Differentially Private Hierarchical Heavy Hitters[*]

Ari Biswas[†] Graham Cormode[‡] Yaron Kanza[§] Divesh Srivastava[§]
Zhengyi Zhou[§]

Abstract

Hierarchical Heavy Hitters (HHH) was introduced by Cormode et al. [1] as a generalisation of the heavy hitter problem. This work studies HHH under the lens of differential privacy in both the streaming and non-streaming setting. In the non-streaming setting, we show that that the relative error in estimating the count for any prefix is independent of the height of the hierarchy. Additionally, our algorithms also improve the error guarantees of Ghazi et al. [2] for the problem of counting over trees. In the streaming setting, the main issue is that although the exact version of HHH has low global sensitivity (as counting queries are 1-sensitive), the approximation functions due to streaming have high global sensitivity linear in the available space. Despite this, we show that the absolute error for estimating frequencies is independent of the available space.

1 Introduction

The task of finding *Heavy Hitters* (HH), a.k.a. frequent items, in a dataset is one of the most well-studied problems in data science. The task has been studied under the streaming model of computation [3, 1], distributed computation [4], and even through the lens of secure computation [5]. In this work, we adopt the lens of *differential privacy* (DP) to study the *Hierarchical* *Heavy Hitters* (HHH) problem, introduced by Cormode et al. [1] as a generalisation of the heavy hitter problem. The problem of DP-HHH is motivated by the observation that data is often both *hierarchical* and *confidential*. Consider checking for evidence of discrimination in mortgage lending decisions, as discussed in Lee and Floridi [6]. An analyst is given a database of historical lending decisions and asked to ascertain if a particular demographic has been treated unfairly. Personal information about loan applicants is inherently hierarchical. For example, a person's residential address can be divided into street address, postcode, village, city, country, and so on. As historical data is often difficult to obtain, any given dataset might not include enough applicants from every fine-grained portion of the hierarchy. However, if we analysed the data at a coarser granularity, we might find a statistically significant number of participants to draw reliable conclusions. Naturally, whether hierarchical or not, demographic information is considered *highly confidential*. It is well known that even releasing summary statistics about a population can leak information about individuals in the dataset. Differential privacy has become the de facto standard for defending against such leakage. As a result, given a dataset, we wish to output its hierarchical heavy hitters privately.

Note that finding hierarchical heavy hitters is *not* the same as finding heavy hitters at each level in a hierarchy (referred to as counting over trees in Ghazi et al. [2]). Hierarchical heavy hitters, which we formally define later (Definition 5) is a generalisation of the heavy hitters problem. At a high level, apart from telling us *if* an element is heavy, it also tells us *how* it is heavy. HHH allows us to distinguish between an element that is heavy because it has a heavy child (or a few heavy children) and an element that is heavy because it has many light children that are cumulatively heavy. Furthermore, if we are given the set of (exact) hierarchical heavy hitters of a dataset, we can derive the heavy hitters at each level of the hierarchy. The converse, however, is not true. Figure 1 illustrates this difference with a toy example. The figure shows a

[*] A preprint of the full version of this paper can be found at https://abiswas3.github.io/papers/DPHHH.pdf, which is currently under review.

[†] University Of Warwick

[‡] Meta AI, University Of Warwick

[§] AT&T Research Labs

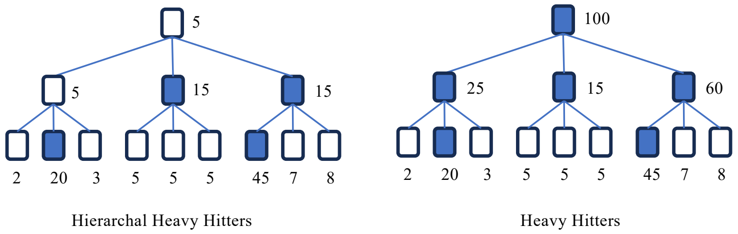

Figure 1: A dataset of 100 elements over a hierarchy with residual counts (left) and unconditional counts (right).

