathrm{Geo}(2/\varepsilon)$ clamped to $[0, n]$.
3. Release $h = \{(x, \tilde{c}_x) : x \in A \text{ and } \tilde{c}_x > b\} \in \mathcal{H}_{n,n}(\mathcal{X})$.

Note that we only release counts for $x \in \mathcal{X}$ whose true count is nonzero, namely elements in the set A. Thus, the output length is $O(n \cdot (\log n + \log |\mathcal{X}|))$. However, releasing the set A is not $(\varepsilon, 0)$-differentially private because this would distinguish between neighboring datasets: one with a count of 0 and the other with a count of 1 for some element $x \in \mathcal{X}$. Thus, we only release noisy counts $\tilde{c}_x$ that exceed a threshold b. If b is large enough, then a count of 1 will only be kept with small probability, yielding approximate differential privacy.

Theorem 5.2. `StabilityHistogram`(D, ε, b) *has the following properties:*

i. `StabilityHistogram`(D, ε, b) *is* (ε, δ)*-differentially private provided that*

$$b \geq 1 + \frac{2}{\varepsilon} \ln \frac{1}{\delta}$$

ii. `StabilityHistogram`(D, ε, b) *has* (a, β)*-per-query accuracy on counts larger than* t *for*

$$t = b + \left\lceil \frac{2}{\varepsilon} \ln \frac{1}{\beta} \right\rceil \quad \textit{and} \quad a = \left\lceil \frac{2}{\varepsilon} \ln \frac{1}{\beta} \right\rceil$$

iii. `StabilityHistogram`(D, ε, b) *has* (a, β)*-simultaneous accuracy for*

$$a = b + \left\lceil \frac{2}{\varepsilon} \ln \frac{n}{\beta} \right\rceil$$

Proof of i. Let D and D' be neighboring datasets. Let $x \in \mathcal{X}$ such that $c_x(D) \neq c_x(D')$ and $S \subseteq [n]$. There are 3 cases to consider:

- $c_x(D) \geq 1$ and $c_x(D') \geq 1$. Then $\tilde{c}_x \sim c_x(D) + \mathrm{Geo}(2/\varepsilon)$ and similarly on the neighboring database we have a noisy count $\tilde{c}'_x \sim c_x(D') + \mathrm{Geo}(2/\varepsilon)$. So by the differential privacy of `GeometricMechanism` (part i of Theorem 3.2) we have

$$\Pr[\mathcal{M}(D)_x \in S] \leq e^{\varepsilon/2} \cdot \Pr[\mathcal{M}(D')_x \in S]$$

- $c_x(D) = 1$ and $c_x(D') = 0$. Notice $\Pr[\mathcal{M}(D')_x \neq 0] = 0$. $\tilde{c}_x$ is distributed as $1 + \text{Geo}(2/\varepsilon)$ clamped to $[0, n]$. Thus,

$$\begin{aligned}\Pr[\mathcal{M}(D)_x \neq 0] = \Pr[\tilde{c}_x > b] &\leq \Pr\left[\tilde{c}_x > 1 + \frac{2}{\varepsilon}\ln\frac{1}{\delta}\right] \\ &\leq \Pr\left[Z > \frac{2}{\varepsilon}\ln\frac{1}{\delta}\right] = \frac{\delta}{e^{\varepsilon/2}+1} \leq \frac{\delta}{2}\end{aligned}$$

  where $Z \sim \text{Geo}(2/\varepsilon)$. Therefore,

$$\Pr[\mathcal{M}(D)_x \in S] \leq \Pr[\mathcal{M}(D')_x \in S] + \delta/2$$

- $c_x(D) = 0$ and $c_x(D') = 1$. This case follows similarly to the previous case by considering $\Pr[\mathcal{M}(D)_x = 0]$.

Then overall

$$\Pr[\mathcal{M}(D)_x \in S] \leq e^{\varepsilon/2} \cdot \Pr[\mathcal{M}(D')_x \in S] + \delta/2$$

Because there are at most two bins on which D and D' have differing counts and each count $\tilde{c}_x$ is computed independently, by Lemma 2.4, this algorithm is (ε, δ)-differentially private. □

Proof of ii. Let $x \in \mathcal{X}$ such that $c_x(D) > t$. Then $x \in A$. So by part *ii* of Theorem 3.2, we have $\Pr\left[|\tilde{c}_x - c_x(D)| \leq a\right] \geq 1 - \beta$. $|\tilde{c}_x - c_x(D)| \leq a$ implies $\tilde{c}_x > t - a = b$. Thus, $(x, \tilde{c}_x)$ is included in the output of $\texttt{StabilityHistogram}(D, \varepsilon, b)$ giving the desired accuracy. □

Proof of iii. Notice that the counts of elements not in A are trivially accurate. Therefore, we only need to consider the counts of elements in A. By part *iii* of Theorem 3.2,

$$\Pr\left[\forall x \in A \;\; |\tilde{c}_x - c_x(D)| \leq \left\lceil \frac{2}{\varepsilon}\ln\frac{|A|}{\beta}\right\rceil\right] \geq 1 - \beta$$

The final step can increase the error additively by at most b. Also, $|A| \leq n$. Therefore,

$$\Pr\left[\forall x \in \mathcal{X} \;\; |h_x - c_x(D)| \leq b + \left\lceil \frac{2}{\varepsilon}\ln\frac{n}{\beta}\right\rceil\right] \geq 1 - \beta$$ □

By using `GeoSample` (see Proposition 3.3), we can construct a computationally efficient algorithm for releasing a histogram that is (ε, δ)-differentially private with the accuracies matching Algorithm 5.1 up to constant factors.

Algorithm 5.3. $\texttt{BoundedStabilityHistogram}(D, \varepsilon, b)$ for $D \in \mathcal{X}^n$, rational $\varepsilon \in (0, 1]$ and $b \in [n]$

1. Let $k = \lceil \log(2/\varepsilon) \rceil$.
2. Let $A = \{x \in \mathcal{X} : c_x(D) > 0\}$.
3. For each $x \in A$, let $\tilde{c}_x = \texttt{GeoSample}(k, n, c_x(D))$.
4. Release $h = \{(x, \tilde{c}_x) : x \in A \text{ and } \tilde{c}_x > b\} \in \mathcal{H}_{n,n}(\mathcal{X})$.

