Tight Data Access Bounds for Private Top-k Selection

Hao Wu [1] **Olga Ohrimenko** [1] **Anthony Wirth** [1]

Abstract

We study the top-k selection problem under the differential privacy model: m items are rated according to votes of a set of clients. We consider a setting in which algorithms can retrieve data via a sequence of accesses, each either a random access or a sorted access; the goal is to minimize the total number of data accesses. Our algorithm requires only $O(\sqrt{mk})$ expected accesses: to our knowledge, this is the first sublinear data-access upper bound for this problem. Our analysis also shows that the well-known exponential mechanism requires only $O(\sqrt{m})$ expected accesses. Accompanying this, we develop the first lower bounds for the problem, in three settings: only random accesses; only sorted accesses; a sequence of accesses of either kind. We show that, to avoid $\Omega(m)$ access cost, supporting *both* kinds of access is necessary, and that in this case our algorithm's access cost is optimal.

1. Introduction

We consider the differentially private top-k selection problem; there are m items to be rated according to n clients' votes. Each client can either vote or not vote for each item, and can vote for an unlimited number of items. Since this data can be sensitive (e.g., visited websites, purchased items, or watched movies), the goal is to identify a set of k items with approximately the highest number of votes, while concealing the votes of individual clients.

Private top-k selection is a fundamental primitive and underlies a wide range of differentially private machine learning and data analytics tasks such as discovering frequent patterns from data (Bhaskar et al., 2010), training wide neural networks (Zhang et al., 2021), tracking data streams (Cardoso & Rogers, 2022), false discovery rate control in hypothesis testing (Qiao et al., 2021), etc.

In recent years, a significant progress has been made towards understanding how accurate the algorithms for this problem can be. For example, (Bafna & Ullman, 2017; Steinke & Ullman, 2017) provide lower bounds for the problem in terms of sample complexity, which can be achieved by a number of existing algorithms (Durfee & Rogers, 2019; Qiao et al., 2021).

Another line of research is devoted to improving the efficiency of the algorithms. Early works such as the peeling solution (Bhaskar et al., 2010) need to iterate k times over all items. The improved mechanisms (Durfee & Rogers, 2019; Qiao et al., 2021) iterate over each item only once. Since k can be much smaller than m, the research community remains interested in the following question:

> Is there a private top-k selection algorithm that accesses only a sublinear number of items?

Although it seems to be an unachievable target, it is possible to address this question by considering how items are accessed. For example, Durfee & Rogers (2019) consider the setting where the data has been pre-processed and resides in an existing data analytics system, that can return the items in sorted order (which we refer to as *sorted access* in our paper). Their top-k algorithm can make a sublinear number of accesses at a cost of potentially returning fewer than k items, while to guarantee that k items are returned, the number of retrieved items can be m. Since retrieving information from an existing system incurs corresponding query processing and communication cost, it is crucial to minimize the number of data accesses.

In this paper, we systematically investigate the minimum number of items an algorithm needs to evaluate (a.k.a. *access cost*) , in order to answer the private top-k selection problem. In addition to *sorted access*, we also consider another common way of accessing items' data, i.e., the *random access*, in which an algorithm can actively request the data of an arbitrary item.[1] Both types of accesses have been considered by previous literature for the non-private version of top-k selection problem (see Ilyas et al. (2008) for a comprehensive survey).

[1]School of Computing and Information Systems, The University of Melbourne. Correspondence to: Hao Wu <whw4@student.unimelb.edu.au>, Olga Ohrimenko <oohrimenko@unimelb.edu.au>, Anthony Wirth <awirth@unimelb.edu.au>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023.

[1]Here *random* carries the sense of Random Access Memory (RAM), rather than the outcome of a random process.

Example 1.1. Consider the example of a movie ranking database. It can present the movies in sorted order, according to their ratings by the clients, or it can return directly the rating of a specific movie.

Our Contributions. Our results are threefold. On the upper bound side,

- If the system supports both sorted access and random access, we design an algorithm with expected access cost $O(\sqrt{mk})$.

To our knowledge, this is the first asymptotically sublinear bound of the access cost for the private top-k selection problem. Our algorithm builds on existing works (Durfee & Rogers, 2019; Qiao et al., 2021) and inherits their error bounds, which are known to be asymptotically optimal (Bafna & Ullman, 2017; Steinke & Ullman, 2017). Additionally, since the exponential mechanism (McSherry & Talwar, 2007), a fundamental technique in differential privacy, can be formulated as a private top-1 selection algorithm (Durfee & Rogers, 2019), our result implies the following corollary:

- If the system supports both sorted access and random access, the exponential mechanism requires only $O(\sqrt{m})$ expected accesses.

On the lower bound side,

- If the system supports either only sorted accesses or only random accesses, but not both, we show a lower bound of $\Omega(m)$.
- If the system supports both sorted accesses and random accesses, we show a lower bound of $\Omega(\sqrt{mk})$.

These statements are informal versions of Theorems 5.1, 5.3, and 5.4, which impose modest assumptions on the privacy guarantee, and relatively weak assumptions on the accuracy guarantee of the algorithms. They show that supporting sorted and random access to the items' data simultaneously is necessary to break the linear barrier, and the access cost of our algorithm is essentially optimal.

Organization. Our paper is organized as follows. Section 2 introduces the problem formally. Section 3 discusses the preliminaries for our algorithm. Section 4 introduces our algorithm and shows the upper bounds. Section 5 presents the lower bounds for the problem. Section 6 discusses the related works. Section 7 summarizes the paper.

2. Model Description

Let $\mathcal{C} \doteq \{1, \ldots, m\}$ be a set of m items, and $\mathcal{U} \doteq \{1, \ldots, n\}$ be a set of n clients. Each client $v \in \mathcal{U}$ can cast at most one vote for each item, and can vote for an unlimited number of items. Hence, client v's votes, denoted by $\vec{x}_v$, can be viewed as a vector in $\mathcal{D} \doteq \{0,1\}^m$, such that for each $i \in \mathcal{C}$, $\vec{x}_v[i] = 1$ if v votes for item i, where $\vec{x}_v[i]$ is the $i^{(th)}$ entry of $\vec{x}_v$. We regard the collection of voting vectors from all n clients as a dataset $\mathcal{X} = \{\vec{x}_1, \ldots, \vec{x}_n\} \in \mathcal{D}^n$.

For each item $i \in \mathcal{C}$, let its score $\vec{h}[i] \doteq \sum_{v \in \mathcal{U}} \vec{x}_v[i]$ be the number of clients that vote for i. The dataset $\mathcal{X}$ can be described by its histogram $\vec{h} \doteq (\vec{h}[1], \ldots, \vec{h}[m]) \in \mathbb{N}^m$. We also define $\pi : \mathcal{C} \to \mathcal{C}$ to be a permutation that puts the entries of $\vec{h}$ in nonincreasing order[2], s.t., $\vec{h}[\pi(1)] \geq \cdots \geq \vec{h}[\pi(m)]$.

Our goal is to design a differentially private algorithm that returns a set S of k items with (approximately) largest scores, while minimizing its data access cost.

In what follows, we discuss the privacy guarantee, the utility guarantee, the data access model of an algorithm formally.

Privacy Guarantee. We call two datasets $\mathcal{X}, \mathcal{X}'$ neighboring, denoted by $\mathcal{X} \sim \mathcal{X}'$, if they differ in addition or deletion of one client vector, e.g., $\mathcal{X}' = \mathcal{X} \cup \{\vec{x}_{n+1}\}$ or $\mathcal{X}' = \mathcal{X} \setminus \{\vec{x}_v\}$ for some $v \in [n]$.

Let $\vec{h}$ and $\vec{h}'$ be the histograms corresponding to $\mathcal{X}$ and $\mathcal{X}'$, respectively. It is easy to see that if $\mathcal{X} \sim \mathcal{X}'$, then the score of each item can differ by at most 1, i.e., $\|\vec{h} - \vec{h}'\|_\infty \leq 1$. Hence, for every $\vec{h}, \vec{h}' \in \mathbb{N}^m$, we also call them neighboring histograms, written as $\vec{h} \sim \vec{h}'$, if and only if $\|\vec{h} - \vec{h}'\|_\infty \leq 1$.

Let $\binom{[m]}{k}$ be the collection of all subsets of $[m]$ of size k, and $\mathcal{A} : \mathbb{N}^m \to \binom{[m]}{k}$ be a top-k selection algorithm. To protect the voting information of individual clients, we would like its output distributions to be similar for neighboring inputs, as defined thus.

Definition 2.1 ((ε, δ)-Private Algorithm (Dwork & Roth, 2014))**.** Given $\varepsilon, \delta > 0$, a randomized algorithm $\mathcal{A} : \mathbb{N}^m \to \binom{[m]}{k}$ is called (ε, δ)-differentially private (DP), if for every $\vec{h}, \vec{h}' \in \mathbb{N}^m$ such that $\vec{h} \sim \vec{h}'$, and all $Z \subseteq \binom{[m]}{k}$,

$$\Pr\left[\mathcal{A}(\vec{h}) \in Z\right] \leq e^{\varepsilon} \cdot \Pr[\mathcal{A}(\vec{h}') \in Z] + \delta\,. \qquad (1)$$

We call ε and δ the *privacy parameters*. Typically, it is required that δ is cryptographically negligible, i.e., $\delta \leq 1/m^{\omega(1)}$ (Vadhan, 2017; Dwork & Roth, 2014). An algorithm $\mathcal{A}$ is also called ε-DP for short, if it is $(\varepsilon, 0)$-DP.

Utility Guarantee. In line with previous research (Bafna & Ullman, 2017; Durfee & Rogers, 2019), we measure the error of an output S by the maximum amount by which $\vec{h}[\pi(k)]$ exceeds the score of any item in S, defined formally as follows.

[2]We break ties arbitrarily.

Definition 2.2 ((α, k)-Accuracy)**.** Given a vector $\vec{h}$, parameters $k \in \mathbb{N}^+$, and $\alpha \in \mathbb{R}^+$, an output $S \in \binom{[m]}{k}$ is called (α, k)-accurate, if for each $i \in S$, $\vec{h}[i] \geq \vec{h}[\pi(k)] - \alpha$.

Data Access. We assume that the histogram $\vec{h}$ has been preprocessing by an existing data management system, and an algorithm $\mathcal{A}$ can access $\vec{h}$ only through the system. We consider two access models that abstract common functionalities supported by a system: *sorted access* and *random access*. Such access models have been widely accepted by the community for non-private top-k selection problems (see Ilyas et al. (2008) for a survey).

Sorted Access. Let $\mathcal{C}_s$ be the set of items already returned by sorted access (initially, $\mathcal{C}_s = \varnothing$). When a new sorted-access request is submitted, the system returns an item-score pair $(i, \vec{h}[i])$, where $i \in [m] \setminus \mathcal{C}_s$ has the largest score, i.e., $i = \arg\max_{j \in [m] \setminus \mathcal{C}_s} \vec{h}[j]$. An alternative view is that the system returns $\big(\pi(1), \vec{h}[\pi(1)]\big), \big(\pi(2), \vec{h}[\pi(2)]\big), \ldots$ in order, one tuple at a time.

Random Access. A request of random access consists of a reference $i \in \mathcal{C}$ to an item. In response, the system returns the corresponding item-score pair $(i, \vec{h}[i])$. We emphasize that a random access does not imply that i must be a randomly chosen item.

Access Cost. Given an algorithm $\mathcal{A}$ and a histogram $\vec{h}$, the *access cost* of an algorithm on $\vec{h}$, $cost\left(\mathcal{A}, \vec{h}\right)$, is the total number of accesses – either sorted or random – to $\vec{h}$. Note that this is an upper bound of distinct number of entries $\mathcal{A}$ learns from $\vec{h}$, as a random access may retrieve a previously encountered item-score pair.

3. Preliminaries

In this section, we review two building blocks for constructing our algorithm: a state-of-the-art algorithm for non-private top-k selection, specifically designed for aggregating data from multiple sources, and a framework of the existing one-shot algorithms for private top-k selection.

3.1. Threshold Algorithm

The threshold algorithm (Fagin et al., 2003) is a top-k selection algorithm, when the information of an item needs to be aggregated from multiple resources. In this scenario, there are m items, each associated with t attributes. Without loss of generality, assume that each attribute is a real number. Therefore, each item i can be represented by a vector $\vec{y}_i \in \mathbb{R}^t$. The score of item i is computed by a function $f : \mathbb{R}^t \to \mathbb{R}$, which is assumed to be *monotone*, s.t., for each $\vec{y}, \vec{y}' \in \mathbb{R}^t$, if $\vec{y}[j] \leq \vec{y}'[j], \forall j \in [t]$, then $f(\vec{y}) \leq f(\vec{y}')$.

The vectors $\vec{y}_1, \ldots, \vec{y}_m$ do not reside in a single data management system, but distributed in t systems $L_1, \ldots, L_t$, s.t., for each item i, its j^{th} attribute $\vec{y}_i[j]$ resides on L_j. Each L_j allows for both *sorted access* and *random access*. We can view it as an array of m tuples $L_j[1], \ldots, L_j[m]$, each of the form $(i, \mathsf{val}) \in [m] \times \mathbb{R}$, where val equals $\vec{y}_i[j]$. The tuples in L_j are sorted in descending order by their val's. Further, L_j is augmented with an inverted index $\sigma_j : [m] \to [m]$ to support random access, such that, for each an item $i \in [m]$, $L_j[\sigma_j(i)]$ contains the tuple $(i, \vec{y}_i[j])$.

The aim is to identify the top-k items with highest scores according to f, while minimizing the access cost, i.e., the total number of data accesses performed by the algorithm to $L_1, \ldots, L_t$. The algorithm is described in Algorithm 1.

It works in round-robin fashion. In each round, it retrieves one tuple from each sorted array L_j. For each tuple (i, val) encountered during sorted access, it retrieves all entries $\vec{y}_i[j]$ of $\vec{y}_i$ by random accesses. This step can be optimized at a cost of memoization: we augment $\mathcal{A}_{TA}$ with a data structure to store previously encountered i's. After retrieving all $\vec{y}_i[j]$, the algorithm computes $f(\vec{y}_i)$, and maintains a set S, consisting of k item-score pairs with largest scores seen so far. The algorithm stops when there are k tuples in S with score at least $\tau \doteq f(\underline{y}_1, \ldots, \underline{y}_m)$, where the $\underline{y}_j$ is the score of the last item in L_j retrieved under sorted access.