dataset of 100 elements drawn from a hierarchy of height 3. Each node in the tree corresponds to an element in the hierarchical universe. The leaf nodes are fully specified elements, while the root node describes the fully generalised element of the hierarchy. The edges between nodes represent a partial order between elements of the hierarchy (see Section A for formal details). Given a public threshold of $\tau = 10$, a node is heavy if its count exceeds τ. The counts listed next to the nodes on the *right* tree are the unconditional counts (Definition 1) of the node in the dataset. The nodes marked in blue on the right tree are the heavy hitters at each level of the hierarchy. The nodes marked in blue on the left tree are the hierarchical heavy hitters of the dataset. The counts listed next to the nodes on the *left* tree are called residual counts (Definition 2). The residual count of a node is the count of a node ignoring its heavy children, whereas the unconditional count of a node is just the sum of the counts of its children. Observe that the root node is *not* a hierarchical heavy hitter, although its absolute count is greater than the threshold. The root node is heavy only because it has heavy children, not because it is an aggregation of several light children. If we were to just see the output of heavy hitters, we would lose this information.

1.1 Related work

Streaming HHH. The hierarchical heavy hitter problem was first defined and studied in the streaming setting, as the offline problem is straightforward. Initial work defined the problem for streams of data drawn from a single hierarchy, and showed upper bounds on the problem, by building streaming heavy hitter summaries of data at each level [1, 7, 8]. Subsequent work extended the problem to data with multiple hierarchical attributes [9, 3], and showing lower bounds on the space required to solve the problem [10, 8]. In the streaming model, data arrives incrementally, and we assume that the algorithm does not have enough space to store the entire dataset, or enough counters for each element of the data universe. Mitzenmacher et al. [8] show that approximating HHH via the Space Saving algorithm (SS) for heavy hitters [11] is optimal in terms of error and space complexity in the streaming setting.

Private counting. Despite its relevance to data analytics, the HHH problem has not been previously studied under differential privacy. However, there has been much research on simpler non-hierarchical heavy hitter estimation under privacy, in both the non-streaming [12, 13, 14, 15, 16], and streaming models [17, 18]. In this work, we show that despite this extensive body of work in the non-hierarchical setting, we need new algorithms to privately estimate HHH efficiently in theory and practice. The most closely related work to ours is concerned with outputting “unconditional counts” in a hierarchy, due to Ghazi et al. [2]. As any fully specified element (“leaf”) affects the counts for nodes at each level of the hierarchy, we must account for it every time we release a count for a node that is an ancestor of said leaf. Hence, the DP error scales linearly with the height h of the hierarchy when using basic composition, or $\sqrt{h}$ under advanced composition [19]. Edmonds et al. [20] provide lower bounds showing that such a polynomial dependence on the hierarchy height is unavoidable if we want to estimate just the unconditional counts for every element in the hierarchy[1] with *pure* or *approximate* differential privacy (see Appendix G).

Ghazi et al. [2] circumvent this dependence on the height of the hierarchy by relaxing the problem to consider *relative* error in estimating node counts, where the estimation gap for a node scales with the absolute

[1] This is a simpler problem than HHH, which involves estimating conditional counts. Therefore these lower bounds apply to HHH as well.

count of the node[2]. Their algorithmic guarantees replace the linear dependence on the height of the hierarchy with a linear dependence of the maximum number of hierarchical heavy hitters in a dataset[3]. Although this is an asymptotic improvement, the number of heavy hitters is much larger than the hierarchy height for any practical scenario we can envisage. Hence the real-world performance will be much worse than the naive baseline of estimating the counts at each level and paying for h levels of composition. Concretely, most real-world datasets are associated with shallow and wide hierarchies. Consider a dataset of bit strings of size $n = 10^6$, and a threshold of 2500, so there are up to 400 hierarchical heavy hitters. For this algorithm to improve on the simple baseline, the hierarchy would need to have more than 400 levels, implying over 2^{400} elements.

2 Our Results

By definition, differential privacy restricts us from outputting exact answers or using a deterministic algorithm to compute approximate values. Thus, keeping in line with the definitions introduced in [8, 1] we define the task of *approximate heavy hitters*, where the estimates are within some approximation error Δ with high confidence. Exact definitions are provided in Problem 1 and Problem 2 in Appendix C.

Non-Streaming Results

In this work, we show that the relative error for any node when estimating private hierarchical heavy hitters scales by a *much* smaller constant that is independent of *both* the height h of the hierarchy and the number of hierarchical heavy hitters in the tree. Our algorithm is simpler than that of Ghazi et al. [2, Algorithm 1], and it can be used to solve the more general problem with better error guarantees than prior attempts would imply. Our algorithm is optimal in the sense that we match the constants in optimal algorithms for DP heavy hitters in the non-hierarchical setting [12], and thus it incurs the lowest error one can hope for in the non-streaming setting. A technical overview is provided below.