Theorem 5.4. *Let rational* $\varepsilon \in (0,1]$, $\tilde{\varepsilon} = 2 \cdot \ln\left(1 + 2^{-\lceil \log(2/\varepsilon) \rceil}\right) \in (4/9 \cdot \varepsilon, \varepsilon]$ *and* $b \in [n]$. *Then* `BoundedStabilityHistogram`(D, ε, b) *satisfies the following properties:*

i. `BoundedStabilityHistogram`(D, ε, b) *is* (ε, δ)*-differentially private provided that*

$$b \geq 1 + \frac{2}{\tilde{\varepsilon}} \ln(1/\delta)$$

ii. `BoundedStabilityHistogram`(D, ε, b) *has* (a, β)*-per-query accuracy on counts larger than* t *for*

$$t = b + \left\lceil \frac{2}{\tilde{\varepsilon}} \ln \frac{1}{\beta} \right\rceil \quad \text{and} \quad a = \left\lceil \frac{2}{\tilde{\varepsilon}} \ln \frac{1}{\beta} \right\rceil$$

iii. `BoundedStabilityHistogram`(D, ε, b) *has* (a, β)*-simultaneous accuracy for*

$$a = b + \left\lceil \frac{2}{\tilde{\varepsilon}} \ln \frac{n}{\beta} \right\rceil$$

iv. `BoundedStabilityHistogram`(D, ε, b) *has running time*

$$\text{poly}(N)$$

where N *is the bit length of this algorithm's input* ($D \in \mathcal{X}^n$, ε *and* b).

Proof of i-iii. By Proposition 3.3, `GeoSample`$(k, n, c_x(D))$ generates a two-sided geometric random variable with scale parameter $2/\tilde{\varepsilon}$ centered at $c_x(D)$ clamped to $[0, n]$. Therefore, this algorithm is identically distributed to `StabilityHistogram`$(D, \tilde{\varepsilon}, b)$. Parts *i-iii* follow from Theorem 5.2. □

Proof of iv. Construction of the true histogram takes $O(n \log |\mathcal{X}|)$ time. And by Proposition 3.3, the at most n calls to `GeoSample` take $\text{poly}(\log(1/\varepsilon), n)$ time. Because the counts do not exceed n, the final step takes $\text{poly}(n, \log |\mathcal{X}|)$ time. □

Therefore, we have constructed an efficient algorithm for releasing a sparse histogram with approximate differential privacy.

6 Lower Bounds

In this section, we prove a lower bound on the per-query accuracy of histogram algorithms whose outputs are restricted to $\mathcal{H}_{\infty,n'}(\mathcal{X})$ (i.e. sparse histograms) using a packing argument [HT10, BBKN14]. First, for completeness we state and reprove existing lower bounds for per-query accuracy and simultaneous accuracy as well as generalize them to the case of $\delta > 0$.

Theorem 6.1 (following [HT10, BBKN14])**.** *Let* $\mathcal{M} : \mathcal{X}^n \to \mathcal{H}_{\infty,|\mathcal{X}|}(\mathcal{X})$ *be* (ε, δ)*-differentially private and* $\beta \in (0, 1/2]$.

i. If $\mathcal{M}$ *has* (a, β)*-per-query accuracy and* $\delta/\varepsilon \leq \beta$, *then*

$$a \geq \frac{1}{2} \cdot \min\left\{ \frac{1}{\varepsilon} \ln\left(\frac{1}{4\beta}\right) - 1,\ n \right\}$$

ii. If $\mathcal{M}$ has (a, β)-simultaneous accuracy, then

$$a \geq \frac{1}{2} \cdot \min\left\{\frac{1}{\varepsilon}\log\left(\frac{|\mathcal{X}|-1}{4\beta}\right) - 1,\ \frac{1}{\varepsilon}\log\left(\frac{\varepsilon}{4\delta}\right) - 1,\ n\right\}$$

Proof of i. Assume $a < n/2$. Let $x, x_0 \in \mathcal{X}$ such that $x \neq x_0$. Define the dataset $D_0 \in \mathcal{X}^n$ such that all rows are x_0. And define the dataset D such that the first $m = \lfloor 2a \rfloor + 1$ rows are x and the remaining $n - m$ rows are x_0. Notice that $\Pr[|\mathcal{M}(D)_x - c_x(D)| > a] \leq \beta$ by the (a, β)-per-query accuracy of $\mathcal{M}$. By Lemma 2.3 and the fact that $c_x(D) > 2a$ while $c_x(D') = 0$,

$$\begin{aligned}\Pr[\mathcal{M}(D)_x - c_x(D)| > a] &\geq e^{-m\varepsilon} \cdot \Pr[\mathcal{M}(D')_x - c_x(D)| > a] - \delta/\varepsilon \\ &\geq e^{-m\varepsilon} \cdot \Pr[\mathcal{M}(D')_x - c_x(D')| \leq a] - \delta/\varepsilon \\ &\geq e^{-m\varepsilon} \cdot (1-\beta) - \delta/\varepsilon\end{aligned}$$

Therefore,

$$e^{-(2a+1)\cdot\varepsilon} \leq \frac{1}{1-\beta} \cdot \left(\beta + \frac{\delta}{\epsilon}\right) \leq 4\beta$$ □

Proof of ii. Assume $a < n/2$. Let $x_0 \in \mathcal{X}$. For each $x \in \mathcal{X}$ define the dataset $D^{(x)} \in \mathcal{X}^n$ such that the first $m = \lfloor 2a \rfloor + 1$ rows are x and the remaining $n - m$ rows are x_0. For all $x \in \mathcal{X}$, let

$$G_x = \{h \in \mathcal{H}_{\infty,|\mathcal{X}|}(\mathcal{X}) : \forall x' \in \mathcal{X}\ \ |h_{x'} - c_{x'}(D^{(x)})| \leq a\}$$

By Lemma 2.3, for all $x \in \mathcal{X}$

$$\begin{aligned}\Pr[\mathcal{M}(D^{(x_0)}) \in G_x] &\geq e^{-m\varepsilon} \cdot \Pr[\mathcal{M}(D^{(x)}) \in G_x] - \delta/\varepsilon \\ &\geq e^{-m\varepsilon} \cdot (1-\beta) - \delta/\varepsilon\end{aligned}$$