Algorithm 1 Threshold Algorithm $\mathcal{A}_{TA}$ (Fagin et al., 2003)

Input: Sorted array L_j and inverted index σ_j, $\forall j \in [t]$.
$S \leftarrow \varnothing$.
repeat
  for each $j \in [t]$ **do**
    Retrieve a tuple from L_j via sorted access, and denote the returned tuple as (i, val).
    Retrieve from $L_1, \ldots, L_t$ by random access (with the help of $\sigma_1, \ldots, \sigma_t$) all attributes of item i, to compute $f(\vec{y}_i)$.
    If $f(\vec{y}_i)$ is among the-k highest scores seen so far, add $(i, f(\vec{y}_i))$ to S; if $|S| > k$, remove the tuple with lowest score from S .
  end for
  For each L_j, let $\underline{y}_j \doteq \vec{y}_i[j]$, where i is the last item seen in L_j under sorted access.
  Define the threshold $\tau \doteq f(\underline{y}_1, \ldots, \underline{y}_m)$.
until there are k tuples in S with score at least τ.
Return the set of items contained in the tuples in S.

The correctness of the algorithm is obvious: when the algorithm stops, since f is monotone, the scores of all unseen items are at most τ, which are lower than the scores of all tuples in S.

Access Cost. Fagin et al. (2003) did not provide asymptotic bound for the access cost. Instead, they proved that $\mathcal{A}_{TA}$ is *instance optimal*. Informally, instance optimally implies that for every algorithm $\mathcal{A}$ which solves the top-k selection problem correctly and whose first access to an item must

be sorted access as opposed to random access, the access cost of $\mathcal{A}_{TA}$ is at most the access cost of $\mathcal{A}$ (up to some multiplicative constant). In Section 4, we apply a different technique to asymptotically bound the access cost of our algorithm.

3.2. One-shot Private Top-k Algorithm

We review an existing framework for the differentially private top-k selection algorithms (Durfee & Rogers, 2019; Qiao et al., 2021). The framework, described in Algorithm 2, does not consider a specific data access model, and instead needs to learn all entries of $\vec{h}$.

Algorithm 2 Private Top-k Algorithm $\mathcal{M}$

1: **Input:** vector $\vec{h}$
2: **for** each item $i \in [m]$ **do**
3: $\quad \vec{v}[i] \leftarrow \vec{h}[i] + Z_i$, where Z_i is an independent noise random variable;
4: **end for**
5: Return a set S of k items that the maximizes the $\vec{v}[i]$'s.

Definition 3.1 (Noise Distributions)**.** Given parameter $b \in \mathbb{R}$, the Laplace distribution, $\mathbb{L}\mathrm{ap}\,(b)$, and the Gumbel distribution, $\mathbb{G}\mathrm{umbel}\,(b)$, have probability density functions $p(z) = \frac{1}{2b} \cdot \exp\left(-\frac{|z|}{b}\right)$, $\forall z \in \mathbb{R}$, and $p(z) = \frac{1}{b} \cdot \exp\left(-\left(\frac{z}{b} + \exp\left(-\frac{z}{b}\right)\right)\right)$, $\forall z \in \mathbb{R}$, respectively.

Candidates noise distributions for Z_i in Algorithm 2 include $\mathbb{L}\mathrm{ap}\,(1/\varepsilon)$ (Qiao et al., 2021) and $\mathbb{G}\mathrm{umbel}\,(1/\varepsilon)$ (Durfee & Rogers, 2019). The corresponding privacy guarantees, are stated as follows.

Fact 3.2 ((Qiao et al., 2021)). Assume that $Z_i \sim \mathbb{L}\mathrm{ap}\,(1/\varepsilon)$, then $\mathcal{M}$ is $2k\varepsilon$-DP. Given $\delta \in [0, 0.05]$, if it holds that $m \geq 2$ and $8\varepsilon\sqrt{k \log(m/\delta)} \leq 0.2$, then $\mathcal{M}$ also satisfies $\left(8\varepsilon\sqrt{k \log(m/\delta)}, \delta\right)$-DP.

Fact 3.3 ((Durfee & Rogers, 2019)). Assume that $Z_i \sim \mathbb{G}\mathrm{umbel}\,(1/\varepsilon)$. For each $\delta \in [0, 1]$, $\mathcal{M}$ is (ε'', δ)-DP, where $\varepsilon'' \doteq \min\left\{k\varepsilon, k\varepsilon\left(\frac{e^\varepsilon - 1}{e^\varepsilon + 1}\right) + \varepsilon\sqrt{k \ln \frac{1}{\delta}}\right\}$.

Next, we discuss their utility guarantees.

Fact 3.4. Given $\beta \in (0, 1)$, if the $Z_i \sim \mathbb{L}\mathrm{ap}\,(1/\varepsilon)$, or $\mathbb{G}\mathrm{umbel}\,(1/\varepsilon)$, then with probability at least $1 - \beta$, the returned solution by Algorithm 2 is (α, k)-accurate, for $\alpha \in O\left(\frac{\ln(m/\beta)}{\varepsilon}\right)$.

Remark 3.5. Compared to Laplace noise, the Gumbel noise allows the algorithm to return a ranked list of indices, instead of a set which contains on order information. For consistency of presentation, we assume that Algorithm 2 returns a set for both choices.

4. Sublinear Access and Time Algorithm

In this section, we present an algorithm for top-k selection problem, which achieves optimal privacy-utility tradeoffs, and with high probability, has an expected access cost $O(\sqrt{mk})$ and computation time $O(\sqrt{mk}\log\log m)$. Our presentation follows two steps: we first present a strawman algorithm with sublinear access cost but only linear computation; next we show how to improve its time complexity to $O(\sqrt{mk}\log\log m)$.

4.1. A Strawman Approach

A natural idea is to combine the threshold algorithm $\mathcal{A}_{TA}$ with the oneshot private top-k algorithm. Each item $i \in [m]$ now has two attributes, namely, $\vec{h}[i]$ and Z_i, where Z_i is sampled independently from $\mathbb{L}\mathrm{ap}\,(1/\varepsilon)$ or $\mathbb{G}\mathrm{umbel}\,(1/\varepsilon)$.

Since the histogram $\vec{h}$ is stored in a database management system, which allows for two types of access: sorted access and random access, we can think of this as a sorted array L_1 of m tuples, each of the form $(i, \mathsf{val}) \in [m] \times \mathbb{R}$, where val equals $\vec{h}[i]$. The tuples in L_1 are sorted in descending order by their val's. Further, L_1 has an inverted index σ_1 to support random access.

Additionally, we can construct another sorted array L_2 of m tuples, each of the form $(i, \mathsf{val}) \in [m] \times \mathbb{R}$, where val equals Z_i. The tuples in L_2 are also sorted in descending order by their val's. L_2 also has an inverted index σ_2 to support random access.

Then we can run the algorithm $\mathcal{A}_{TA}$, with input L_1, L_2, σ_1, σ_2, and an aggregating function $f(\vec{h}[i], Z_i) \doteq \vec{h}[i] + Z_i$. It is easy to see that f is monotone. The pseudo-code is in Algorithm 3.

Algorithm 3 Private Threshold Algorithm $\mathcal{A}_{PrivTA}$

1: Let $I_1 = 1, I_2 = 2, \ldots, I_m = m$. Generate m tuples $(I_1, Z_1), \ldots, (I_m, Z_m)$, where the Z_i's are i.i.d. random variables; sort the tuples in descending order by the values of the Z_i's, denote the sorted sequence by $\left(I_{(1)}, Z_{(1)}\right), \ldots, \left(I_{(m)}, Z_{(m)}\right)$, and store this sequence in an array L_2; construct $\sigma_2 : [m] \to [m]$, s.t., $L_2[\sigma_2(i)] = (i, Z_i)$ for each item $i \in [m]$.
2: Run Algorithm 1 on input $L_1, L_2, \sigma_1, \sigma_2$, with an aggregation function $f\left((\vec{h}[i], Z_i)\right) \doteq \vec{h}[i] + Z_i$;

Privacy and Utility Guarantee. The privacy and utility guarantee of the algorithm inherits directly from Algorithm 2.

Access and Time Complexity. It is easier to first discuss the time complexity and then the access cost. Generating the random variables takes $O(m)$ time, and sorting them takes $O(m \log m)$ time. Hence the total running time is bounded by $O(m \log m)$.

It remains to discuss the number of accesses the algorithm performs on L_1. Our analysis relies on the following important observation.

Lemma 4.1. *The $I_{(1)}, \ldots, I_{(m)}$ are distributed uniformly over all possible permutations over $[m]$, and are independent of the random variables $Z_{(1)}, \ldots, Z_{(m)}$.*

Intuitively, the claim holds since each Z_i in Algorithm 3 follows the same distribution independently. The proof of the lemma is included in Appendix B.1.

Theorem 4.2. *The expected access cost of Algorithm 3 on L_1, $\mathbb{E}\left[cost\left(\mathcal{A}_{PrivTA}, L_1\right)\right]$, is bounded by $O\left(\sqrt{mk}\right)$.*

The rigorous proof of the Theorem is presented in Appendix B.1. Here we offer an intuitive and informal explanation. Let $\mathcal{S}_r \doteq \left\{i_{(1)}, \ldots, i_{(r)}\right\}$ be the top-r items with highest scores in $\vec{h}$, and $I_{(1:r)} = \left\{I_{(1)}, \ldots, I_{(r)}\right\}$ be the items in the first r tuples in the array L_2. Since $I_{(1:r)}$ is a uniform random subset of $[m]$, $\mathcal{S}_r \cap I_{(1:r)}$ has expected size $r \cdot (r/m) = \left(r^2/m\right)$, which equals k when $r = \sqrt{mk}$. Applying a technique introduced by Fagin (1999), we can show that when $\left|\mathcal{S}_r \cap I_{(1:r)}\right| \geq k$, the algorithm will not access any item outside $\mathcal{S}_r \cup I_{(1:r)}$, since any such item will have a score lower than or equal to any item in $\mathcal{S}_r \cap I_{(1:r)}$. Therefore the algorithm should have access cost roughly $O\left(\sqrt{mk}\right)$.

Application. We discuss an interesting application of our algorithm to the exponential mechanism (McSherry & Talwar, 2007), a fundamental technique in differential privacy to choose a single item from a set of items.

Following the setup in this paper, the exponential mechanism works as follows: it selects an item $i \in [m]$ with probability proportional to $e^{\varepsilon \cdot \vec{h}[i]}$. Moreover, Durfee & Rogers (2019) show that the exponential mechanism is equivalent to Algorithm 2 with $k = 1$ and $Z_i \sim \mathbb{G}\text{umbel}\left(1/\varepsilon\right), \forall i \in [m]$. This variant of Algorithm 2 is commonly referred to as the Report-Noisy-Max algorithm with Gumbel noise. Applying the same k and the Z_i's to Algorithm 3, Theorem 4.2 immediately implies the following corollary.

Corollary 4.3. *When given access to both sorted and random access to data, the exponential mechanism has expected access cost $O(\sqrt{m})$.*

4.2. An Online Sampling Approach

Pre-generating all m noise values may be excessive. For problems with small values of k, e.g., $k = 10$, the Algorithm 3 may need to know only a small subset of tuples in L_2. It is of interest whether we can also reduce the expected number of noisy random variables generated to $O(\sqrt{mk})$, by constructing the L_2 (and σ_2) on the fly.

One can consider applying existing algorithms (such as those presented in Lurie & Hartley 1972; Devroye 1986) to generate the random variables $\left(I_{(1)}, Z_{(1)}\right), \ldots, \left(I_{(m)}, Z_{(m)}\right)$ sequentially, one tuple at a time, each taking $O(1)$ time. However, since the threshold algorithm relies on non-sequential accesses to the variables (due to the *random access* operation), these algorithms cannot be applied to reduce the number of variables generated to $O\left(\sqrt{mk}\right)$. In this section, instead, we present an algorithm that can generate the variables in an arbitrary order and "on the fly". The main result of this section is stated as follows.

Theorem 4.4. *There is an algorithm $\mathcal{A}_{oracle}$, that,*

- *does not require to pre-generate L_2;*
- *answers sorted access and random access query to L_2 in $O(\log \log m)$ time in expectation.*

Further, the tuples returned by $\mathcal{A}_{oracle}$ have the same marginal distribution as those generated by Algorithm 3.

There are two key ingredients for constructing $\mathcal{A}_{oracle}$.

Sampling the $I_{(j)}$'s. The first ingredient is Lemma 4.1, which allows $\mathcal{A}_{oracle}$ to sample the $I_{(j)}$ and the $Z_{(j)}$ independently according to their marginal distributions, without changing the joint distribution of the $I_{(j)}$ and the $Z_{(j)}$. The lemma states that the $I_{(j)}$'s are distributed uniformly over all possible permutations over $[m]$. It is not hard to sample an $I_{(j)}$ on the fly: let $\mathcal{J}$ be the set of indexes such that the values of $I_{(j')}, j' \in \mathcal{J}$ have been determined; if $j \notin \mathcal{J}$, then $I_{(j)}$ just distributes uniformly over the subset of unseen items, i.e., $[m] \setminus I_{(\mathcal{J})}$, where $I_{(\mathcal{J})} \doteq \left\{I_{(j')} : j' \in \mathcal{J}\right\}$. Correspondingly, we can also construct the inverted index σ_2 on the fly: given an item $i \in [m]$, if it has not been encountered, then $\sigma_2(i)$ should equal one of the undetermined indexes, namely $[m] \setminus \mathcal{J}$, uniformly at random.

Sampling the $Z_{(j)}$'s. The second ingredient is an algorithm $\mathcal{A}_{ord\text{-}stat}$ which generates the $Z_{(j)}$'s on the fly. Formally, for each $\mathcal{J} \subseteq [m]$, define $Z_{(\mathcal{J})} \doteq (Z_{(j)}, j \in \mathcal{J})$, and let $z_{(\mathcal{J})}$ refer to a vector $(z_{(j)}, j \in \mathcal{J}) \in \mathbb{R}^{|\mathcal{J}|}$. Denote by $p_{Z_{(\mathcal{J})}}(\cdot)$ the marginal density of $Z_{(\mathcal{J})}$, induced by the generating procedure of Algorithm 3. Call $z_{(\mathcal{J})}$ a feasible realization of $Z_{(\mathcal{J})}$, if $p_{Z_{(\mathcal{J})}}(z_{(\mathcal{J})}) > 0$. Given such a feasible realization, let $p_{Z_{(j)}|Z_{(\mathcal{J})}}(z_{(j)} \mid z_{(\mathcal{J})})$ be the density function of $Z_{(j)}$, conditioned on $Z_{(\mathcal{J})} = z_{(\mathcal{J})}$. The property of $\mathcal{A}_{ord\text{-}stat}$ is stated as follows.