A first approach to hierarchical heavy hitter estimation probelem is to release h Laplace histograms by adding noise drawn from a Laplace noise distribution with scale $\frac{h}{\varepsilon}$ to every node in the hierarchy. By the privacy of the Laplace histograms (Fact 1), each level is ε/h-private. As any node contributes to at most h counts, by basic composition and the privacy of the Laplace mechanism, the set of realised counts is $(\varepsilon, 0)$-DP. Then, one could compute hierarchical heavy hitters via post-processing, which does not affect the privacy of the algorithm (Fact 3). As we add noise with scale $\frac{h}{\varepsilon}$ to each node in $\mathcal{H}$, the DP error per node scales linearly in the height of the hierarchy. Furthermore, if we wanted to upper bound the simultaneous error over all prefixes in the hierarchy, then we need to apply the union bound over all nodes of the hierarchy. As the number of nodes is exponential in the height of the hierarchy, the relative estimation error scales $\texttt{poly}(h)$.

One solution to circumventing such a union bound over all the elements of the universe when the size of the universe is very large is to use stability histograms [21, 12, 22] instead of Laplace histograms. For fixed privacy parameters $\varepsilon \in (0, \log n)$ and $\delta = \mathsf{negl}(n)$, the key intuition behind stability histograms is that if we only released private estimates (using the Laplace mechanism) for nodes with "large enough" non-zero frequency, the simultaneous error bound improves from $\log(|\mathcal{H}|)$ to $\log(n) < \log(1/\delta)$ (as there are at most n elements in X). This is advantageous when $|\mathcal{H}| \gg n$ meaning that many nodes have zero frequency. Thus, if we output stability histograms with $\delta' = \delta/h$ at each level then, by basic composition, the final output does not contain any isolated elements with probability at least $1 - \delta$. In this way, stability histograms allow us to circumvent the union bound over all the nodes in $\mathcal{H}$, at the cost of going from pure to approximate DP. It does not however, circumvent the issue discussed above where the scale of the DP noise is $\frac{h}{\varepsilon}$. One approach to resolving this issue is to make repeated use of the Above-Threshold Algorithm by Dwork et al. [23]. Given a hierarchy $\mathcal{H}$ then, by definition, there can be at most $c = \frac{n}{\tau - \Delta}$ hierarchical heavy hitters. Observe that c is a constant that depends only on τ and Δ, and is independent of the height of the hierarchy h. When the height of the hierarchy h is large and $c \ll h$, we would incur lower per-node DP error if the scale of the noise were $\frac{c}{\varepsilon}$ instead of $\frac{h}{\varepsilon}$. Ghazi et al. [2] observed this fact and proposed an algorithm that traverses the tree bottom up. At each level of the tree, their algorithm inspects only one node, the one with maximal

[2] In other words, nodes with large unconditional frequencies are allowed to tolerate more estimation error than nodes with smaller unconditional frequencies.

[3] Algorithm 1 of their paper uses the constant c, independent of the height of the hierarchy, as an upper bound on the maximum number of hierarchical heavy hitters.