Notice that $\Pr[\mathcal{M}(D^{(x_0)}) \notin G_{x_0}] \leq \beta$ and $\{G_x\}_{x\in\mathcal{X}}$ is a collection of disjoint sets. Then

$$\begin{aligned}\Pr[\mathcal{M}(D^{(x_0)}) \notin G_{x_0}] &\geq \sum_{x\in\mathcal{X}:x\neq x_0} \Pr[\mathcal{M}(D^{(x_0)}) \in G_x] \\ &\geq (|\mathcal{X}|-1) \cdot \left(e^{-m\varepsilon} \cdot (1-\beta) - \delta/\varepsilon\right)\end{aligned}$$

Therefore,

$$e^{-(2a+1)\cdot\varepsilon} \leq \frac{1}{1-\beta} \cdot \left(\frac{\beta}{|\mathcal{X}|-1} + \frac{\delta}{\varepsilon}\right)$$

which implies the desired lower bound. □

We now state our lower bound over sparse histograms.

Theorem 6.2. *Let $\mathcal{M} : \mathcal{X}^n \to \mathcal{H}_{\infty,n'}(\mathcal{X})$ be (ε, δ)-differentially private with (a, β)-per-query accuracy with $\beta \in (0, 1/2]$ and $\delta/\varepsilon \leq \beta$. Then*

$$a \geq \frac{1}{2} \cdot \min\left\{\frac{1}{2\varepsilon}\ln\left(\frac{|\mathcal{X}|}{16\beta n'}\right) - 1,\ \frac{1}{2\varepsilon}\ln\left(\frac{\varepsilon}{16\beta\delta}\right) - 1,\ n\right\}$$

The histogram algorithms of Sections 4 and 5 achieve $(O(\log(1/\beta)/\varepsilon), \beta)$-per-query accuracy on large enough counts. However, on smaller counts we can only guarantee (a, β)-per-query accuracy with $a = O(\log(|\mathcal{X}|/\beta)/\varepsilon)$ and $a = O(\log(1/(\beta\delta))/\varepsilon)$ (by taking threshold $b = O(\log(1/\delta)/\varepsilon)$) for algorithms from Sections 4 and 5 respectively. Theorem 6.2 shows these bounds are the best possible when $|\mathcal{X}| \geq \text{poly}(n')$ and $\delta \leq \text{poly}(1/\varepsilon)$.

Proof. Assume $a < n/2$. Let $x_0 \in \mathcal{X}$. For each $x \in \mathcal{X}$ define the dataset $D^{(x)} \in \mathcal{X}^n$ such that the first $m = \lceil 2a \rceil$ rows are x and the remaining $n - m$ rows are x_0. By definition of (a, β)-per-query accuracy and the fact that $c_x(D^{(x)}) \geq 2a$, we have

$$\Pr\left[\mathcal{M}(D^{(x)})_x \geq a\right] \geq \Pr\left[\left|\mathcal{M}(D^{(x)})_x - c_x(D^{(x)})\right| \leq a\right] \geq 1 - \beta$$

Then, by Lemma 2.3 and that $D^{(x)}$ is at distance at most m from $D^{(x_0)}$, we have

$$\Pr\left[\mathcal{M}(D^{(x_0)})_x \geq a\right] \geq (1 - \beta)e^{-m\varepsilon} - \delta/\varepsilon$$

Thus, by linearity of expectations

$$\mathrm{E}\left[\left|\left\{x \in \mathcal{X} : \mathcal{M}(D^{(x_0)})_x \geq a\right\}\right|\right] \geq |\mathcal{X}| \cdot \left((1 - \beta)e^{-m\varepsilon} - \delta/\varepsilon\right)$$

On the other hand, as $\mathcal{M}(D^{(x_0)}) \in \mathcal{H}_{\infty,n'}(\mathcal{X})$ we have

$$\mathrm{E}\left[\left|\left\{x \in \mathcal{X} : \mathcal{M}(D^{(x_0)})_x \geq a\right\}\right|\right] \leq n'$$

Therefore,

$$e^{-\lceil 2a \rceil \cdot \varepsilon} \leq \frac{1}{1 - \beta} \cdot \left(\frac{n'}{|\mathcal{X}|} + \frac{\delta}{\varepsilon}\right)$$

which along with $\lceil 2a \rceil \leq 2a + 1$ implies the lower bound of

$$a \geq \frac{1}{2} \cdot \min\left\{\frac{1}{\varepsilon}\ln\left(\frac{|\mathcal{X}|}{4n'}\right) - 1,\ \frac{1}{\varepsilon}\ln\left(\frac{\varepsilon}{4\delta}\right) - 1,\ n\right\}$$

Therefore, along with part i of Theorem 6.1, we have

$$\begin{aligned}
a &\geq \frac{1}{2} \cdot \min\left\{\max\left\{\min\left\{\frac{1}{\varepsilon}\ln\left(\frac{|\mathcal{X}|}{4n'}\right) - 1,\ \frac{1}{\varepsilon}\ln\left(\frac{\varepsilon}{4\delta}\right) - 1\right\},\ \frac{1}{\varepsilon}\ln\left(\frac{1}{4\beta}\right) - 1\right\},\ n\right\} \\
&\geq \frac{1}{2} \cdot \min\left\{\frac{1}{2} \cdot \left(\min\left\{\frac{1}{\varepsilon}\ln\left(\frac{|\mathcal{X}|}{4n'}\right) - 1,\ \frac{1}{\varepsilon}\ln\left(\frac{\varepsilon}{4\delta}\right) - 1\right\} + \frac{1}{\varepsilon}\ln\left(\frac{1}{4\beta}\right) - 1\right),\ n\right\} \\
&\geq \frac{1}{2} \cdot \min\left\{\frac{1}{2\varepsilon}\ln\left(\frac{|\mathcal{X}|}{16\beta n'}\right) - 1,\ \frac{1}{2\varepsilon}\ln\left(\frac{\varepsilon}{16\beta\delta}\right) - 1,\ n\right\}
\end{aligned}$$

□

7 Better Per-Query Accuracy via Compact, Non-Sparse Representations

In this section, we present a histogram algorithm whose running time is poly-logarithmic in $|\mathcal{X}|$, but, unlike Algorithm 4.13, is able to achieve (a, β)-per query accuracy with $a = O(\log(1/\beta)/\varepsilon)$. Our histogram algorithm will output a histogram from a properly chosen family. This family necessarily contains histograms that have many nonzero counts to avoid the lower bound of Theorem 6.2.