Lemma 4.5. *For each $\mathcal{J} \subseteq [m]$ s.t., $\mathcal{J} \neq [m]$, each $j \in [m] \setminus \mathcal{J}$, and each feasible realization $z_{(\mathcal{J})}$ of $Z_{(\mathcal{J})}$, $\mathcal{A}_{ord\text{-}stat}$ samples a random variable with the conditional density $p_{Z_{(j)}|Z_{(\mathcal{J})}}(z_{(j)} \mid z_{(\mathcal{J})})$ in $O(\log \log m)$ expected time.*

The proof of the Lemma is discussed in Section 4.2.1 . Now, we return to the construction of $\mathcal{A}_{oracle}$. The algorithm is described in Algorithm 4.

Algorithm 4 Algorithm $\mathcal{A}_{oracle}$

Initialization

$\mathcal{J} \leftarrow \emptyset$, $\mathsf{idx} \leftarrow 0$;
$L_2[i] \leftarrow nil, \sigma_2(i) \leftarrow nil, \forall i \in [m]$

Sorted Access

$\mathsf{idx} \leftarrow \mathsf{idx} + 1$;
if $\mathsf{idx} \notin \mathcal{J}$ **then**
  Sample $I_{(\mathsf{idx})}$ uniformly from $[m] \setminus I_{(\mathcal{J})}$;
  Invoke $\mathcal{A}_{ord\text{-}stat}$ to sample $Z_{(\mathsf{idx})}$;
  $\sigma_2(I_{(\mathsf{idx})}) \leftarrow \mathsf{idx}$;
  $L_2[\mathsf{idx}] \leftarrow (I_{(\mathsf{idx})}, Z_{(\mathsf{idx})})$; $\mathcal{J} \leftarrow \mathcal{J} \cup \{\mathsf{idx}\}$.
end if
return $L_2[\mathsf{idx}]$

Random Access (Input: item $i \in [m]$)

if $\sigma_2(i) = nil$ **then**
  Sample j uniformly from $[m] \setminus \mathcal{J}$;
  Invoke $\mathcal{A}_{ord\text{-}stat}$ to sample $Z_{(j)}$;
  $\sigma_2(i) \leftarrow j$;
  $L_2[j] \leftarrow (i, Z_{(j)})$; $\mathcal{J} \leftarrow \mathcal{J} \cup \{j\}$.
end if
return $L_2[\sigma_2(i)]$

Initialization. The algorithm creates an empty array L_2 and an empty inverted index σ_2. Further, it creates a set $\mathcal{J}$, to record the positions of L_2 which are already sampled, and a variable idx, to record the last visited position by sorted access. In practice, L_2 and σ_2 need not to be physically initialized, and can be implemented by hash sets with constant initialization time.

Sorted Access. Indeed, it is trivial to handle the sorted access. We just maintain an index, $\mathsf{idx} \in \mathbb{N}$, of last tuple returned by sorted access. When a new request of sorted access arrives, we increase idx by 1. If $\mathsf{idx} \in \mathcal{J}$, then $\mathcal{A}_{oracle}$ returns $L_2[\mathsf{idx}]$ directly; otherwise, it generates $L_2[\mathsf{idx}]$ before returning it.

Random Access. A random access request comes with a reference to an item $i \in [m]$. We need to identify the tuple $L_2[j] = (I_{(j)}, Z_{(j)})$ s.t., $I_{(j)} = i$. There are two cases: if item i has been encountered previously ($\sigma_2(i) \neq nil$), then $\mathcal{A}_{oracle}$ returns $L_2[\sigma_2(i)]$ directly; otherwise, $\mathcal{A}_{oracle}$ randomly pick an index j from $[m] \setminus \mathcal{J}$, and set $I_{(j)} \leftarrow i$, $\sigma_2(i) \leftarrow j$, and calls $\mathcal{A}_{ord\text{-}stat}$ to generate $Z_{(j)}$. Finally, it returns $L_2[j]$.

4.2.1. Sampling Ordered Noises

In this section, we show how to construct $\mathcal{A}_{ord\text{-}stat}$ and prove Lemma 4.5. Deciding the conditional distribution of the $Z_{(j)}$'s and sampling them directly from such distribution can be non-trivial. As a result, we follow the three-step approach outlined below:

- *Transform $U_{(j)}$ to $Z_{(j)}$*: we show that the sorted sequence $Z_{(1)}, \ldots, Z_{(m)}$ can be transformed from a sequence $U_{(1)}, \ldots, U_{(m)}$ of sorted independent uniform random variables.
- *Distribution of $U_{(j)}$*: to avoid generating the entire sequence of random variables, we study the distribution of $U_{(j)}$, conditioned on a set of $U_{(j')}$ which have already been sampled.
- *Sampling $U_{(j)}$*: we show how to sample an $U_{(j)}$ from such a distribution in $O(\log \log m)$ expected time.

Transform $U_{(j)}$ to $Z_{(j)}$. Since all potential noise distributions ($\mathbb{L}\mathrm{ap}\,(1/\varepsilon)$ or $\mathbb{G}\mathrm{umbel}\,(1/\varepsilon)$) of the Z_i's (the noise random variables, before sorting) have continuous cumulative distribution function, we can sample them indirectly via uniform random variables, based on the *inversion method*.

Fact 4.6 (Inversion Method (Devroye, 1986)). Let F be a continuous cumulative distribution function on $\mathbb{R}$ with inverse F^{-1} defined by

$$F^{-1}(u) \doteq \inf\{x : F(x) = u, 0 < u < 1\}.$$

If U is a uniform $[0, 1]$ random variable, then $F^{-1}(U)$ has distribution function F.

As a result, an ordered sequence of random variables can also be generated via the inversion method.

Fact 4.7 (Gerontidis & Smith 1982). Let $U_1, \ldots, U_m$ be independent uniform random variables on $[0, 1]$, and $U_{(1)}, \ldots, U_{(m)}$ the corresponding sorted sequence in descending order. Let F be a continuous cumulative distribution function shared by the random variables $Z_1, \ldots, Z_m$. Then the sequence $F^{-1}(U_{(1)}), \ldots, F^{-1}(U_{(m)})$ has the same distribution as the sequence $Z_{(1)}, \ldots, Z_{(m)}$.

For completeness, we include a proof for this fact in Appendix B.2. We can thus sample $U_{(j)}$ first and then compute $F^{-1}(U_{(j)})$ to generate $Z_{(j)}$. It remains to study the distribution of $U_{(j)}$, and the algorithm for sampling a random variable efficiently from such distribution.

Distribution of $U_{(j)}$. Recall that $\mathcal{J}$ is the set of indexes which have been previously queried. Denote $U_{(\mathcal{J})} \doteq \{U_{(j')} : j' \in \mathcal{J}\}$ a shorthand for the order random variables that have been sampled. Further, write $u_{(\mathcal{J})} \doteq \{u_{(j')} : j' \in \mathcal{J}\} \in [0, 1]^{|\mathcal{J}|}$ as a set of numbers within $[0, 1]$, indexed by $\mathcal{J}$. Call $u_{(\mathcal{J})}$ a *feasible realization* of $U_{(\mathcal{J})}$, if for each $j, j' \in \mathcal{J}$ s.t. $j < j'$, it holds that $u_{(j)} \geq u_{(j')}$.

Given a new query index $j \in [m] \setminus \mathcal{J}$, we are interested in the conditional probability density, $p_{U_{(j)}|U_{(\mathcal{J})}}(u_{(j)} \mid u_{(\mathcal{J})})$,

of $U_{(j)}$, given the occurrence of a feasible realization $u_{(\mathcal{J})}$ of $U_{(\mathcal{J})}$. For ease of reading, we omit the subscripts of the conditional probability densities, whenever their meaning can be unambiguously determined from their parameters.

Depending on the relative position of j w.r.t. the indexes in $\mathcal{J}$, we consider the following three cases:

- $\mathcal{J}$ is empty. It reduces to study the un-conditional probability density $p(u_{(j)})$ of $U_{(j)}$.
- $\mathcal{J}$ is not empty, and j is greater than the largest index in $\mathcal{J}$; in this case, j has a predecessor (the largest index that is smaller than j), denoted by ℓ, in $\mathcal{J}$.
- $\mathcal{J}$ is not empty, and j is smaller than the largest index in $\mathcal{J}$; in this case, j has both a predecessor, denoted by ℓ, and a successor (the smallest index that is larger than j), denoted by r, in $\mathcal{J}$.

Hereafter, if $\mathcal{J} \neq \varnothing$, we consider only feasible realization $u_{(\mathcal{J})}$ of $U_{(\mathcal{J})}$. The probability densities corresponding to these three cases are given thus.

Theorem 4.8. (1) *If $\mathcal{J}$ is empty, then the density $p(u_{(j)})$ of $U_{(j)}$ is given by: $\forall\, u_{(j)} \in [0,1]$,*

$$p(u_{(j)}) = \tfrac{m!}{(j-1)!(m-j)!} \cdot (1-u_{(j)})^{j-1} (u_{(j)})^{m-j}. \quad (2)$$

(2) *If $\mathcal{J}$ is not empty, and j is greater than the largest index in $\mathcal{J}$, then given $U_{(\ell)} = u_{(\ell)}$, $U_{(j)}$ is independent of all other random variables $U_{(j')}$ for all $j' \in \mathcal{J} \setminus \{\ell\}$, i.e., $p\big(u_{(j)} \mid u_{(\mathcal{J})}\big) = p\big(u_{(j)} \mid u_{(\ell)}\big)$; further, for each $u_{(j)} \in [0, u_{(\ell)}]$,*

$$p\big(u_{(j)} \mid u_{(\ell)}\big) = \tfrac{(m-\ell)!}{(j-\ell-1)!(m-j)!} \cdot \left(\tfrac{u_{(\ell)}-u_{(j)}}{u_{(\ell)}}\right)^{j-\ell-1} \left(\tfrac{u_{(j)}}{u_{(\ell)}}\right)^{m-j} \tfrac{1}{u_{(\ell)}}. \quad (3)$$

(3) *If $\mathcal{J}$ is not empty, and j is smaller than the largest index in $\mathcal{J}$, then given $U_{(\ell)} = u_{(\ell)}$ and $U_{(r)} = u_{(r)}$, $U_{(j)}$ is independent of all other random variables $U_{(j')}$ for all $j' \in \mathcal{J} \setminus \{\ell, r\}$, i.e., $p\big(u_{(j)} \mid u_{(\mathcal{J})}\big) = p\big(u_{(j)} \mid u_{(\ell)}, u_{(r)}\big)$; further, for each $u_{(j)} \in [u_{(r)}, u_{(\ell)}]$,*

$$p\big(u_{(j)} \mid u_{(\ell)}, u_{(r)}\big) = \tfrac{(r-\ell-1)!}{(j-\ell-1)!(r-j-1)!} \cdot \left(\tfrac{u_{(\ell)}-u_{(j)}}{u_{(\ell)}-u_{(r)}}\right)^{j-\ell-1} \left(\tfrac{u_{(j)}-u_{(r)}}{u_{(\ell)}-u_{(r)}}\right)^{r-j-1} \tfrac{1}{u_{(\ell)}-u_{(r)}}. \quad (4)$$

The theorem removes the dependency of $U_{(j)}$ from all but at most two variables in $U_{(\mathcal{J})}$. The detailed proof is non-trivial and can be found in Appendix B.3. Assuming that $U_{(j)}$ depends on at most two variables in $U_{(\mathcal{J})}$, we can provide an informal, but intuitive, explanation of the conditional probability densities. Take the Equation (4) for example. Conditioned on $U_{(r)} = u_{(r)}$ and $U_{(\ell)} = u_{(\ell)}$, $r-\ell-1$ uniform random variables fall into the interval $[u_{(r)}, u_{(\ell)}]$. Of these $r-\ell-1$ random variables, $j-\ell-1$ of them are $\geq u_i$, and $r-j-1$ of them are $< u_i$. The number of possible combinations is given by $\frac{(r-\ell-1)!}{(j-\ell-1)!(r-j-1)!}$. For a fixed combination, the former happens with probability $\left(\frac{u_{(\ell)}-u_{(j)}}{u_{(\ell)}-u_{(r)}}\right)^{j-\ell-1}$, the latter happens with probability $\left(\frac{u_{(j)}-u_{(r)}}{u_{(\ell)}-u_{(r)}}\right)^{r-j-1}$, and the probability density of $U_{(j)} = u_{(j)}$ is $\frac{1}{u_{(\ell)}-u_{(r)}}$.

Sampling $U_{(j)}$. We now discuss how to sample the $U_{(j)}$ efficiently from their conditional distributions. First, note that determining the conditional distributions may need to find the index j's predecessor or successor in $\mathcal{J}$. This can be done by Van Emde Boas tree (van Emde Boas, 1975) in $O(\log\log m)$ time. Next, we show that sampling from such conditional distributions takes $O(1)$ expected time. Specifically, we will sample random variables with *Beta distributions*, then convert them into ones which follow desired conditional distributions.

Definition 4.9 (Beta Distribution (Ross, 2018))**.** The beta distribution $\mathcal{B}\mathrm{eta}\,(\alpha,\beta)$ is a distribution defined on $[0,1]$ whose density is given by

$$p(x) = \tfrac{x^{\alpha-1}(1-x)^{\beta-1}}{\mathrm{B}(\alpha,\beta)}, \qquad \forall x \in [0,1], \quad (5)$$

where $\alpha, \beta > 0$ are *shape parameters*, $\mathrm{B}(\alpha,\beta) \doteq \int_0^1 x^{\alpha-1}(1-x)^{\beta-1}\,dx$ is a normalisation constant.

It is known that, when $\alpha \geq 1, \beta \geq 1$, a random variable $X \sim \mathcal{B}\mathrm{eta}\,(\alpha,\beta)$ can be generated in $O(1)$ expected time (Devroye, 1986; Gentle, 2009).