unconditional count for that level. There are at most h such nodes (one per level). However, even in the worst case, at most c out of h of these nodes can be conditionally heavy. Thus, we have h counting queries, out of which c are conditionally heavy. Their solution uses the Sparse Vector Technique (SVT) by composing Above-threshold h times, but they pay only for the c conditionally heavy nodes. Once c levels have been identified, the algorithm prunes the tree, and restricts it to just the c levels with at least one conditionally heavy node. Then it estimates the counts of the pruned tree by solving c instances of the private stability histogram estimation problem[4] with privacy budget ε/c. As commented in the introduction, although this is an asymptotic improvement, it is very hard to imagine cases where $c \ll h$. Thus, the simple Laplace stability histograms with composition error h would outperform the algorithm by Ghazi et al. [2] in almost all feasible situations. Our algorithm is based on the following critical observation. Although it is true that there are at most c conditionally heavy levels in the tree, *only one* of the conditionally heavy nodes is really influenced by x'. For the remainder of the heavy nodes, there is no difference in the conditional frequencies of X and X'. It is this structure that we exploit to improve error our guarantees. In prior work, Kaplan et al. [24] show that there are instances where the composition error of the SVT algorithm is suboptimal, even when dealing with a stream of adaptively chosen queries by an unbounded adversary. Instead, they propose a general algorithm (called Threshold Monitor) where DP error of a query scales linearly by a constant $\frac{75\varepsilon(k+1)}{\log 1/\delta}$, where k represents the number of times any data point can contribute to a counting query being heavy. Note that both the SVT algorithm and Threshold monitor are general and accommodate a stream of adaptively chosen counting queries with no structure. In estimating HHH in the non-streaming setting with central differential privacy, we can pre-select the order of queries to be bottom up (leveraging monotonicity and limited influence of any node), and we do not have to deal with adaptive queries. Thus, in the case of hierarchical heavy hitter estimation, paying a price of c/ε of SVT or the larger constants of Threshold Monitor for adaptivity in composition is wasteful. Instead, we can use a simpler algorithm, and with more specialized privacy analysis we show that DP noise per node with scales $\frac{4}{\varepsilon}$ instead of $\frac{c}{\varepsilon}$, and thus is independent of *both* the height of the hierarchy and the number of heavy hitters in $\mathcal{H}$. Note that this matches the constants for non-hierarchical private algorithm for detecting a single heavy query from a stream [12]. Thus, this is the best one can hope to do. For full details about the algorithms and proofs, we refer the reader Appendix B.

Streaming Setting In the streaming setting, along with DP error and composition error, we also need to account for the approximation error due to space constraints. The main issue with streaming algorithms is that although the exact version of the function (exact hierarchical heavy hitters) has low global sensitivity (as counting queries are 1-sensitive), the approximation function can have high global sensitivity. For instance, the Space Saving (SS) algorithm described by Mitzenmacher et al. [8] is optimal in the non-private setting, but Chan et al. [18] show that the global sensitivity of the approximation function induced by SS scales linearly with the number of counters per sketch (denoted with κ in this document). This would imply an error that scales linearly with κ. Lebeda and Tetek [17] improve on this by providing a DP mechanism for non-hierarchical heavy hitters, where the estimation error of the protocol does not rely on global sensitivity. Inspired by [17], we design algorithms for hierarchical heavy hitters with DP noise whose variance is independent of κ. The intuition behind our algorithm is that although the approximation algorithm we use has high sensitivity, the counters in a sketch are highly correlated, with few degrees of freedom. This structure (correlated counters) can be used to bypass composition bounds typically enforced due to high global sensitivity of the function. This observation is closely related to why the seminal Sparse Vector Technique algorithm (SVT) by Dwork et al. [23] also circumvents composition bounds. Although the two appear unrelated at first glance, we show that releasing private counts for our sketching algorithm and SVT algorithm are essentially equivalent and use the *same* underlying theoretical concepts to bypass composition. More broadly, the message of this work is that *"where we have structure (sparsity, monotonicity, correlation, etc.), we can leverage this structure to circumvent basic or advanced composition bounds"*. Hierarchies offer structure in that the frequency of elements higher up in the hierarchy is computed using frequencies of elements lower down. In streams, we show that despite high global sensitivity, we can leverage correlation between counters in a sketch to circumvent composition bounds. We refer the reader to Appendix F for further discussion on the role of structure in circumventing composition and Appendix C for full proofs.

[4] While Ghazi et al. do not explicitly refer to stability histograms, their use of bounded truncated Laplace noise is equivalent. In the case of bounded noise, the union bound is affected by the size of the support of the truncated noise distribution, which is upper bounded by $O(1/\delta)$. For stability histograms, it is affected by the number of elements which is upper bounded by $1/\delta$.

References

[1] Graham Cormode, Flip Korn, Shanmugavelayutham Muthukrishnan, and Divesh Srivastava. Finding hierarchical heavy hitters in data streams. In *Proceedings 2003 VLDB Conference*, pages 464–475. Elsevier, 2003.

[2] Badih Ghazi, Pritish Kamath, Ravi Kumar, Pasin Manurangsi, and Kewen Wu. On differentially private counting on trees. *arXiv preprint arXiv:2212.11967*, 2022.