Lemma 7.1. *Let $k \in \mathbb{N}$, $n \in \mathbb{N}_+$ and $Z \sim \texttt{GeoSample}(k, n, 0)$. There exists a multiset of histograms $\mathcal{G}_{k,n}(\mathcal{X})$ satisfying:*

i. Let h be drawn uniformly at random from $\mathcal{G}_{k,n}(\mathcal{X})$. For all $x \in \mathcal{X}$ and $c \in [n]$

$$e^{-(2/3)\cdot 2^{-k+1}} \cdot \Pr[Z = c] \leq \Pr[h_x = c] \leq e^{(1/3)\cdot 2^{-k+1}} \cdot \Pr[Z = c]$$

ii. Let h be drawn uniformly at random from $\mathcal{G}_{k,n}(\mathcal{X})$. For all $x \in \mathcal{X}$ and $a \in [n]$

$$\Pr[h_x \leq a] \geq \Pr[Z \leq a]$$

iii. Let h be drawn uniformly at random from $\mathcal{G}_{k,n}(\mathcal{X})$. For all $B \subseteq \mathcal{X}$ such that $|B| \leq n+1$ and $c \in [n]^B$

$$\Pr[\forall x \in B \quad h_x = c_x] = \prod_{x \in B} \Pr[h_x = c_x]$$

iv. For all $h \in \mathcal{G}_{k,n}$, the histogram h can be represented by a string of length $\text{poly}(k, n, \log|\mathcal{X}|)$ and given this representation for all $x \in \mathcal{X}$ the count h_x can be evaluated in time

$$\text{poly}\left(k, n, \log|\mathcal{X}|\right)$$

v. For all $A \subseteq \mathcal{X}$ such that $|A| \leq n$ and $c \in [n]^A$ sampling a histogram h uniformly at random from $\{h' \in \mathcal{G}_{k,n}(\mathcal{X}) : \forall x \in A \quad h'_x = c_x\}$ can be done in time

$$\text{poly}\left(k, n, \log|\mathcal{X}|\right)$$

Parts *i* and *ii* state when a histogram is sampled uniformly at random from $\mathcal{G}_{n,k}(\mathcal{X})$ the marginal distribution on each count is closely distributed to a two-sided geometric distribution centered at 0 clamped to $[0, n]$.

Proof. (Construction) Let $d = (2^{k+1}+1)(2^k+1)^{n-1}$. We would like to construct a $(n+1)$-wise independent hash family consisting of functions $p : \mathcal{X} \to \{1, \ldots, d\}$, i.e. for all $x \in \mathcal{X}$, $p(x)$ is uniformly distributed over $\{1, \ldots, d\}$ and for all distinct $x_1, \ldots, x_{n+1} \in \mathcal{X}$, the random variables $p(x_1), \ldots, p(x_{n+1})$ are independent.

Given any function p in this family we can construct a histogram by using $p(x)$ as the randomness for evaluating the noisy count of x via inverse transform sampling in a similar manner to Algorithm 3.4 as the marginal distribution of $p(x)$ is uniformly distributed over $\{1, \ldots, d\}$.

The set of all degree at most n polynomial over a finite field $\mathbb{F}_q$ is a $(n+1)$-wise independent hash family. Ideally, we would have $|\mathcal{X}| \leq d$ and take $q = d$. In this case, we can map $\mathcal{X}$ to a subset of $\mathbb{F}_q$ and use a bijection between $\{1, \ldots d\}$ and $\mathbb{F}_q$ to get the desired family of functions.

However, $|\mathcal{X}|$ may be larger than d and d is not a prime power. Therefore, we must pick a large enough field $\mathbb{F}_q$ so that $|\mathcal{X}| \leq q$ and that when mapping $\mathbb{F}_q$ to $\{1, \ldots, d\}$, the resulting marginal distributions are approximately uniform. So let

$$m = \max\left\{\lceil \log|\mathcal{X}| \rceil, \left\lceil \log(3 \cdot 2^{k-1} \cdot d) \right\rceil\right\}$$

and define $\mathcal{G}'_{k,n}(\mathcal{X})$ as the family of polynomials over the finite field $\mathbb{F}_{2^m}$ with degree at most n. We construct a histogram $h \in \mathcal{G}_{k,n}(\mathcal{X})$ for each polynomial $p_h \in \mathcal{G}'_{k,n}(\mathcal{X})$ by taking

$$h_x = \min\left\{z \in [n] : F(z) \geq (T(p_h(T^{-1}(x))) \bmod d) + 1\right\}$$

for all $x \in \mathcal{X}$ where $T : \mathbb{F}_{2^m} \to [2^m - 1]$ is a bijection such that T and T^{-1} have running time $\text{poly}(m)$ and

$$F(z) = \begin{cases} 0 & \text{if } z < 0 \\ d - 2^{k(z+1)} \left(2^k + 1\right)^{n-1-z} & \text{if } 0 \leq z < n \\ d & \text{if } z = n \end{cases}$$

as defined in Algorithm 3.4 for $c = 0$. Notice that if h is drawn uniformly at random from $\mathcal{G}_{k,n}(\mathcal{X})$, then p_h is drawn uniformly at random from $\mathcal{G}'_{k,n}(\mathcal{X})$.