Theorem 4.10. *Assume that $\ell < j < r \leq m$, and $1 \geq u_{(\ell)} > u_{(r)} \geq 0$. Then*

1. *If $X \sim \mathcal{B}\mathrm{eta}\,(m-j+1, j)$, then the density function of X is the same as Equation (2).*
2. *If $X \sim \mathcal{B}\mathrm{eta}\,(m-j+1, j-\ell)$, then the density function of $Y \doteq u_{(\ell)} \cdot X$ is the same as Equation (3).*
3. *If $X \sim \mathcal{B}\mathrm{eta}\,(r-j, j-\ell)$, then the density function of $Y \doteq (u_{(\ell)} - u_{(r)}) \cdot X$ is the same as Equation (4).*

The proof of the Theorem is included in Appendix B.2.

5. Lower Bounds

In this section, we generate the lower bounds for the problem. Following the setting in Section 2, since $\vec{h}$ is the sum of voting vectors of n clients, we have $||\vec{h}||_\infty \leq n$. It follows that each $S \in \binom{[m]}{k}$ is (n,k)-accurate. All lower bounds in this section hold for algorithms that are $(n - O(1), k)$-accurate, which is just slightly better than the trivial error guarantee.

5.1. Random Access

We first present a lower bound for the random access case.

Theorem 5.1. *Assume that $0 \le \beta < 0.1$. Let $\mathcal{A}$ be an algorithm that has only* random access *to $\vec{h}$, does not return items which it has not seen , and for each input, with probability at least $1-\beta$, returns a solution that is $(n-1,k)$-accurate. Then there exists a family of histograms $\mathcal{H}$, and a distribution μ on $\mathcal{H}$, if $\vec{h}$ is sampled from $\mathcal{H}$ according to distribution μ, it holds that*

$$\mathop{\mathbb{E}}_{\vec{h}}\left[cost\Big(\mathcal{A},\vec{h}\Big)\right] \in \Omega(m). \tag{6}$$

Note that the theorem does not even require $\mathcal{A}$ to be a (ε,δ)-DP algorithm. The proof of the theorem is in Appendix C.1. At a high level, our construction focuses on a family of histograms the values of whose entries are either n or 0. Further, if an $\vec{h}$ is sampled from $\mathcal{H}$, there are roughly $2k$ entries of $\vec{h}$ that have value n, and those entries appear at random positions of $\vec{h}$, so that $\mathcal{A}$ is unlikely to identify more than k of them, before learning the values of $\Omega(m)$ entries.

5.2. Sorted Access

The lower bound for the sorted access case relies on the following lemma.

Lemma 5.2. *Let $\mathcal{A}$ be an (ε,δ)-DP algorithm, which for each input histogram, with probability at least $1-\beta$, returns a solution that is $(n-2,k)$-accurate. Let $S \subseteq [m]$, and $\vec{h}_S$ be a histogram, s.t.*

$$\vec{h}_S[i] \doteq \begin{cases} n-1, & \forall\, i \in S, \\ 0, & \forall\, i \in \bar{S}, \end{cases} \tag{7}$$

where $\bar{S} \doteq [m] \setminus S$. Let S_L be a subset of S sampled uniformly at random from $\binom{S}{|S|/k}$, $S_H \doteq S \setminus S_L$, and $\vec{h}_{S_H,S_L}$ be a histogram neighboring to $\vec{h}$ s.t.

$$\vec{h}_{S_H,S_L}[i] \doteq \begin{cases} n, & \forall\, i \in S_H, \\ n-1, & \forall\, i \in S_L, \\ 0, & \forall\, i \notin S. \end{cases} \tag{8}$$

Then,

$$\Pr_{S_L,\mathcal{A}}\left[\mathcal{A}(\vec{h}_{S_H,S_L}) \cap S_L \neq \emptyset\right] \ge \tfrac{1-\beta-\delta-e^{-1}}{e^{\varepsilon}}, \tag{9}$$

where the randomness is first over the choice of S_L then over the output of $\mathcal{A}$.

The formal proof of the lemma is included in Appendix C.2. Note that, for each $i \in S_L$, $\vec{h}_{S_H,S_L}[i]$ is among the $|S|/k$ smallest entries of the $|S|$ largest entries in $\vec{h}_{S_H,S_L}$. The lemma states that the probability that output of $\mathcal{A}$ contains some item $i \in S_L$ is not too "small". Informally, for each subset $S_k \in \binom{S}{k}$, when S_L is sampled uniformly from $\binom{S}{|S|/k}$, then the probability that $S_k \cap S_L \neq \emptyset$ is not too "small" (observe that $\mathbb{E}\left[|S_k \cap S_L|\right] = k \cdot \frac{|S|/k}{|S|} = 1$). Further, if $\mathcal{A}$'s output is $(n-2,k)$ accurate, then it must belong to $\binom{S}{k}$. So the probability that $\mathcal{A}$'s output has non-empty intersection with S_L should not be significantly smaller than the probability that $\mathcal{A}$'s output is $(n-2,k)$ accurate.

Theorem 5.3. *Let ε,δ,β be non-negative parameters, s.t., $\varepsilon \in O(1), \delta+\beta \le 0.6$. Let $\mathcal{A}$ be an (ε,δ)-DP algorithm that has only* sorted access*, does not return items which it has not seen, and for each input histogram, with probability at least $1-\beta$, returns a solution that is $(n-2,k)$-accurate. Then there exists a family of histograms $\mathcal{H}$ so that, if $\vec{h}$ is sampled uniformly at random from $\mathcal{H}$, it holds that*

$$\mathop{\mathbb{E}}_{\vec{h}}\left[cost\Big(\mathcal{A},\vec{h}\Big)\right] \in \Omega(m). \tag{10}$$

Proof. Let $S = [m/2]$, S_L be sampled uniformly at random from $\binom{S}{|S|/k}$, and $\vec{h}_{S_H,S_L}$ be a histogram built as outlined in Equation (8). Let $\mathcal{H}$ be the collection of all possible outcomes of $\vec{h}_{S_H,S_L}$. Then by Lemma 5.2,

$$\Pr_{S_L,\mathcal{A}}\left[\mathcal{A}(\vec{h}_{S_H,S_L}) \cap S_L \neq \emptyset\right] \ge \tfrac{1-\beta-\delta-e^{-1}}{e^{\varepsilon}}.$$

But for each $i \in S_L$, $\vec{h}_{S_H,S_L}[i]$ is among the top $(|S|-|S|/k+1)^{(th)}$ to $|S|^{(th)}$ largest numbers in $\vec{h}_{S_H,S_L}$. Since $\mathcal{A}$ has only sorted access to $\vec{h}_{S_H,S_L}$ and does not return an item which it has not seen, if it returns an item in S_L, it needs to invoke at least $|S|-|S|/k \in \Omega(m)$ sorted accesses. It follows that the expected access cost of $\mathcal{A}$ is at least $\frac{1-\beta-\delta-e^{-1}}{e^{\varepsilon}} \cdot \Omega(m)$. Inequality (10) follows from the assumption that $\varepsilon \in O(1)$, and $\beta+\delta \le 0.6$. □

5.3. Random and Sorted Access

In this section, we present a lower bound for algorithms that can retrieve data via both random access and sorted access.

Theorem 5.4. *Let ε,δ,β be non-negative parameters, s.t., $\varepsilon \le 1, \delta+\beta \le 0.05$. Let $\mathcal{A}$ be an algorithm that has both* sorted access *and* random access *to $\vec{h}$, does not return items which it has not seen, and for each input, with probability at least $1-\beta$, returns a solution that is $(n-2,k)$-accurate. Then there exists a family of histograms $\mathcal{H}$, if $\vec{h}$ is sampled uniformly at random from $\mathcal{H}$, it holds that*

$$\mathop{\mathbb{E}}_{\vec{h}}\left[cost\Big(\mathcal{A},\vec{h}\Big)\right] \in \Omega\Big(\sqrt{mk}\Big). \tag{11}$$

Proof. Let $\mathcal{H}$ be the collection of all possible $\vec{h}_{S_H,S_L}$ generated as follows:

- First, we sample an S from $\binom{[m]}{\tau}$ uniformly at random, where $\tau \doteq \sqrt{mk}$.
- Then we sample an S_L from $\binom{S}{|S|/k}$ uniformly at random, and construct a histogram $\vec{h}_{S_H,S_L}$ as described by Equation (8).

Since Lemma 5.2 holds for each $S \subseteq [m]$, we have

$$\begin{aligned}\Pr_{S,S_L,\mathcal{A}}\left[\mathcal{A}\Big(\vec{h}_{S_H,S_L}\Big) \cap S_L \neq \emptyset\right] &\ge \tfrac{1}{e^{\varepsilon}}\big(1-\beta-\delta-\tfrac{1}{e}\big) \\ &\overset{(a)}{\ge} e^{-1}\big(1-0.05-e^{-1}\big) \ge 0.21,\end{aligned}$$

where the randomness is first over the choice of S, then over the choice of S_L, and finally over the output of $\mathcal{A}$, and inequality (a) follows from the assumption that $\varepsilon \leq 1$ and $\beta + \delta \leq 0.05$. In what follows, we omit the subscripts $S, S_L, \mathcal{A}$ from the probability notations, when the source of randomness is clear from the context.

Consider the event $\mathcal{E}$: $\mathcal{A}$ accesses some item $i \in S_L$. As $\mathcal{A}$ does not return an item which it has not seen, the event $\mathcal{E}$ is a necessary condition for $\mathcal{A}\left(\vec{h}_{S_H,S_L}\right) \cap S_L \neq \emptyset$. Hence,

$$\Pr[\mathcal{E}] \geq \Pr\left[\mathcal{A}\left(\vec{h}_{S_H,S_L}\right) \cap S_L \neq \emptyset\right].$$

Let $\eta \doteq \tau/20$. We decompose $\mathcal{E}$ into two mutually exclusive events: $\mathcal{E}_1$: $\mathcal{A}$ accesses some item $i \in S_L$ for the first time within η access operations; $\mathcal{E}_2$: $\mathcal{A}$ accesses some item $i \in S_L$ for the first time after η access operations. Then $\Pr[\mathcal{E}] = \Pr[\mathcal{E}_1] + \Pr[\mathcal{E}_2]$.

Lemma 5.5. *The probability that $\mathcal{A}$ accesses some item $i \in S_L$ for the first time within η access operations, denoted by $\Pr[\mathcal{E}_1]$, is upper bounded by $\Pr[\mathcal{E}_1] \leq 0.19$.*

The proof of Lemma 5.5 is omitted here, and is included in Appendix C.3. Intuitively, the lemma holds since: 1) $\mathcal{A}$ can not access some $i \in S_L$ within η sorted accesses; 2) because of the way it is generated, S_L is a random subset from $[m]$ of size $\sqrt{mk}/k$, hence it is also unlikely for $\mathcal{A}$ to come across some $i \in S_L$ with at most $\eta = \sqrt{mk}/20$ random access. To conclude the justification of (11), and hence prove the Theorem, we apply Lemma 5.5.

$$\begin{aligned}\Pr[\mathcal{E}_2] &= \Pr[\mathcal{E}] - \Pr[\mathcal{E}_1] \\ &\geq \Pr\left[\mathcal{A}\left(\vec{h}_{S_H,S_L}\right) \cap S_L \neq \emptyset\right] - \Pr[\mathcal{E}_1] \\ &\geq 0.21 - 0.19 \in \Omega(1).\end{aligned}$$

But when $\mathcal{E}_2$ happens, the access cost is $\Omega(\eta)$. Therefore, the expected access cost of $\mathcal{A}$ is lower bounded by $\Omega(\eta) = \Omega(\sqrt{mk})$. □

6. Related Work

Private Selection. The private top-1 selection problem is a special case of the private top-k problem. The latter has been studied extensively, e.g., the exponential mechanism (McSherry & Talwar, 2007), report noisy max (Dwork & Roth, 2014), permute-and-flip (McKenna & Sheldon, 2020; Ding et al., 2021). Of interest is the permute-and-flip mechanism: when the largest score of the items is known a prior , the mechanism can potentially terminate without iterating over all m items. However, in this scenario, an asymptotic upper bound for the number of items evaluated remains unresolved.

Private Top-k Mechanisms. Bhaskar et al. (2010) were the first to apply the "peeling exponential mechanism", which iteratively invoked the exponential mechanism to select the item with highest score, then remove it. They also proposed an oneshot Laplace mechanism for private top-k selection. Bhaskar et al. (2010) analyzed the pure differential privacy guarantees of both algorithms. Subsequently, Durfee & Rogers (2019) showed that the peeling exponential mechanism has an equivalent oneshot implementation (i.e., Algorithm 2 with Gumbel noise), and studied its approximate privacy guarantee. Qiao et al. (2021) provided the approximate privacy guarantee for the oneshot Laplace mechanism, without the help of the composition theorem.

Both Bhaskar et al. (2010) and Durfee & Rogers (2019) have proposed private algorithms which estimate top-k based on the true top $\bar{k}$ items for some $\bar{k} \geq k$. Given an integer k, both algorithm may need to set $\bar{k} = m$, in order to return k items.

Accuracy Lower Bound. Bafna & Ullman (2017) and Steinke & Ullman (2017) show that, for approximate private algorithms, the error guarantees of existing algorithms (McSherry & Talwar, 2007; Dwork & Roth, 2014; Durfee & Rogers, 2019; Qiao et al., 2021) are essentially optimal.

7. Conclusions and Future Directions

In this paper, we systematically advance our understanding of the access cost of private top-k selection algorithm. We introduce the first algorithm with sublinear access cost, and provide lower bounds for three access models, showing that supporting both sorted access and random access is the key to breaking the linear access cost barrier, and that the access cost of our algorithm is optimal.

We believe our work is a first step towards a comprehensive study of building a differentially private top-k algorithm on top of existing data analytics systems. Our focus in this work is primarily on advancing theoretical understanding of the problem, assuming that sorted access and random access operations have the same cost. Interesting future directions include conducting empirical evaluations, and investigating scenarios where the costs of these two operations differ.

Acknowledgements

We thank the anonymous reviewers for their constructive feedback, which has helped us to improve our manuscript. In particular, we acknowledge the suggestion that our algorithm can be applied to the exponential mechanism, resulting in an $O(\sqrt{m})$ expected access cost.

Hao Wu is supported by an Australian Government Research Training Program (RTP) Scholarship.