[3] Graham Cormode, Flip Korn, S Muthukrishnan, and Divesh Srivastava. Finding hierarchical heavy hitters in streaming data. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 1(4):1–48, 2008.

[4] Albert Cheu. Differential privacy in the shuffle model: A survey of separations. *arXiv preprint arXiv:2107.11839*, 2021.

[5] Henry Corrigan-Gibbs and Dan Boneh. Prio: Private, robust, and scalable computation of aggregate statistics. In *14th USENIX symposium on networked systems design and implementation (NSDI 17)*, pages 259–282, 2017.

[6] Michelle Seng Ah Lee and Luciano Floridi. Algorithmic fairness in mortgage lending: from absolute conditions to relational trade-offs. *Minds and Machines*, 31(1):165–191, 2021.

[7] Yuan Lin and Hongyan Liu. Separator: sifting hierarchical heavy hitters accurately from data streams. In *Advanced Data Mining and Applications: Third International Conference, ADMA 2007 Harbin, China, August 6-8, 2007. Proceedings 3*, pages 170–182. Springer, 2007.

[8] Michael Mitzenmacher, Thomas Steinke, and Justin Thaler. Hierarchical heavy hitters with the space saving algorithm. In *2012 Proceedings of the Fourteenth Workshop on Algorithm Engineering and Experiments (ALENEX)*, pages 160–174. SIAM, 2012.

[9] Graham Cormode, Flip Korn, S Muthukrishnan, and Divesh Srivastava. Diamond in the rough: Finding hierarchical heavy hitters in multi-dimensional data. In *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, pages 155–166, 2004.

[10] John Hershberger, Nisheeth Shrivastava, Subhash Suri, and Csaba D Tóth. Space complexity of hierarchical heavy hitters in multi-dimensional data streams. In *Proceedings of the twenty-fourth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 338–347, 2005.

[11] Ahmed Metwally, Divyakant Agrawal, and Amr El Abbadi. An integrated efficient solution for computing frequent and top-k elements in data streams. *ACM Transactions on Database Systems (TODS)*, 31(3): 1095–1133, 2006.

[12] Victor Balcer and Salil Vadhan. Differential privacy on finite computers. *arXiv preprint arXiv:1709.05396*, 2017.

[13] Raef Bassily, Kobbi Nissim, Uri Stemmer, and Abhradeep Guha Thakurta. Practical locally private heavy hitters. *Advances in Neural Information Processing Systems*, 30, 2017.

[14] Graham Cormode, Cecilia Procopiuc, Divesh Srivastava, and Thanh TL Tran. Differentially private summaries for sparse data. In *Proceedings of the 15th International Conference on Database Theory*, pages 299–311, 2012.

[15] Aleksandra Korolova, Krishnaram Kenthapadi, Nina Mishra, and Alexandros Ntoulas. Releasing search queries and clicks privately. In *Proceedings of the 18th international conference on World wide web*, pages 171–180, 2009.

[16] Arpita Ghosh, Tim Roughgarden, and Mukund Sundararajan. Universally utility-maximizing privacy mechanisms. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 351–360, 2009.

[17] Christian Janos Lebeda and Jakub Tetek. Better differentially private approximate histograms and heavy hitters using the misra-gries sketch. In *Proceedings of the 42nd ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, pages 79–88, 2023.

[18] T H Hubert Chan, Mingfei Li, Elaine Shi, and Wenchang Xu. Differentially private continual monitoring of heavy hitters from distributed streams. In *Privacy Enhancing Technologies: 12th International Symposium, PETS 2012, Vigo, Spain, July 11-13, 2012. Proceedings 12*, pages 140–159. Springer, 2012.

[19] Cynthia Dwork, Aaron Roth, et al. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3–4):211–407, 2014.

[20] Alexander Edmonds, Aleksandar Nikolov, and Jonathan Ullman. The power of factorization mechanisms in local and central differential privacy. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, pages 425–438, 2020.

[21] Mark Bun, Kobbi Nissim, and Uri Stemmer. Simultaneous private learning of multiple concepts. *Journal of Machine Learning Research*, 20(94):1–34, 2019.

[22] Abhradeep Guha Thakurta and Adam Smith. Differentially private feature selection via stability arguments, and the robustness of the lasso. In *Conference on Learning Theory*, pages 819–850. PMLR, 2013.