Proof of i. Let $Z \sim \texttt{GeoSample}(k, n, 0)$. For all $x \in \mathcal{X}$ and $v \in [n]$, we have

$$\begin{aligned} \Pr[h_x = v] &= \Pr[\min\left\{z \in [n] : F(z) \geq (T(p_h(T^{-1}(x))) \bmod d) + 1\right\} = v] \\ &= \Pr[F(v-1) < (T(p_h(T^{-1}(x))) \bmod d) + 1 \leq F(v)] \\ &= \frac{1}{2^m} \cdot |\{y \in \mathbb{F}_{2^m} : F(v-1) < (T(y) \bmod d) + 1 \leq F(v)\}| \\ &\leq \frac{1}{2^m} \cdot \left(\frac{2^m}{d} + 1\right)(F(v) - F(v-1)) \\ &= \left(1 + \frac{d}{2^m}\right) \cdot \Pr[Z = v] && \text{(by Lemma 3.5)} \\ &\leq e^{d/2^m} \cdot \Pr[Z = v] \end{aligned}$$

Similarly, because $d/2^m \leq 2/3$

$$\begin{aligned} \Pr[h_x = v] &\geq \frac{1}{2^m} \cdot \left(\frac{2^m}{d} - 1\right)(F(v) - F(v-1)) \\ &\geq \left(1 - \frac{d}{2^m}\right) \cdot \Pr[Z = v] \\ &\geq e^{-2 \cdot d/2^m} \cdot \Pr[Z = v] \end{aligned}$$

Now, part i follows as $d/2^m \leq (1/3) \cdot 2^{-k+1}$ by choice of m. □

Proof of ii. Pick $q \in \mathbb{N}$ and $r \in [d-1]$ such that $2^m = q \cdot d + r$. We proceed by splitting the probability of $\Pr[h_x \leq a]$ based on whether or not the integer value of its randomness lies within one of the q intervals of length d. Let U be drawn uniformly at random from $[2^m]$, D be drawn uniformly at random from $\{1, \ldots, d\}$ and $Z \sim \texttt{GeoSample}(k, n, 0)$. For all $x \in \mathcal{X}$ and $a \in [n]$

$$\begin{aligned} \Pr[h_x \leq a] &= \Pr[\min\left\{z \in [n] : F(z) \geq (T(p_h(T^{-1}(x))) \bmod d) + 1\right\} \leq a] \\ &= \Pr[(T(p_h(T^{-1}(x))) \bmod d) + 1 \leq F(a)] \\ &= \Pr[(U \bmod d) + 1 \leq F(a)] \\ &= \frac{q \cdot d}{2^m} \cdot \Pr[D \leq F(a)] + \frac{r}{2^m} \cdot \Pr[D \leq F(a) \mid D \leq r] \\ &\geq \frac{q \cdot d}{2^m} \cdot \Pr[D \leq F(a)] + \frac{r}{2^m} \cdot \Pr[D \leq F(a)] && \text{(by monotonicity of } F) \\ &= \Pr[Z \leq a] \end{aligned}$$

□

Proof of iii. Because $\mathcal{G}'_{k,n}(\mathcal{X})$ is a $(n+1)$-wise independent hash family, for all $B \subseteq \mathcal{X}$ such that $|B| \leq n+1$ and $c \in [n]^B$

$$\begin{aligned}\Pr[\forall x \in B \;\; h_x = c_x] &= \Pr\left[\forall x \in B \;\; F(c_x - 1) < (T(p_h(T^{-1}(x))) \bmod d) + 1 \leq F(c_x)\right] \\ &= \prod_{x \in B} \Pr\left[F(c_x - 1) < (T(p_h(T^{-1}(x))) \bmod d) + 1 \leq F(c_x)\right] \\ &= \prod_{x \in B} \Pr[h_x = v_x]\end{aligned}$$

□

Proof of iv. $\mathbb{F}_{2^m}$ can be represented by an irreducible polynomial of degree m over $\mathbb{F}_2$ encoded as a binary string of length m. Likewise, all elements of $\mathbb{F}_{2^m}$ can be represented by a polynomial of degree at most $m-1$ over $\mathbb{F}_2$ (requires m bits). This encoding defines an efficient bijection T between $\mathbb{F}_{2^m}$ and $[2^m - 1]$ by also interpreting the string as the binary representation of an element in $[2^m - 1]$.

For all $h \in \mathcal{G}_{k,n}(\mathcal{X})$, h can be represented by k, n, the coefficients of p_h and a description of the field $\mathbb{F}_{2^m}$. This representation can be encoded in $\text{poly}(k, n, \log|\mathcal{X}|)$ bits.

Now, given this encoding, evaluation of $p_h(x)$ can be done in $\text{poly}(m) = \text{poly}(k, n, \log|\mathcal{X}|)$ time. And computing

$$h_x = \min\left\{z \in [n] : F(z) \geq (T(p_h(T^{-1}(x))) \bmod d) + 1\right\}$$

to get an approximate count takes $\text{poly}(k, n, \log|\mathcal{X}|)$ time. □

Proof of v. Let $A \subseteq \mathcal{X}$ such that $|A| \leq n$ and $c \in [n]^A$. We can sample h given by the coefficients $a_0, \ldots, a_n \in \mathbb{F}_{2^m}$ uniformly at random from $\mathcal{G}_{k,n}(\mathcal{X})$ such that $h_x = c_x$ for all $x \in A$ with the following steps.

1. Construct $\mathbb{F}_{2^m}$ by finding an irreducible polynomial of degree m over $\mathbb{F}_2$.
2. For each $x \in \mathcal{X}$, sample $\tilde{u}_x$ uniformly at random from the set

   $$S_x = \{y \in \mathbb{F}_{2^m} : c_x = \min\{z \in [n] : F(z) \geq (T(y) \bmod d) + 1\}\}$$

   We can sample from this set by observing that

   $$\begin{aligned}S_x &= \{y \in \mathbb{F}_{2^m} : F(c_x - 1) < (T(y) \bmod d) + 1 \leq F(c_x)\} \\ &= \{y \in \mathbb{F}_{2^m} : \exists i, r \in \mathbb{N} \text{ such that } T(y) = i \cdot d + r \text{ and } F(c_x - 1) < r + 1 \leq F(c_x)\} \\ &= T^{-1}\left(\{i \cdot d + r \in [2^m - 1] : i \in \mathbb{N} \text{ and } F(c_x - 1) < r + 1 \leq F(c_x)\}\right) \\ &= T^{-1}\left(\bigcup_{i=0}^{\lfloor 2^m/d \rfloor - 1} \{i \cdot d + r \in [2^m - 1] : F(c_x - 1) < r + 1 \leq F(c_x)\}\right)\end{aligned}$$

3. Let $B \subseteq \mathcal{X}$ such that $A \subseteq B$ and $|B| = n + 1$. For all $x \in B \setminus A$, sample $\tilde{u}_x$ uniformly at random from $\mathbb{F}_{2^m}$.
4. Take the coefficients $a_0, \ldots, a_n \in \mathbb{F}_{2^m}$ to be the coefficients of the interpolating polynomial over $\mathbb{F}_{2^m}$ given the set of points $(x, \tilde{u}_x)$ for all $x \in B$. This polynomial exists and is unique.