References

Bafna, M. and Ullman, J. R. The price of selection in differential privacy. In Kale, S. and Shamir, O. (eds.), *Proceedings of the 30th Conference on Learning Theory, COLT 2017, Amsterdam, The Netherlands, 7-10 July 2017*, volume 65 of *Proceedings of Machine Learning Research*, pp. 151–168. PMLR, 2017.

Bhaskar, R., Laxman, S., Smith, A., and Thakurta, A. Discovering frequent patterns in sensitive data. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '10, pp. 503–512, New York, NY, USA, 2010. Association for Computing Machinery. ISBN 9781450300551.

Blitzstein, J. and Hwang, J. *Introduction to Probability*. 07 2014. ISBN 9780429102103.

Brassard, G. and Kannan, S. The generation of random permutations on the fly. *Inf. Process. Lett.*, 28(4):207–212, 1988.

Cardoso, A. R. and Rogers, R. Differentially private histograms under continual observation: Streaming selection into the unknown. In Camps-Valls, G., Ruiz, F. J. R., and Valera, I. (eds.), *International Conference on Artificial Intelligence and Statistics, AISTATS 2022, 28-30 March 2022, Virtual Event*, volume 151 of *Proceedings of Machine Learning Research*, pp. 2397–2419. PMLR, 2022.

Devroye, L. *Non-Uniform Random Variate Generation*. Springer, 1986. ISBN 978-1-4613-8645-2.

Ding, Z., Kifer, D., E., S. M. S. N., Steinke, T., Wang, Y., Xiao, Y., and Zhang, D. The permute-and-flip mechanism is identical to report-noisy-max with exponential noise. *CoRR*, abs/2105.07260, 2021.

Durfee, D. and Rogers, R. M. Practical differentially private top-k selection with pay-what-you-get composition. In Wallach, H. M., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E. B., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 3527–3537, 2019.

Dwork, C. and Roth, A. The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci.*, 9(3-4):211–407, 2014.

Fagin, R. Combining fuzzy information from multiple systems. *J. Comput. Syst. Sci.*, 58(1):83–99, 1999.

Fagin, R., Lotem, A., and Naor, M. Optimal aggregation algorithms for middleware. *J. Comput. Syst. Sci.*, 66(4): 614–656, 2003.

Gentle, J. E. *Computational Statistics*. Statistics and Computing. Springer, New York, NY, December 2009.

Gerontidis, I. and Smith, R. L. Monte carlo generation of order statistics from general distributions. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 31 (3):238–243, 1982. ISSN 00359254, 14679876.

Guntuboyina, A. Lecture Notes for 201A Fall, 2019. https://www.stat.berkeley.edu/~aditya/resources/FullLectureNotes201AFall2019.pdf (Accessed on Oct. 21, 2022).

Hoeffding, W. *Probability Inequalities for sums of Bounded Random Variables*, pp. 409–426. Springer New York, New York, NY, 1994. ISBN 978-1-4612-0865-5.

Ilyas, I. F., Beskales, G., and Soliman, M. A. A survey of top-*k* query processing techniques in relational database systems. *ACM Comput. Surv.*, 40(4):11:1–11:58, 2008.

Lurie, D. and Hartley, H. O. Machine-generation of order statistics for monte carlo computations. *The American Statistician*, 26:26–27, 1972.

Matousek, J. and Nesetril, J. *Invitation to Discrete Mathematics (2. ed.)*. Oxford University Press, 2009. ISBN 978-0-19-857042-4.

McKenna, R. and Sheldon, D. Permute-and-flip: A new mechanism for differentially private selection. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

McSherry, F. and Talwar, K. Mechanism design via differential privacy. In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2007), October 20-23, 2007, Providence, RI, USA, Proceedings*, pp. 94–103. IEEE Computer Society, 2007.

Mitzenmacher, M. and Upfal, E. *Probability and computing: Randomization and probabilistic techniques in algorithms and data analysis*. Cambridge university press, 2017.

Qiao, G., Su, W. J., and Zhang, L. Oneshot differentially private top-k selection. In Meila, M. and Zhang, T. (eds.), *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pp. 8672–8681. PMLR, 2021.

Robbins, H. A remark on stirling's formula. *The American Mathematical Monthly*, 62(1):26–29, 1955. ISSN 00029890, 19300972.

Ross, S. *A First Course in Probability*. Pearson, Upper Saddle River, NJ, 10 edition, November 2018.

Steinke, T. and Ullman, J. R. Tight lower bounds for differentially private selection. In Umans, C. (ed.), *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pp. 552–563. IEEE Computer Society, 2017.

Vadhan, S. P. The complexity of differential privacy. In Lindell, Y. (ed.), *Tutorials on the Foundations of Cryptography*, pp. 347–450. Springer International Publishing, 2017.

van Emde Boas, P. Preserving order in a forest in less than logarithmic time. In *16th Annual Symposium on Foundations of Computer Science, Berkeley, California, USA, October 13-15, 1975*, pp. 75–84. IEEE Computer Society, 1975.

Zhang, H., Mironov, I., and Hejazinia, M. Wide network learning with differential privacy. *CoRR*, abs/2103.01294, 2021.

A. Probability Inequalities

Fact A.1 (Chernoff Bound, (Mitzenmacher & Upfal 2017, Theorem 4.4 and 4.5)). Let $X_1, X_2, \ldots, X_n$ be independent Poisson trials such that, for $i \in [m]$, $\Pr[X_i = 1] = p_i$, where $0 < p_i < 1$. Then, for $X = \sum_{i=1}^{n} X_i$, $\mu = \mathbb{E}[X]$,

$$\Pr[X \geq (1+\lambda)\mu] \leq \left(\frac{e^{\lambda}}{(1+\lambda)^{1+\lambda}}\right)^{\mu}, \qquad \forall \lambda > 0, \tag{12}$$

$$\Pr[X \geq (1+\lambda)\mu] \leq \left(\frac{e^{\lambda}}{(1+\lambda)^{1+\lambda}}\right)^{\mu} \leq e^{-\lambda^2\mu/3}, \qquad \forall \lambda \in (0,1], \tag{13}$$

$$\Pr[X \leq (1-\lambda)\mu] \leq \left(\frac{e^{-\lambda}}{(1-\lambda)^{1-\lambda}}\right)^{\mu} \leq e^{-\lambda^2\mu/2}, \qquad \forall \lambda \in (0,1). \tag{14}$$

The known concentration inequalities for sampling with replacement can be transferred to the case of sampling without replacement, based on a notable reduction technique.

Fact A.2 ((Hoeffding, 1994)). Let $\mathcal{X} = (x_1, \ldots, x_N)$ be a finite population of N real points, $Y_1, \ldots, Y_n$ denote a random sample without replacement from $\mathcal{X}$ and $X_1, \ldots, X_n$ denote a random sample with replacement from $\mathcal{X}$. If $f : \mathbb{R} \to \mathbb{R}$ is continuous and convex, then

$$\mathbb{E}\left[f\left(\sum_{i=1}^{n} Y_i\right)\right] \leq \mathbb{E}\left[f\left(\sum_{i=1}^{n} X_i\right)\right].$$

In particular, the lower bound presented in Fact A.1 can be converted into the following one, by combining its proof in (Mitzenmacher & Upfal, 2017) and Fact A.2.

Corollary A.3 (Chernoff bound)**.** *Let $\mathcal{X} = (x_1, \ldots, x_N) \in \{0,1\}^N$ be a finite population of N binary points and $Y_1, \ldots, Y_n$ be a random sample drawn without replacement from the population. Then, for $Y = \sum_{i=1}^{n} Y_i$,*

$$\Pr\left[\sum_{i=1}^{n} Y_i \leq (1-\lambda)\mu\right] \leq \left(\frac{e^{-\lambda}}{(1-\lambda)^{1-\lambda}}\right)^{\mu} \leq e^{-\lambda^2\mu/2}, \qquad \forall \lambda \in [0,1). \tag{15}$$

where $\mu = np$ is the expectation of $\sum_{i=1}^{n} Y_i$, and $p \doteq \frac{1}{N}\sum_{i=1}^{N} x_i$ is the mean of $\mathcal{X}$.

Note that, in Corollary A.3, we also extend the range of λ from $\{0,1\}$, to $[0,1)$. When $\lambda = 0$, $\left(\frac{e^{-\lambda}}{(1-\lambda)^{1-\lambda}}\right)^{\mu} = e^{-\lambda^2\mu/2} = 1$, and Inequality (15) holds trivially.

Fact A.4 ((Blitzstein & Hwang, 2014)). Let $\Gamma(a) \doteq \int_0^{\infty} x^{a-1}e^{-x}\,dx$, $\forall a > 0$ be the *Gamma function*. It holds that $\Gamma(a+1) = a\Gamma(a)$, and $\Gamma(\frac{1}{2}) = \sqrt{\pi}$. Then for each $k \in \mathbb{N}^+$, we have $\Gamma(k) = (k-1)!$ and

$$\Gamma\left(\frac{1}{2}+k\right) = \left(k-\frac{1}{2}\right)\left(k-\frac{3}{2}\right)\cdots\frac{1}{2}\Gamma\left(\frac{1}{2}\right) = \frac{2k-1}{2}\frac{2k-3}{2}\cdots\frac{1}{2}\sqrt{\pi} = \frac{(2k)!}{4^k k!}\sqrt{\pi}. \tag{16}$$

Fact A.5 ((Ross, 2018)). Given *shape parameters* $\alpha, \beta > 0$, define *beta function* $\mathrm{B}(\alpha,\beta) \doteq \int_0^1 x^{\alpha-1}(1-x)^{\beta-1}\,dx$. Then $\mathrm{B}(\alpha,\beta) = \frac{\Gamma(\alpha)\Gamma(\beta)}{\Gamma(\alpha+\beta)}$, In particular,

$$\mathrm{B}(\alpha,\beta) = \frac{(\alpha-1)!(\beta-1)!}{(\alpha+\beta-1)!}, \qquad \forall \alpha, \beta \in \mathbb{N}^+. \tag{17}$$

The factorials can be estimated as follows.

Fact A.6 (Stirling's Approximation (Robbins, 1955; Matousek & Nesetril, 2009)). For $k = 1, 2, \ldots$

$$\sqrt{2\pi k}\left(\frac{k}{e}\right)^k \exp\left(\frac{1}{12k+1}\right) \leq k! \leq \sqrt{2\pi k}\left(\frac{k}{e}\right)^k \exp\left(\frac{1}{12k}\right). \tag{18}$$

B. Proofs for Section 4

B.1. Proofs for Section 4.1

Proof for Lemma 4.1. For each $i \in [m]$, let $p_{Z_i}(\cdot)$ be the density function of random variable Z_i. Since the Z_i's are i.i.d. random variables, they share the same density function, i.e., $p_{Z_1}(\cdot) = \cdots = p_{Z_m}(\cdot)$.

Let S_m be the collection of all possible permutations over $[m]$, and s refer to a permutation in S_m . Further, let $z_{(1)} \geq \ldots \geq z_{(m)}$ denote a possible realization of $Z_{(1)}, \ldots, Z_{(m)}$, and $z_{(1:m)} \doteq (z_{(1)}, \ldots, z_{(m)})$. We write $I_{(1:m)} = s$, if $I_{(j)} = s(j), \forall j \in [m]$, and $Z_{(1:m)} = z_{(1:m)}$, when $Z_{(j)} = z_{(j)}, \forall j \in [m]$.

Let $p_{I_{(1:m)}, Z_{(1:m)}}(s, z_{(1:m)})$ be the probability density when $I_{(1:m)} = s$ and $Z_{(1:m)} = z_{(1:m)}$. The probability density, $p_{Z_{(1:m)}}(z_{(1:m)})$ of $Z_{(1:m)} = z_{(1:m)}$ is given by

$$\begin{aligned} p_{Z_{(1:m)}}(z_{(1:m)}) &= \sum_{s \in S_m} p_{I_{(1:m)}, Z_{(1:m)}}(s, z_{(1:m)}) \\ &= \sum_{s \in S_m} \prod_{j \in [m]} p_{Z_{s(j)}}(z_{(j)}) \\ &= \sum_{s \in S_m} \prod_{j \in [m]} p_{Z_1}(z_{(j)}) \\ &= m! \prod_{j \in [m]} p_{Z_1}(z_{(j)}). \end{aligned}$$

Hence, for a given $s \in S_m$, the probability density of $I_{(1:m)} = s$, conditioned on $Z_{(1:m)} = z_{(1:m)}$, is given by

$$p_{I_{(1:m)} | Z_{(1:m)}}(s \mid z_{(1:m)}) = \frac{p_{I_{(1:m)}, Z_{(1:m)}}(s, z_{(1:m)})}{p_{Z_{(1:m)}}(z_{(1:m)})} = \frac{\prod_{j \in [m]} p_{Z_1}(z_{(j)})}{m! \prod_{j \in [m]} p_{Z_1}(z_{(j)})} = \frac{1}{m!},$$

which is independent of the values of the $Z_{(1:m)}$. Finally,

$$p_{I_{(1:m)}}(s) = \mathop{\mathbb{E}}_{Z_{(1:m)}} \left[p_{I_{(1:m)} | Z_{(1:m)}}(s \mid Z_{(1:m)}) \right] = \frac{1}{m!}.$$

□

B.1.1. Proof of Theorem 4.2

Before the proof of Theorem 4.2, we present two supporting lemmas.

Lemma B.1. *Let cost* $(\mathcal{A}_{PrivTA}, L_1)$ *be the access cost of Algorithm 3. Then*

$$\Pr\left[cost\left(\mathcal{A}_{PrivTA}, L_1\right) \geq 2 \cdot r\right] \leq \frac{e^k}{k^k} \cdot \frac{\left(r^2/m\right)^k}{e^{r^2/m}}, \qquad \forall r \in \mathbb{N},\ s.t.,\ r \geq \sqrt{mk}. \tag{19}$$

Proof for Lemma B.1. Let $\mathcal{S}_r \doteq \left\{i_{(1)}, \ldots, i_{(r)}\right\}$ be the top-r items with highest scores in $\vec{h}$. Also consider the items in the first r tuples in the array L_2, denoted by $I_{(1:r)} = \left\{I_{(1)}, \ldots, I_{(r)}\right\}$.