We first prove correctness. Notice this procedure can only return a histogram $h \in \mathcal{G}_{k,n}(\mathcal{X})$ such that $h_x = v_x$ for all $x \in A$. Let h be any such histogram. Then

$$\begin{aligned}
\Pr[\text{Sampling } h] &= \Pr[(a_0, \ldots, a_n) \text{ are the coefficients of } p_h] \\
&= \Pr[\forall x \in B \;\; \tilde{u}_x = p_h(x)] \\
&= \prod_{x \in B} \Pr[\tilde{u}_x = p_h(x)] \\
&= \left(\prod_{x \in A} \frac{1}{|S_x|} \right) \cdot \left(\frac{1}{2^m} \right)^{|B \setminus A|}
\end{aligned}$$

Therefore, these steps output $h \in \mathcal{G}_{k,n}(\mathcal{X})$ uniformly at random such that $h_x = v_x$ for all $x \in A$.

Construction of $\mathbb{F}_{2^m}$ can be done in $O(m^4)$ time [Sho90]. Integer operations are limited to numbers not exceeding 2^m. And polynomial interpolation takes $O(n^2) \cdot \text{poly}(m)$ time [BP70]. Therefore, this procedure runs in time $\text{poly}(k, n, \log |\mathcal{X}|)$. □

Algorithm 3.6 (`BoundedGeometricMechanism`) has on each bin a count with marginal distribution $\texttt{Geosample}(k, n, c_x(D))$ and its counts are independent. By using the previously defined family, the following algorithm has essentially the same marginal distributions as Algorithm 3.6, but the counts are only $(n+1)$-wise independent. This yields an efficient algorithm as we only need a small number of random bits (polynomial in $\log |\mathcal{X}|$) compared to the amount required for all counts to be independent (linear in $|\mathcal{X}|$).

Algorithm 7.2. $\texttt{CompactHistogram}(D, \varepsilon)$ for $D \in \mathcal{X}^n$ and rational $\varepsilon \in (0, 1]$

1. Let $k = \lceil \log(4/\varepsilon) \rceil$.
2. Let $A = \{x \in \mathcal{X} : c_x(D) > 0\}$.
3. For each $x \in A$, let $\tilde{c}_x = \texttt{GeoSample}(k, n, c_x(D))$.
4. Release h drawn uniformly at random from $\{h' \in \mathcal{G}_{k,n}(\mathcal{X}) : \forall x \in A \;\; h'_x = \tilde{c}_x\}$.

Theorem 7.3. *Let rational $\varepsilon \in (0, 1]$ and $\tilde{\varepsilon} = 2 \cdot \ln\left(1 + 2^{-\lceil \log(4/\varepsilon) \rceil}\right) \in (2\varepsilon/9, \varepsilon/2]$. Then* $\texttt{CompactHistogram}(D, \varepsilon)$ *has the following properties:*

i. $\texttt{CompactHistogram}(D, \varepsilon)$ *is $(\varepsilon, 0)$-differentially private.*

ii. $\texttt{CompactHistogram}(D, \varepsilon)$ *has (a, β)-per-query accuracy for*

$$a = \left\lceil \frac{2}{\tilde{\varepsilon}} \ln \frac{1}{\beta} \right\rceil$$

iii. $\texttt{CompactHistogram}(D, \varepsilon)$ *has (a, β)-simultaneous accuracy for*

$$a = \left\lceil \frac{2}{\tilde{\varepsilon}} \ln \frac{|\mathcal{X}|}{\beta} \right\rceil$$

iv. $\texttt{CompactHistogram}(D, \varepsilon)$ *has running time*

$$\text{poly}(N)$$

where N *is the bit length of this algorithm's input* ($D \in \mathcal{X}^n$ *and* ε)

Proof of i. Let $D, D' \in \mathcal{X}^n$ be neighboring datasets. Let $A = \{x \in \mathcal{X} : c_x(D) > 0\}$. Similarly, define $A' = \{x \in \mathcal{X} : c_x(D') > 0\}$. Let $B = A \cup A'$. Notice $|B| \leq n + 1$. Let $g \in \mathcal{G}_{k,n}(\mathcal{X})$. Let h be drawn uniformly at random from $\mathcal{G}_{k,n}(\mathcal{X})$. Then

$$\begin{aligned}
&\Pr[\texttt{CompactHistogram}(D, \varepsilon) = g] \\
&\quad = \Pr[\forall x \in B \;\; \texttt{CompactHistogram}(D, \varepsilon)_x = g_x] \cdot \Pr[h = g \mid \forall x \in B \;\; h_x = g_x] \\
&\quad = \left(\prod_{x \in A} \Pr[Z_x(D) = g_x] \right) \left(\prod_{x \in B \setminus A} \Pr[h_x = g_x] \right) \cdot \Pr[h = g \mid \forall x \in B \;\; h_x = g_x]
\end{aligned}$$

where $Z_x(D) \sim \texttt{GeoSample}(k, n, c_x(D))$. Now, because $|B \setminus A| = 1$ and and $|B \setminus A'| = 1$ along with Proposition 3.3 and part i of Lemma 7.1

$$\begin{aligned}
&\frac{\Pr[\texttt{CompactHistogram}(D, \varepsilon) = g]}{\Pr[h = g \mid \forall x \in B \;\; h_x = g_x]} \\
&\quad = \left(\prod_{x \in A} \Pr[Z_x(D) = g_x] \right) \left(\prod_{x \in B \setminus A} \Pr[h_x = g_x] \right) \\
&\quad \leq e^{(1/3) \cdot 2^{-k+1}} \cdot \prod_{x \in B} \Pr[Z_x(D) = g_x] \\
&\quad \leq e^{\tilde{\varepsilon} + (1/3) \cdot 2^{-k+1}} \cdot \prod_{x \in B} \Pr[Z_x(D') = g_x] \\
&\quad \leq \left(e^{\tilde{\varepsilon} + (1/3) \cdot 2^{-k+1}} \cdot \prod_{x \in A'} \Pr[Z_x(D') = g_x] \right) \left(e^{(2/3) \cdot 2^{-k+1}} \cdot \prod_{x \in B \setminus A'} \Pr[h_x = g_x] \right) \\
&\quad \leq e^{\varepsilon} \cdot \frac{\Pr[\texttt{CompactHistogram}(D', \varepsilon) = g]}{\Pr[h = g \mid \forall x \in B \;\; h_x = g_x]}
\end{aligned}$$

as $\tilde{\varepsilon} \leq \varepsilon/2$ and $2^{-k+1} \leq \varepsilon/2$. Therefore, $\texttt{CompactHistogram}(D, \varepsilon)$ is $(\varepsilon, 0)$-differentially private. □

Proof of ii. Let $h \sim \texttt{CompactHistogram}(D, \varepsilon)$ and $A = \{x \in \mathcal{X} : c_x(D) > 0\}$. Let $x \in A$. Then, by construction, $h_x \sim \texttt{GeoSample}(k, n, c_x(D))$ and (a, β)-per-query accuracy follows from part *ii* of Theorem 3.7.

Let $x \in \mathcal{X} \setminus A$ and h' be drawn uniformly at random from $\mathcal{G}_{k,n}(\mathcal{X})$. Notice $c_x(D) = 0$ and $|A| \leq n$. By parts *ii* and *iii* of Lemma 7.1

$$\begin{aligned}
\Pr[|h_x| \leq a] &= \Pr[h'_x \leq a \mid \forall x \in A \;\; h'_x = h_x] \\
&= \Pr[h'_x \leq a] \\
&\geq \Pr[Z \leq a]
\end{aligned}$$

where $Z \sim \texttt{GeoSample}(k, n, 0)$. Thus, (a, β)-per-query accuracy follows from part *ii* of Theorem 3.7. □

Proof of iii. A union bound over each $x \in \mathcal{X}$ along with the previous part gives a bound on the (a, β)-simultaneous accuracy. □

Proof of iv. `CompactHistogram`(D, ε) makes at most n calls to `GeoSample`. Therefore, by Proposition 3.3 and by part v of Lemma 7.1, we get the desired bound on running time. □

Therefore, we have constructed a histogram algorithm with running time polynomial in and accuracy matching lower bounds for releasing private histograms up to constant factors.

Acknowledgments

We thank the Harvard Privacy Tools differential privacy research group, particularly Mark Bun and Kobbi Nissim, for informative discussions and feedback, and the anonymous TPDP reviewers for helpful comments.

References

[BBKN14] Amos Beimel, Hai Brenner, Shiva Prasad Kasiviswanathan, and Kobbi Nissim. Bounds on the sample complexity for private learning and private data release. *Machine learning*, 94(3):401–437, 2014.

[BLR13] Avrim Blum, Katrina Ligett, and Aaron Roth. A learning theory approach to non-interactive database privacy. *Journal of the ACM (JACM)*, 60(2):12, 2013. URL `https://arxiv.org/pdf/1109.2229.pdf`.

[BNS16] Mark Bun, Kobbi Nissim, and Uri Stemmer. Simultaneous private learning of multiple concepts. In *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science*, pages 369–380. ACM, 2016. URL `https://arxiv.org/pdf/1511.08552.pdf`.

[BP70] Åke Björck and Victor Pereyra. Solution of vandermonde systems of equations. *Mathematics of Computation*, 24(112):893–903, 1970.

[CDK17] Bryan Cai, Constantinos Daskalakis, and Gautam Kamath. Priv'it: Private and sample efficient identity testing. *arXiv preprint arXiv:1703.10127*, 2017. URL `https://arxiv.org/pdf/1703.10127.pdf`.

[CKKL12] Mahdi Cheraghchi, Adam Klivans, Pravesh Kothari, and Homin K Lee. Submodular functions are noise stable. In *Proceedings of the twenty-third annual ACM-SIAM symposium on Discrete Algorithms*, pages 1586–1592. Society for Industrial and Applied Mathematics, 2012. URL `https://arxiv.org/pdf/1106.0518.pdf`.

[CPST11] Graham Cormode, Magda Procopiuc, Divesh Srivastava, and Thanh TL Tran. Differentially private publication of sparse data. *arXiv preprint arXiv:1103.0825*, 2011. URL `https://arxiv.org/pdf/1103.0825.pdf`.

[CTUW14] Karthekeyan Chandrasekaran, Justin Thaler, Jonathan Ullman, and Andrew Wan. Faster private release of marginals on small databases. In *Proceedings of the 5th conference on Innovations in theoretical computer science*, pages 387–402. ACM, 2014. URL `https://arxiv.org/pdf/1304.3754.pdf`.

[DL09] Cynthia Dwork and Jing Lei. Differential privacy and robust statistics. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 371–380. ACM, 2009.

[DMNS06] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of Cryptography Conference*, pages 265–284. Springer, 2006.

[DNT15] Cynthia Dwork, Aleksandar Nikolov, and Kunal Talwar. Efficient algorithms for privately releasing marginals via convex relaxations. *Discrete & Computational Geometry*, 53(3):650–673, 2015. URL `https://arxiv.org/pdf/1308.1385.pdf`.

[GMP16] Ivan Gazeau, Dale Miller, and Catuscia Palamidessi. Preserving differential privacy under finite-precision semantics. *Theoretical Computer Science*, 655:92–108, 2016. URL `https://arxiv.org/pdf/1306.2691.pdf`.

[GRS12] Arpita Ghosh, Tim Roughgarden, and Mukund Sundararajan. Universally utility-maximizing privacy mechanisms. *SIAM Journal on Computing*, 41(6):1673–1693, 2012.

[GRU12] Anupam Gupta, Aaron Roth, and Jonathan Ullman. Iterative constructions and private data release. *Theory of Cryptography*, pages 339–356, 2012. URL `https://arxiv.org/pdf/1107.3731.pdf`.