Consider the following event:

$$\mathcal{E} \doteq \left|\mathcal{S}_r \cap I_{(1:r)}\right| > k\,.$$

We claim that

1. When event $\mathcal{E}$ happens, $cost\left(\mathcal{A}_{PrivTA}, L_1\right) < 2r$.
2. The complement of $\mathcal{E}$, denoted by $\bar{\mathcal{E}}$, happens with probability $\Pr\left[\bar{\mathcal{E}}\right] \leq \frac{e^k}{k^k} \cdot \frac{\left(r^2/m\right)^k}{e^{r^2/m}}$.

Combing both claims, we get

$$\Pr\left[cost\left(\mathcal{A}_{PrivTA}, L_1\right) \geq 2r\right] \leq \Pr\left[\bar{\mathcal{E}}\right] \leq \frac{e^k}{k^k} \cdot \frac{\left(r^2/m\right)^k}{e^{r^2/m}}.$$

Claim One. It suffices to show that, when event $\mathcal{E}$ happens, Algorithm 1 stops before r rounds , where each round involves executing lines 4 to 10 in Algorithm 1. In such a case, the number of random accesses incurred (to $\vec{h}$) is less than r. Therefore, the total number of access cost to $\vec{h}$ is less than $2r$.

Assume that Algorithm 1 runs for r rounds. Since $i_{(r)}$ and $I_{(r)}$ are the last encountered items, the corresponding threshold (Algorithm 1, Line 9) is given by $\tau_r \doteq f(\vec{h}[i_{(r)}], Z_{I_{(r)}})$.

But for each $i \in \left\{i_{(1)}, \ldots, i_{(r)}\right\} \cap I_{(1:r)}$, it holds that $\vec{h}[i] \geq \vec{h}[i_{(r)}]$, and $Z_i \geq Z_{I_{(r)}}$. Since f is monotone, the score of item i, $f(\vec{h}[i], Z_i)$, is at least τ_r. Since event $\mathcal{E}$ happens, at least k items encountered by the algorithm have score at least τ. Therefore, the stopping condition of Algorithm 1 should be satisfied.

Claim Two. Via Lemma 4.1, the $I_{(1)}, \ldots, I_{(m)}$ distribute uniformly over all permutations of $[m]$. Therefore, the set $I_{(1:r)}$ can be viewed as a uniform sample (without replacement) of r elements from $[m]$. Hence, for each $j \in [r]$, the probability that $i_{(j)}$ belongs to $I_{(1:r)}$ is given by

$$\Pr\left[i_{(j)} \in I_{(1:r)}\right] = r/m\,.$$

Let $\mathbb{1}_{\left[i_{(j)} \in I_{(1:r)}\right]}$ be the indicator for the event that $i_{(j)} \in I_{(1:r)}$, and $Y \doteq \sum_{j\in[r]} \mathbb{1}_{\left[i_{(j)} \in I_{(1:r)}\right]}$. The event $\bar{\mathcal{E}}$ is equivalent to $Y \leq k$. Observe that

$$\Pr\left[\mathbb{1}_{\left[i_{(j)} \in I_{(1:r)}\right]} = 1\right] = \Pr\left[i_{(j)} \in I_{(1:r)}\right] = r/m, \tag{20}$$

$$\mu \doteq \mathbb{E}\left[Y\right] = \sum_{j\in[r]} \mathbb{E}\left[\mathbb{1}_{\left[i_{(j)} \in I_{(1:r)}\right]}\right] = r^2/m \geq k. \tag{21}$$

Since $k/\mu \in (0, 1]$, $\lambda \doteq 1 - k/\mu \in [0, 1)$. By the Chernoff bound for sampling without replacement (Fact A.3), we have

$$\Pr\left[Y \leq k\right] = \Pr\left[Y \leq (1-\lambda)\cdot\mu\right] \leq \left(\frac{e^{-\lambda}}{(1-\lambda)^{1-\lambda}}\right)^{\mu} = \left(\frac{e^{k/\mu-1}}{(k/\mu)^{k/\mu}}\right)^{\mu} = \frac{e^{k-\mu}}{(k/\mu)^k} = \frac{e^k\mu^k}{k^k e^{\mu}} = \frac{e^k}{k^k} \cdot \frac{\left(r^2/m\right)^k}{e^{r^2/m}}.$$

□

Lemma B.2. *Let $cost\left(\mathcal{A}_{PrivTA}, L_1\right)$ be the access cost of Algorithm 3. Then*

$$\Pr\left[cost\left(\mathcal{A}_{PrivTA}, L_1\right) \geq 2\cdot r\right] \leq \frac{e^k}{k^k} \cdot \frac{\left(r^2/m\right)^k}{e^{r^2/m}}, \qquad \forall r \in \mathbb{R},\ s.t.,\ r \geq \sqrt{mk}. \tag{22}$$

Proof for Lemma B.2. Since Algorithm 3 executes the threshold algorithm (Algorithm 1) with two sorted arrays L_1 and L_2, the access cost of Algorithm 3 on L_1, denoted by $cost\left(\mathcal{A}_{PrivTA}, L_1\right)$, is an even integer due to the way the threshold algorithm operates. Therefore, the event $cost\left(\mathcal{A}_{PrivTA}, L_1\right) \geq 2\cdot r$ is equivalent to the event $cost\left(\mathcal{A}_{PrivTA}, L_1\right) \geq \lceil 2\cdot r\rceil_{even}$, where $\lceil 2\cdot r\rceil_{even}$ denotes the smallest even integer that is at least $2\cdot r$.

Define $r_\star \doteq \frac{1}{2}\cdot\lceil 2\cdot r\rceil_{even} \in \mathbb{N}$. Clearly it holds that $r_\star \geq r$. Via Lemma B.1, for each $r \in \mathbb{R}$, $s.t.$, $r \geq \sqrt{mk}$,

$$\begin{aligned}\Pr\left[cost\left(\mathcal{A}_{PrivTA}, L_1\right) \geq 2\cdot r\right] &= \Pr\left[cost\left(\mathcal{A}_{PrivTA}, L_1\right) \geq \lceil 2\cdot r\rceil_{even}\right] \\ &= \Pr\left[cost\left(\mathcal{A}_{PrivTA}, L_1\right) \geq 2\cdot r_\star\right] \leq \frac{e^k}{k^k} \cdot \frac{\left(r_\star^2/m\right)^k}{e^{r_\star^2/m}}.\end{aligned}$$

Consider the function $y \doteq x^k/e^x$. As $y' = \frac{kx^{k-1}e^x - x^k e^x}{e^{2x}}$, y is decreasing when $x \geq k$. Noting that $r_\star^2/m \geq r^2/m$, we have

$$\Pr\left[cost\left(\mathcal{A}_{PrivTA}, L_1\right) \geq 2\cdot r\right] \leq \frac{e^k}{k^k} \cdot \frac{\left(r_\star^2/m\right)^k}{e^{r_\star^2/m}} \leq \frac{e^k}{k^k} \cdot \frac{\left(r^2/m\right)^k}{e^{r^2/m}}, \qquad \forall r \in \mathbb{R},\ s.t.,\ r \geq \sqrt{mk}.$$

□

We are now ready to prove Theorem 4.2.

Proof of Theorem 4.2. First, we can rewrite

$$\mathbb{E}\left[cost\left(\mathcal{A}_{PrivTA}, L_1\right)\right] = \int_0^\infty \Pr\left[cost\left(\mathcal{A}_{PrivTA}, L_1\right) \geq s\right]\, ds = 2 \cdot \int_0^\infty \Pr\left[cost\left(\mathcal{A}_{PrivTA}, L_1\right) \geq 2 \cdot r\right]\, dr,$$

where the last inequality follows from a change of variable of $r \doteq s/2$.

Decomposing the integral further, we have

$$\begin{aligned}\int_0^\infty \Pr\left[cost\left(\mathcal{A}_{PrivTA}, L_1\right) \geq 2r\right]\, dr &= \int_0^{\sqrt{mk}} \Pr\left[cost\left(\mathcal{A}_{PrivTA}, L_1\right) \geq 2r\right]\, dr + \int_{\sqrt{mk}}^\infty \Pr\left[cost\left(\mathcal{A}_{PrivTA}, L_1\right) \geq 2r\right]\, dr \\ &\leq \sqrt{mk} + \int_{\sqrt{mk}}^\infty \Pr\left[cost\left(\mathcal{A}_{PrivTA}, L_1\right) \geq 2r\right]\, dr.\end{aligned}$$

Via Lemma B.2, we can bound the last integral by

$$\int_{\sqrt{mk}}^\infty \Pr\left[cost\left(\mathcal{A}_{PrivTA}, L_1\right) \geq 2r\right]\, dr \leq \int_{\sqrt{mk}}^\infty \frac{e^k}{k^k} \cdot \frac{\left(r^2/m\right)^k}{e^{r^2/m}}\, dr = \int_k^\infty \frac{e^k}{k^k} \cdot t^k e^{-k} \sqrt{m} \frac{dt}{2\sqrt{t}},$$

where the last inequality follows from a change of variable of $t \doteq r^2/m$. Via the definition and property of Gamma function (Fact A.4),

$$\begin{aligned}\int_k^\infty \frac{e^k}{k^k} \cdot t^k e^{-k} \sqrt{m} \frac{dt}{2\sqrt{t}} &\leq \frac{\sqrt{m}}{2} \cdot \frac{e^k}{k^k} \cdot \int_0^\infty t^{k-1/2} e^{-k}\, dt \\ &= \frac{\sqrt{m}}{2} \cdot \frac{e^k}{k^k} \cdot \Gamma(k+1/2) = \frac{\sqrt{m}}{2} \cdot \frac{e^k}{k^k} \cdot \frac{(2k)!}{4^k k!}.\end{aligned}$$

Finally, by Stirling's approximation (Fact A.6),

$$\frac{e^k}{k^k} \cdot \frac{(2k)!}{4^k k!} \leq \frac{e^k}{k^k} \cdot \frac{\sqrt{2\pi \cdot 2k}\left(\frac{2k}{e}\right)^{2k} \exp\left(\frac{1}{12 \cdot 2k}\right)}{4^k \sqrt{2\pi \cdot k}\left(\frac{k}{e}\right)^k \exp\left(\frac{1}{12 \cdot k + 1}\right)} \leq \sqrt{2}.$$

Combing the previous inequalities, we show that

$$\mathbb{E}\left[cost\left(\mathcal{A}_{PrivTA}, L_1\right)\right] \leq 2 \cdot \left(\sqrt{mk} + \frac{\sqrt{m}}{\sqrt{2}}\right),$$

which proves the theorem. □

B.2. Proofs for Section 4.2

Proof of Theorem 4.4. This is a directly consequence of the facts that

- Based on a technique by Brassard & Kannan (1988) for sampling a random perturbation on the fly, and on the discussions in Section 4.2, it holds that each $I_{(j)}, j \in [m]$, and each value of inverted index $\sigma_2(i), i \in [m]$ can be sampled on the fly in $O(1)$ times.
- Based on Lemma 4.5, each $Z_{(j)}, j \in [m]$ can be sampled on the fly with $O(\log \log m)$ expected time.

□

Proof of Fact 4.7. Recall that F is the cumulative distribution function of $Z_1, \ldots, Z_m$, and $U_1, \ldots, U_m$ are independent uniform random variables on $[0, 1]$. We compare the two post-processing procedures:

- By Fact 4.6, we can obtain $Z_1, \ldots, Z_m$ by computing $F^{-1}(U_1), \ldots, F^{-1}(U_m)$; then we obtain $Z_{(1)}, \ldots, Z_{(m)}$ by sorting this sequence in descending order.
- Alternatively, we first sort $U_1, \ldots, U_m$ in descending order, to obtain a sequence $U_{(1)}, \ldots, U_{(m)}$; then we compute $F^{-1}(U_{(1)}), \ldots, F^{-1}(U_{(m)})$.

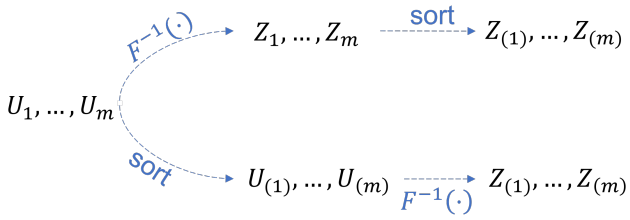

Figure 1. A pictorial comparison of the two procedures.

We claim that the two procedures output the same sequence.

First, observe that the following two multisets are the same $\{F^{-1}(U_1), \ldots, F^{-1}(U_m)\}$ and $\{F^{-1}(U_{(1)}), \ldots, F^{-1}(U_{(m)})\}$. By construction, $Z_{(1)}, \ldots, Z_{(m)}$ is a sorted sequence of the multiset. To prove the claim, it suffices to show that $F^{-1}(U_{(1)}), \ldots, F^{-1}(U_{(m)})$ is also a sorted sequence. This is true: since F is a cumulative distribution function on $\mathbb{R}$, it is non-decreasing; it follows that for every two $1 \le j < j' \le m$, $U_{(j)} \ge U_{(j')}$ implies that $F^{-1}(U_{(j)}) \ge F^{-1}(U_{(j')})$. □

Proof of Theorem 4.10. In this proof, we write the density function of X as $p_X(\cdot)$ and the density function of Y as $p_Y(\cdot)$, to distinguish between the two.

If $X \sim \mathcal{B}\text{eta}(m-j+1, j)$, then its density function satisfies

$$p_X(x) = \frac{(x)^{m-j}(1-x)^{j-1}}{\mathrm{B}(m-j+1, j)}, \qquad \forall x \in [0,1].$$

Based on Fact A.5, $\mathrm{B}(m-j+1, j) = \frac{(m-j)!(j-1)!}{m!}$, which proves the first claim.

Secondly, if $X \sim \mathcal{B}\text{eta}(m-j+1, j-\ell)$, then the density of $Y = u_{(\ell)} \cdot X$, is given by,

$$p_Y(y) = p_X\left(\frac{y}{u_{(\ell)}}\right) \cdot \frac{dX}{dY} = \frac{1}{\mathrm{B}(m-j+1, j-\ell)} \cdot \left(\frac{y}{u_{(\ell)}}\right)^{m-j} \left(1 - \left(\frac{y}{u_{(\ell)}}\right)\right)^{j-\ell-1} \cdot \frac{1}{u_{(\ell)}},$$

Noting that $\mathrm{B}(m-j+1, j-\ell) = \frac{(m-j)!(j-\ell-1)!}{(m-\ell)!}$ proves the second claim.

The third claim follows from similar argument as the second one.

□

B.3. Proof for Theorem 4.8

We need the following fact.

Fact B.3 ((Guntuboyina, 2019)). The joint density of $U_{(1)}, \ldots, U_{(m)}$ is given by

$$p(u_{(1)}, u_{(2)}, \ldots, u_{(m)}) = m!, \tag{23}$$

where $1 \ge u_{(1)} \ge u_{(2)} \cdots \ge u_{(m)} \ge 0$.

As a sanity check, note that the simplex $\Delta_n^* \doteq \{u_{(1)}, u_{(2)} \ldots, u_{(m)} \in \mathbb{R}^m : 1 \ge u_{(1)} \ge u_{(2)} \ldots \ge u_{(m)} \ge 0\}$ has volume $1/m!$. Since $U_{(1)}, \ldots, U_{(m)}$ distribute uniformly over Δ_n^*, each point of the simplex has density $m!$. Based on this fact, we can derive the following lemma.

Lemma B.4. *Let $j_1, \dots, j_t \in [m]$ be an increasing sequence of indexes. Then the joint distribution of $U_{(j_1)}, \dots, U_{(j_t)}$ is given by*

$$p\big(u_{(j_1)}, \dots, u_{(j_t)}\big) = m! \cdot \frac{\big(u_{(j_t)}\big)^{m-j_t}}{(m-j_t)!} \cdot \frac{\big(1-u_{(j_1)}\big)^{j_1-1}}{(j_1-1)!} \cdot \prod_{s=2}^{t} \frac{\big(u_{(j_{s-1})} - u_{(j_s)}\big)^{j_s-j_{s-1}-1}}{(j_s-j_{s-1}-1)!}. \tag{24}$$

Proof of Lemma. $p\big(u_{(j_1)}, \dots, u_{(j_t)}\big)$ is given by the following integration:

$$\int_0^1 \cdots \int_0^1 m! \cdot \mathbb{1}_{\left[0 \le u_{(m)} \le \dots \le u_{(1)} \le 1\right]} \, d_{u_{(m)}} \dots d_{u_{(1+j_t)}} \, d_{u_{(-1+j_t)}} \dots d_{u_{(1+j_{t-1})}} \cdots \, d_{u_{(-1+j_1)}} \dots d_{u_{(1)}}.$$

Integrating with respect to $u_{(m)}$ over the range $[0, u_{(m-1)}]$, we have

$$\int_0^1 \cdots \int_0^1 m! \cdot u_{(m-1)} \cdot \mathbb{1}_{\left[0 \le u_{(m-1)} \le \dots \le u_{(1)} \le 1\right]} \, d_{u_{(m-1)}} \dots d_{u_{(1+j_t)}} \, d_{u_{(-1+j_t)}} \dots d_{u_{(1+j_{t-1})}} \cdots \, d_{u_{(-1+j_1)}} \dots d_{u_{(1)}}.$$

Then integrate $u_{(m-1)}$ over the range $[0, u_{(m-2)}]$, all the way down to the integral with respect to $u_{(1+j_t)}$ over the range $[0, u_{(j_t)}]$. We obtain

$$\int_0^1 \cdots \int_0^1 m! \cdot \frac{\big(u_{(j_t)}\big)^{m-j_t}}{(m-j_t)!} \cdot \mathbb{1}_{\left[0 \le u_{(j_t)} \le u_{(-1+j_t)} \cdots \le u_{(1)} \le 1\right]} \, d_{u_{(-1+j_t)}} \dots d_{u_{(1+j_{t-1})}} \cdots \, d_{u_{(-1+j_1)}} \dots d_{u_{(1)}}.$$

Integrating with respect to $u_{(-1+j_t)}$ over the range $[u_{(j_t)}, u_{(-2+j_t)}]$, we have

$$\int_0^1 \cdots \int_0^1 m! \frac{\big(u_{(j_t)}\big)^{m-j_t}}{(m-j_t)!} \big(u_{(-2+j_t)} - u_{(j_t)}\big) \cdot \mathbb{1}_{\left[0 \le u_{(j_t)} \le u_{(-2+j_t)} \cdots \le u_{(1)} \le 1\right]} \, d_{u_{(-2+j_t)}} \dots d_{u_{(1+j_{t-1})}} \cdots \, d_{u_{(-1+j_1)}} \dots d_{u_{(1)}}.$$

Then integrate $u_{(-2+j_t)}$ over the range $[u_{(j_t)}, u_{(-3+j_t)}]$, all the way down to $u_{(1+j_{t-1})}$ over the range $[u_{(j_t)}, u_{(j_{t-1})}]$:

$$\int_0^1 \cdots \int_0^1 \left\{ \begin{array}{c} m! \cdot \dfrac{\big(u_{(j_t)}\big)^{m-j_t}}{(m-j_t)!} \cdot \dfrac{\big(u_{(j_{t-1})} - u_{(j_t)}\big)^{j_t-j_{t-1}-1}}{(j_t-j_{t-1}-1)!} \\ \cdot \mathbb{1}_{\left[0 \le u_{(j_{t-1})} \le u_{(-1+j_{t-1})} \cdots \le u_{(1)} \le 1\right]} \end{array} \right\} d_{u_{(-1+j_{t-1})}} \dots d_{u_{(1+j_{t-2})}} \cdots \, d_{u_{(-1+j_1)}} \dots d_{u_{(1)}}.$$

Repeating the above efforts proves Equation (24). □

Proof of Theorem 4.8.

Claim One: Applying Lemma B.4 directly, we have

$$p\big(u_{(j)}\big) = m! \cdot \frac{\big(u_{(j)}\big)^{m-j}}{(m-j)!} \cdot \frac{\big(1-u_{(j)}\big)^{j-1}}{(j-1)!}, \qquad \forall u_{(j)} \in [0,1],$$

which proves the first claim.

Claim Two: let $\mathcal{J} = \{\zeta_1, \dots, \zeta_c\}$, s.t., $\zeta_1 < \cdots < \zeta_c < j$. Following the notation of Theorem 4.8, we have $\ell = \zeta_c$. By Lemma B.4,

$$p\big(u_{(\mathcal{J})}\big) = m! \cdot \frac{\big(u_{(\zeta_c)}\big)^{m-\zeta_c}}{(m-\zeta_c)!} \cdot \frac{\big(1-u_{(\zeta_1)}\big)^{\zeta_1-1}}{(\zeta_1-1)!} \cdot \prod_{s=2}^{c} \frac{\big(u_{(\zeta_{s-1})} - u_{(\zeta_j)}\big)^{\zeta_s-\zeta_{s-1}-1}}{(\zeta_s-\zeta_{s-1}-1)!},$$

$$p\big(u_{(\mathcal{J}\cup\{j\})}\big) = m! \cdot \frac{\big(u_{(j)}\big)^{m-j}}{(m-j)!} \cdot \frac{\big(u_{(\zeta_c)} - u_{(j)}\big)^{j-\zeta_c-1}}{(j-\zeta_c-1)!} \cdot \frac{\big(1-u_{(\zeta_1)}\big)^{\zeta_1-1}}{(\zeta_1-1)!} \cdot \prod_{s=2}^{c} \frac{\big(u_{(\zeta_{s-1})} - u_{(\zeta_s)}\big)^{\zeta_s-\zeta_{s-1}-1}}{(\zeta_s-\zeta_{s-1}-1)!}.$$

Hence,

$$p\big(u_{(j)} \mid u_{(\mathcal{J})}\big) = \frac{p\big(u_{(\mathcal{J}\cup\{j\})}\big)}{p\big(u_{(\mathcal{J})}\big)} = \frac{(m-\zeta_c)!}{(j-\zeta_c-1)!(m-j)!}\left(\frac{u_{(\zeta_c)}-u_{(j)}}{u_{(\zeta_c)}}\right)^{j-\zeta_c-1}\left(\frac{u_{(j)}}{u_{(\zeta_c)}}\right)^{m-j}\frac{1}{u_{(\zeta_c)}}.$$

Similarly, the densities of $p\big(u_{(\zeta_c)}\big), p\big(u_{(\zeta_c)}, u_{(j)}\big)$ are given by

$$\begin{aligned}
p\big(u_{(\zeta_c)}\big) &= m! \cdot \frac{\big(u_{(\zeta_c)}\big)^{m-j}}{(m-\zeta_c)!} \cdot \frac{\big(1-u_{(\zeta_c)}\big)^{\zeta_c-1}}{(\zeta_c-1)!}\,, \\
p\big(u_{(\zeta_c)}, u_{(j)}\big) &= m! \cdot \frac{\big(u_{(j)}\big)^{m-j}}{(m-j)!} \cdot \frac{\big(u_{(\zeta_c)}-u_{(j)}\big)^{j-\zeta_c-1}}{(j-\zeta_c-1)!} \cdot \frac{\big(1-u_{(\zeta_c)}\big)^{\zeta_c-1}}{(\zeta_c-1)!}\,.
\end{aligned}$$

It is easy to see that

$$p\big(u_{(j)} \mid u_{(\zeta_c)}\big) = \frac{p\big(u_{(\zeta_c)}, u_{(j)}\big)}{p\big(u_{(\zeta_c)}\big)} = p\big(u_{(j)} \mid u_{(\mathcal{J})}\big).$$

Claim Three: let $\mathcal{J} = \{\zeta_1, \ldots, \zeta_c\}$, s.t., $\zeta_1 < \ldots < \zeta_c$, and there exists $c' \in [c-1]$ for which $\zeta_{c'} < j < \zeta_{c'+1}$. Following the notation of Theorem 4.8, we have $\ell = \zeta_{c'}, r = \zeta_{c'+1}$. By Lemma B.4,

$$\begin{aligned}
p\big(u_{(\mathcal{J})}\big) &= m! \cdot \frac{\big(u_{(\zeta_c)}\big)^{m-\zeta_c}}{(m-\zeta_c)!} \cdot \frac{\big(1-u_{(\zeta_1)}\big)^{\zeta_1-1}}{(\zeta_1-1)!} \cdot \prod_{s=2}^{c} \frac{\big(u_{(\zeta_{s-1})}-u_{(\zeta_s)}\big)^{\zeta_s-\zeta_{s-1}-1}}{(\zeta_s-\zeta_{s-1}-1)!}\,, \\
p\big(u_{(\mathcal{J}\cup\{j\})}\big) &= m! \cdot \frac{\big(u_{(\zeta_c)}\big)^{m-\zeta_c}}{(m-\zeta_c)!} \cdot \frac{\big(1-u_{(\zeta_1)}\big)^{\zeta_1-1}}{(\zeta_1-1)!} \cdot \frac{\big(u_{(\zeta_{c'})}-u_{(j)}\big)^{j-\zeta_{c'}-1}}{(j-\zeta_{c'}-1)!} \cdot \frac{\Big(u_{(j)}-u_{(\zeta_{c'+1})}\Big)^{\zeta_{c'+1}-j-1}}{(\zeta_{c'+1}-j-1)!} \\
&\quad \cdot \prod_{j=2}^{c'} \frac{\big(u_{(\zeta_{s-1})}-u_{(\zeta_s)}\big)^{\zeta_s-\zeta_{s-1}-1}}{(\zeta_s-\zeta_{s-1}-1)!} \cdot \prod_{j=c'+2}^{c} \frac{\big(u_{(\zeta_{s-1})}-u_{(\zeta_s)}\big)^{\zeta_s-\zeta_{s-1}-1}}{(\zeta_s-\zeta_{s-1}-1)!}\,.
\end{aligned}$$

Hence,

$$\begin{aligned}
p\big(u_{(j)} \mid u_{(\mathcal{J})}\big) = \frac{p\big(u_{(\mathcal{J}\cup\{j\})}\big)}{p\big(u_{(\mathcal{J})}\big)} &= \frac{(\zeta_{c'+1}-\zeta_{c'}-1)!}{(j-\zeta_{c'}-1)!(\zeta_{c'+1}-j-1)!} \\
&\quad \cdot \left(\frac{u_{(\zeta_{c'})}-u_{(j)}}{u_{(\zeta_{c'})}-u_{(\zeta_{c'+1})}}\right)^{j-\zeta_{c'}-1}\left(\frac{u_{(j)}-u_{(\zeta_{c'+1})}}{u_{(\zeta_{c'})}-u_{(\zeta_{c'+1})}}\right)^{\zeta_{c'+1}-j-1}\frac{1}{u_{(\zeta_{c'})}-u_{(\zeta_{c'+1})}}\,.
\end{aligned}$$

Similarly, the densities of $p\Big(u_{(\zeta_{c'})}, u_{(\zeta_{c'+1})}\Big), p\Big(u_{(\zeta_{c'})}, u_{(j)}, u_{(\zeta_{c'+1})}\Big)$ is given by

$$\begin{aligned}
p\Big(u_{(\zeta_{c'})}, u_{(\zeta_{c'+1})}\Big) &= m! \cdot \frac{\Big(u_{(\zeta_{c'+1})}\Big)^{m-\zeta_{c'+1}}}{(m-\zeta_{c'+1})!} \cdot \frac{\Big(u_{(\zeta_{c'})}-u_{(\zeta_{c'+1})}\Big)^{\zeta_{c'+1}-\zeta_{c'}-1}}{(\zeta_{c'+1}-\zeta_{c'}-1)!} \cdot \frac{\big(1-u_{(\zeta_{c'})}\big)^{\zeta_{c'}-1}}{(\zeta_{c'}-1)!}, \\
p\Big(u_{(\zeta_{c'})}, u_{(j)}, u_{(\zeta_{c'+1})}\Big) &= m! \cdot \frac{\Big(u_{(\zeta_{c'+1})}\Big)^{m-\zeta_{c'+1}}}{(m-\zeta_{c'+1})!} \cdot \frac{\big(u_{(\zeta_{c'})}-u_{(j)}\big)^{j-\zeta_{c'}-1}}{(j-\zeta_{c'}-1)!} \\
&\quad \cdot \frac{\Big(u_{(j)}-u_{(\zeta_{c'+1})}\Big)^{\zeta_{c'+1}-j-1}}{(\zeta_{c'+1}-j-1)!} \cdot \frac{\big(1-u_{(\zeta_{c'})}\big)^{\zeta_{c'}-1}}{(\zeta_{c'}-1)!}.
\end{aligned}$$

It is easy to see that

$$p\Big(u_{(j)} \mid u_{(\zeta_{c'})}, u_{(\zeta_{c'+1})}\Big) = \frac{p\Big(u_{(\zeta_{c'})}, u_{(j)}, u_{(\zeta_{c'+1})}\Big)}{p\Big(u_{(\zeta_{c'})}, u_{(\zeta_{c'+1})}\Big)} = p\big(u_{(j)} \mid u_{(\mathcal{J})}\big).$$

□

C. Proofs for Section 5

C.1. Proofs for Section 5.1

Proof of Theorem 5.1. We do not specify the family $\mathcal{H}$ and the distribution μ on $\mathcal{H}$ directly. Instead, we show how we can sample a $\vec{h}$ from $\mathcal{H}$ according to μ: for each $i \in [m]$, independently set

$$\vec{h}[i] \doteq \begin{cases} n, & \text{w.p. } \frac{2k}{m}\,, \\ 0, & \text{w.p. } 1 - \frac{2k}{m}\,. \end{cases} \tag{25}$$

This can also be understood as, for each item $i \in [m]$, with probability $2k/m$ all of the n clients votes for i; and with probability $1 - 2k/m$, none of the clients votes for i.