[HRS12] Moritz Hardt, Guy N Rothblum, and Rocco A Servedio. Private data release via learning thresholds. In *Proceedings of the twenty-third annual ACM-SIAM symposium on Discrete Algorithms*, pages 168–187. Society for Industrial and Applied Mathematics, 2012. URL `https://arxiv.org/pdf/1107.2444.pdf`.

[HT10] Moritz Hardt and Kunal Talwar. On the geometry of differential privacy. In *Proceedings of the forty-second ACM symposium on Theory of computing*, pages 705–714. ACM, 2010. URL `https://arxiv.org/pdf/0907.3754.pdf`.

[KLN+11] Shiva Prasad Kasiviswanathan, Homin K Lee, Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. What can we learn privately? *SIAM Journal on Computing*, 40(3):793–826, 2011. URL `https://arxiv.org/pdf/0803.0924.pdf`.

[Knu05] Donald E Knuth. *Combinatorial Algorithms, Part 1, volume 4 of The Art of Computer Programming*. Addison-Wesley Professional, 2005.

[Mir12] Ilya Mironov. On significance of the least significant bits for differential privacy. In *Proceedings of the 2012 ACM conference on Computer and communications security*, pages 650–661. ACM, 2012.

[Sho90] Victor Shoup. New algorithms for finding irreducible polynomials over finite fields. *Mathematics of Computation*, 54(189):435–447, 1990.

[TUV12] Justin Thaler, Jonathan Ullman, and Salil Vadhan. Faster algorithms for privately releasing marginals. In *International Colloquium on Automata, Languages, and Programming*, pages 810–821. Springer, 2012. URL `https://arxiv.org/pdf/1205.1758.pdf`.

[UV11] Jonathan Ullman and Salil P Vadhan. PCPs and the hardness of generating private synthetic data. In *TCC*, volume 6597, pages 400–416. Springer, 2011. URL `https://pdfs.semanticscholar.org/bf6f/e68b91283787b9ad28494a78b4cead10fa12.pdf`.

A1 Generating a Sparse Histogram Uniformly at Random

Being able to efficiently compute $|\mathcal{H}_{n,n'}(\mathcal{X})|$ and sample a uniformly random element from $\mathcal{H}_{n,n'}(\mathcal{X})$ is needed for an efficient implementation of Algorithm 4.13.

Lemma A1.1. *$|\mathcal{H}_{n,n'}(\mathcal{X})|$ can be calculated and a uniformly random element of $\mathcal{H}_{n,n'}(\mathcal{X})$ can be sampled in time*

$$\text{poly}(n', \log n, \log |\mathcal{X}|)$$

Proof. We show $\texttt{HistSample}(\mathcal{X}, n, n')$ defined below efficiently samples a uniformly random element of $\mathcal{H}_{n,n'}(\mathcal{X})$ by using a bijection between $\mathcal{H}_{n,n'}(\mathcal{X})$ and $\{1, \ldots, |\mathcal{H}_{n,n'}(\mathcal{X})|\}$.

Algorithm A1.2. $\texttt{HistSample}(\mathcal{X}, n, n')$ for $n, n' \in \mathbb{N}_+$ with $n' \leq |\mathcal{X}|$

1. Pick U uniformly at random from $\{1, \ldots, |\mathcal{H}_{n,n'}(\mathcal{X})|\} = \{1, \ldots, \sum_{i=0}^{n'} \binom{|\mathcal{X}|}{i} n^i\}$.
2. Find the smallest $\ell \in [n']$ such that $\sum_{i=0}^{\ell} \binom{|\mathcal{X}|}{i} n^i \geq U$.
3. Let $u' = U - 1 - \sum_{i=0}^{\ell-1} \binom{|\mathcal{X}|}{i} n^i \in \left[\binom{|\mathcal{X}|}{\ell} \cdot n^\ell - 1\right]$.
4. Use integer division to find $q \in \left[\binom{|\mathcal{X}|}{\ell} - 1\right]$ and $r \in \left[n^\ell - 1\right]$ such that $u' = qn^\ell + r$.
5. Map q to a corresponding subset of $\mathcal{X}$ of size ℓ specified by sorted elements $q_1 < \ldots < q_\ell$. Specifically, we can let $q_1, \ldots, q_\ell \in \mathcal{X}$ be the sequence representing q in the combinatorial number system of degree ℓ, i.e. the unique sequence satisfying

   $$q = \binom{q_\ell}{\ell} + \ldots + \binom{q_1}{1}$$

   This sequence can be found greedily; for each j decreasing from ℓ to 1, using binary search, find the largest q_j such that $\sum_{i=j}^{\ell} \binom{q_i}{i} \leq q$ [Knu05].
6. Let $r_0, \ldots, r_{\ell-1} \in [n-1]$ be the digits of the base n representation of r, i.e.

   $$r = r_{\ell-1} n^{\ell-1} + \ldots + r_1 n + r_0$$

7. For each $i \in \{1, \ldots, \ell\}$, release $(q_i, r_{i-1} + 1)$.

Steps 3 and 4 define a bijection mapping u to (ℓ, u'). Step 5 defines a bijection mapping (ℓ, u') to (ℓ, q, r). In steps 6 and 7, the decompositions of q and r into their respective number systems are bijections. Likewise, there is a bijection mapping $(q_1, \ldots, q_\ell, r_0, \ldots, r_{\ell-1})$ to an element of $\mathcal{H}_{n,n'}(\mathcal{X})$. Therefore, our mapping from $\{1, \ldots, |\mathcal{H}_{n,n'}(\mathcal{X})|\}$ to $\mathcal{H}_{n,n'}(\mathcal{X})$ is bijective proving correctness.

Computing $|\mathcal{H}_{n,n'}(\mathcal{X})| = \sum_{i=0}^{n'} \binom{|\mathcal{X}|}{i} n^i$ takes $\text{poly}(n', \log n, \log |\mathcal{X}|)$ time. The remaining steps can be done in $\text{poly}(n', \log n, \log |\mathcal{X}|)$ time as no number used exceeds $|\mathcal{H}_{n,n'}(\mathcal{X})|$, all numbers used are expressible as the sum of at most n' other numbers and steps 5-7 each consist of at most n' iterations. ∎