Assume that: 1) $\mathcal{A}$ retrieves entries from $\vec{h}$ (via random access) without repetition. This only decreases its access cost, since accessing a previously encountered entry does not provide additional information. 2) If $\mathcal{A}$ terminates before retrieving all entries in $\vec{h}$, it continues to read the remaining entries without being charged for the additional accesses. This enables $\mathcal{A}$ to obtain more entries for free. Now, let $J_1, \ldots, J_m \in [m]$ be the order in which $\mathcal{A}$ accesses the entries. The sequence constitutes a permutation of $[m]$, and for each $t \in [m]$, the choice of J_t can depend on previous choices $J_1, \ldots, J_{t-1}$ and outcomes $\vec{h}[J_1], \ldots, \vec{h}[J_{t-1}]$. However, whatever the choice of J_t is, the distribution of $\vec{h}[J_t]$ (conditioned on previous choices and outcomes) is till given by Equation (25). Hence, $\vec{h}[J_1], \ldots, \vec{h}[J_m]$ can be viewed as independent random variables.

Consider the following events. Event E_1 : $\vec{h}$ has at least k non-zero entries. Since each entry of $\vec{h}$ is generated independently, via Chernoff bound (Fact A.1),

$$\Pr\left[\bar{E_1}\right] \le \exp\left(-\frac{(0.5)^2 \cdot 2k}{2}\right) \le \exp\left(-(0.5)^2\right) \le 0.78\,.$$

Event E_2 : the number of non-zero entries among $\vec{h}[J_1]\ , \ldots, \vec{h}[J_{m/50}]$ is less than k. Since $\vec{h}[J_1], \ldots, \vec{h}[J_{m/50}]$ are independent, via Chernoff bound (Fact A.1), and noting that $k \ge 1$,

$$\Pr\left[\bar{E_2}\right] \le \left(\frac{e^{24}}{25^{25}}\right)^{k/25} \le \left(\frac{e^{24}}{25^{25}}\right)^{1/25} \le 0.11\,.$$

Event E_3 : $\mathcal{A}$ returns an $(n-1, k)$-accurate solution. By assumption, we have $\Pr\left[\bar{E_3}\right] \le \beta < 0.1$.

Hence, $\Pr\left[E_1 \cap E_2 \cap E_3\right] \ge 1 - \Pr\left[\bar{E_1}\right] - \Pr\left[\bar{E_2}\right] - \Pr\left[\bar{E_3}\right] \in \Omega(1)$. Observe that, when E_1 and E_3 happens, each item i returned by $\mathcal{A}$ must have frequency $\vec{h}[i] = n$. However, when E_2 happens, $\mathcal{A}$ cannot see k items with non-zero frequency, from its first $m/50$ retrievals.

It follows that , with probability at least $\Pr\left[E_1 \cap E_2 \cap E_3\right]$, $\mathcal{A}$ has access cost at least $m/50$, which proves the theorem.

□

C.2. Proofs for Section 5.2

Proof of Lemma 5.2. Let $\mathcal{O} \doteq \binom{[m]}{k}$ be a shorthand for the collection of all possible outputs of $\mathcal{A}$. Observe that, given an input histogram $\vec{h}_S$ or $\vec{h}_{S_H, S_L}$, if $\mathcal{A}$'s output is $(n-2, k)$-accurate, then it must be a subset of S of size k. Therefore, define $\mathcal{G} \doteq \binom{S}{k}$ be a shorthand for the collection of all $(n-2, k)$-accurate outputs. Via the assumption that $\mathcal{A}$ outputs an $(n-2, k)$-accurate solution with probability at least $1 - \beta$, it holds that

$$\Pr\left[\mathcal{A}(\vec{h}_S) \in \mathcal{G}\right] \ge 1 - \beta\,.$$

Let $\mathcal{F} \doteq \binom{S}{|S|/k}$ be a shorthand for the collection all possible outcomes of S_L. For each $s_\ell \in \mathcal{F}$, define $\mathcal{O}_{s_\ell} \doteq \{o \in \mathcal{O} : o \cap s_\ell \neq \emptyset\}$, the collection of sets in $\mathcal{O}$ that has nonempty intersection with s_ℓ, and $\mathcal{G}_{s_\ell} \doteq \{o \in \mathcal{G} : o \cap s_\ell \neq \emptyset\}$, the collection of sets in $\mathcal{G}$ that has nonempty intersection with s_ℓ. Conditioned on $S_L = s_\ell$,

$$\Pr\left[\mathcal{A}(\vec{h}_{S_H, S_L}) \cap S_L \neq \emptyset \mid S_L = s_\ell\right] = \Pr\left[\mathcal{A}(\vec{h}_{S_H, S_L}) \in \mathcal{O}_{s_\ell} \mid S_L = s_\ell\right] \ge \Pr\left[\mathcal{A}(\vec{h}_{S_H, S_L}) \in \mathcal{G}_{s_\ell} \mid S_L = s_\ell\right],$$

where the inequality holds as $\mathcal{O}_{s_\ell} \supseteq \mathcal{G}_{s_\ell}$. Moreover, since $\mathcal{A}$ is (ε, δ)-DP, it holds that

$$\Pr\left[\mathcal{A}(\vec{h}_{S_H,S_L}) \in \mathcal{G}_{s_\ell} \mid S_L = s_\ell\right] \geq \frac{1}{e^\varepsilon}(\Pr\left[\mathcal{A}(\vec{h}_S) \in \mathcal{G}_{s_\ell}\right] - \delta)\,. \tag{26}$$

Further, since the events that $\mathcal{A}(\vec{h}_S) = o$ are mutually exclusive for different values of $o \in \mathcal{G}$, we have

$$\Pr\left[\mathcal{A}(\vec{h}_S) \in \mathcal{G}\right] = \sum_{o\in\mathcal{G}} \Pr\left[\mathcal{A}(\vec{h}_S) = o\right] \geq 1 - \beta\,, \tag{27}$$

$$\Pr\left[\mathcal{A}(\vec{h}_S) \in \mathcal{G}_{s_\ell}\right] = \sum_{o\in\mathcal{G}, o\cap s_\ell\neq\emptyset} \Pr\left[\mathcal{A}(\vec{h}_S) = o\right] = \sum_{o\in\mathcal{G}} \mathbb{1}_{[o\cap s_\ell\neq\emptyset]} \cdot \Pr\left[\mathcal{A}(\vec{h}_S) = o\right]\,. \tag{28}$$

Finally,

$$\begin{aligned}
\Pr_{S_L,\mathcal{A}}\left[\mathcal{A}(\vec{h}_{S_H,S_L}) \cap S_L \neq \emptyset\right] &= \sum_{s_\ell\in\mathcal{F}} \Pr\left[S_L = s_\ell\right] \cdot \Pr\left[\mathcal{A}(\vec{h}_{S_H,S_L}) \in \mathcal{O}_{s_\ell} \mid S_L = s_\ell\right] \\
&\geq \sum_{s_\ell\in\mathcal{F}} \Pr\left[S_L = s_\ell\right] \cdot \Pr\left[\mathcal{A}(\vec{h}_{S_H,S_L}) \in \mathcal{G}_{s_\ell} \mid S_L = s_\ell\right] \\
&\geq \sum_{s_\ell\in\mathcal{F}} \Pr\left[S_L = s_\ell\right] \cdot e^{-\varepsilon}\left(\sum_{o\in\mathcal{G}} \mathbb{1}_{[o\cap s_\ell\neq\emptyset]} \cdot \Pr\left[\mathcal{A}(\vec{h}_S) = o\right] - \delta\right) \\
&= -e^{-\varepsilon}\delta + e^{-\varepsilon}\sum_{o\in\mathcal{G}} \Pr\left[\mathcal{A}(\vec{h}_S) = o\right]\left(\sum_{s_\ell\in\mathcal{F}} \mathbb{1}_{[o\cap s_\ell\neq\emptyset]} \cdot \Pr\left[S_L = s_\ell\right]\right) \\
&\overset{(a)}{=} -e^{-\varepsilon}\cdot\delta + e^{-\varepsilon}\sum_{o\in\mathcal{G}} \Pr\left[\mathcal{A}(\vec{h}_S) = o\right]\left(1 - \frac{\binom{|S|-k}{|S|/k}}{\binom{|S|}{|S|/k}}\right) \\
&\overset{(b)}{\geq} -e^{-\varepsilon}\cdot\delta + e^{-\varepsilon}\sum_{o\in\mathcal{G}} \Pr\left[\mathcal{A}(\vec{h}_S) = o\right](1 - \exp(-1)) \\
&\geq -e^{-\varepsilon}\cdot\delta + e^{-\varepsilon}(1-\beta)\left(1 - e^{-1}\right) \\
&\geq e^{-\varepsilon}\left(1 - \beta - \delta - e^{-1}\right),
\end{aligned}$$

where equation (a) follows, since $\sum_{s_\ell\in\mathcal{F}} \mathbb{1}_{[o\cap s_\ell\neq\emptyset]} \cdot \Pr\left[S_L = s_\ell\right]$ can interpreted as the probability that given a subset o of S of size k, the sampled subset S_L has nonempty intersection with o; and inequality (b) follows, since

$$\begin{aligned}
\frac{\binom{|S|-k}{|S|/k}}{\binom{|S|}{|S|/k}} &= \frac{|S|-k}{|S|}\cdots\frac{|S|-k-|S|/k+1}{|S|-|S|/k+1} \\
&\leq \left(\frac{|S|-k}{|S|}\right)^{|S|/k} \\
&\leq \exp\left(-\frac{k}{|S|}\cdot\frac{|S|}{k}\right) \\
&= \exp(-1)\,.
\end{aligned}$$

□

C.3. Proofs for Section 5.3

Proof of Lemma 5.5. Assume that: 1) if $\mathcal{A}$ terminates before performing η accesses operations, it continues to perform more until it reaches η, and will not be charged for any additional access. Now, let $J_1, \ldots, J_\eta$ represents the first η operations of $\mathcal{A}$: each J_t is either a character 's', implying that the $t^{(th)}$ operation is a sorted access, or an integer in $[m]$, implying that the $t^{(th)}$ operation is a random access to the entry $\vec{h}_{S_H,S_L}[J_t]$. Further, let $J'_1, \ldots, J'_t$ be the subsequence of all random access in $J_1, \ldots, J_\eta$.

Consider an alternative algorithm $\mathcal{A}'$ that operates as follows: it first performs 9η sorted accesses to obtain all frequencies of all items from S_H, followed by random accesses $J'_1, \ldots, J'_t$. It is clear that, $\mathcal{A}'$ retrieves a greater number of entries than the first η operations of $\mathcal{A}$.

We bound the probability that $\mathcal{A}'$ does not retrieve an entry from S_L. First consider random access J'_1. If J'_1 had already been retrieved by sorted access, then clearly $J'_1 \notin S_L$. Otherwise, conditioned on the fact that the entries S_H have been determined, according to the manner S and S_L are generated, each item in $[m] \setminus (S_H)$ belongs to S_L with equal probability. Hence,

$$\Pr\left[J'_1 \notin S_L\right] \geq 1 - \frac{\tau/k}{m - (k-1)\tau/k}\,.$$

In general, for each $1 < \ell \leq t$, suppose that $J'_1, \ldots, J'_{\ell-1}$ does not belong to S_L, and has revealed $m_\ell \leq \ell - 1$ distinct items from $[m] \setminus (S_H)$. Whatever the choice of J'_ℓ is, and the items $J'_1, \ldots, J'_{\ell-1}$ are, each of the remaining items that have not been queried belong to S_L with equal probability. Hence

$$\Pr\left[J'_\ell \notin S_L\right] \geq 1 - \frac{\tau/k}{m - (k-1)\tau/k - m_\ell}.$$

Noting that $m_\ell < t \leq \eta$, we have

$$\begin{aligned}
\Pr\left[J'_1, \ldots, J'_t \notin S_L\right] &\geq \prod_{\ell=1}^{t} \left(1 - \frac{\tau/k}{m - (k-1)\tau/k - m_\ell}\right) \\
&\geq \prod_{\ell=1}^{t} \left(1 - \frac{\tau/k}{m - (k-1)\tau/k - \eta}\right) \\
&\overset{(a)}{\geq} \prod_{\ell=1}^{t} \exp\left(-\frac{2\tau/k}{m - (k-1)\tau/k - \eta}\right) \\
&\geq \exp\left(-\frac{\tau^2/(10k)}{m - (k-1)\tau/k - \eta}\right) \\
&\geq \exp\left(-\frac{m/10}{m - 21\sqrt{mk}/20}\right) \\
&\geq \exp\left(-\frac{m/10}{m - 21m/40}\right) \\
&\geq 0.81.
\end{aligned}$$

where inequality (a) holds, since $1 - x \geq e^{-2x}, \forall x \in [0, 3/4]$, and

$$\tau/k \leq \frac{3}{4}(m - (k-1)\tau/k - \eta) \tag{29}$$

$$\iff \tau\left(\frac{3}{4} + \frac{1}{4k} + \frac{3}{40}\right) \leq \frac{3}{4}m \tag{30}$$

The last inequality holds, since $\frac{3}{4} + \frac{1}{4k} + \frac{3}{40} \leq 3/2$ for $k \in \mathbb{N}^+$, and $2\tau = 2\sqrt{mk} \leq m$, i.e., $k \leq m/4$. □