[23] Cynthia Dwork, Moni Naor, Omer Reingold, Guy N Rothblum, and Salil Vadhan. On the complexity of differentially private data release: efficient algorithms and hardness results. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 381–390, 2009.

[24] Haim Kaplan, Yishay Mansour, and Uri Stemmer. The sparse vector technique, revisited. In *Conference on Learning Theory*, pages 2747–2776. PMLR, 2021.

[25] Pankaj K Agarwal, Graham Cormode, Zengfeng Huang, Jeff M Phillips, Zhewei Wei, and Ke Yi. Mergeable summaries. *ACM Transactions on Database Systems (TODS)*, 38(4):1–28, 2013.

[26] Cynthia Dwork, Guy N Rothblum, and Salil Vadhan. Boosting and differential privacy. In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, pages 51–60. IEEE, 2010.

[27] 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, 2010.

[28] Min Lyu, Dong Su, and Ninghui Li. Understanding the sparse vector technique for differential privacy. *arXiv preprint arXiv:1603.01699*, 2016.

[29] Mark Bun and Thomas Steinke. Concentrated differential privacy: Simplifications, extensions, and lower bounds. In *Theory of Cryptography Conference*, pages 635–658. Springer, 2016.

[30] Cynthia Dwork, Frank McSherry, and Kunal Talwar. The price of privacy and the limits of lp decoding. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, pages 85–94, 2007.

[31] Prosenjit Bose, Evangelos Kranakis, Pat Morin, and Yihui Tang. Bounds for frequency estimation of packet streams. In *SIROCCO*, volume 10, pages 18–20, 2003.

[32] Thomas Steinke and Jonathan Ullman. Between pure and approximate differential privacy. *arXiv preprint arXiv:1501.06095*, 2015.

[33] Peter Kairouz, Sewoong Oh, and Pramod Viswanath. The composition theorem for differential privacy. In *International conference on machine learning*, pages 1376–1385. PMLR, 2015.

[34] Jack Murtagh and Salil Vadhan. The complexity of computing the optimal composition of differential privacy. In *Theory of Cryptography Conference*, pages 157–175. Springer, 2015.

[35] Yan Chen and Ashwin Machanavajjhala. On the privacy properties of variants on the sparse vector technique. *arXiv preprint arXiv:1508.07306*, 2015.

[36] Mark Bun, Thomas Steinke, and Jonathan Ullman. Make up your mind: The price of online queries in differential privacy. In *Proceedings of the twenty-eighth annual ACM-SIAM symposium on discrete algorithms*, pages 1306–1325. SIAM, 2017.

[37] Cynthia Dwork, Vitaly Feldman, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Aaron Leon Roth. Preserving statistical validity in adaptive data analysis. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, pages 117–126, 2015.

[38] Kobbi Nissim, Uri Stemmer, and Salil Vadhan. Locating a small cluster privately. In *Proceedings of the 35th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, pages 413–427, 2016.

[39] Wei Dong, Dajun Sun, and Ke Yi. Better than composition: How to answer multiple relational queries under differential privacy. *Proceedings of the ACM on Management of Data*, 1(2):1–26, 2023.

[40] Thomas Steinke. Composition of differential privacy & privacy amplification by subsampling. *arXiv preprint arXiv:2210.00597*, 2022.

[41] Fuheng Zhao, Dan Qiao, Rachel Redberg, Divyakant Agrawal, Amr El Abbadi, and Yu-Xiang Wang. Differentially private linear sketches: Efficient implementations and applications. *Advances in Neural Information Processing Systems*, 35:12691–12704, 2022.

[42] David Pujol and Damien Desfontaines. Open problem - better privacy guarantees for larger groups. DifferentialPrivacy.org, 06 2023. https://differentialprivacy.org/open-problem-better-privacy-guarantees-for-larger-groups/.

A Prelims And Problem Statement

General Notation. We describe sets with calligraphic font $\mathcal{H}$. For a probability distribution D, we denote with $x \leftarrow\!\!\$\, D$ the event of sampling x according to D. We highlight random variables and samples to distinguish them from constants, as shown above. We write $[n]$ to denote the set $\{1,\ldots,n\}$. For any event E, we denote with $\overline{E}$, the complement of the event. Next, we formalise the idea of a hierarchical domain. We assume that the reader is familiar with the definition of differential privacy and its basic properties. Otherwise, we refer the reader to Appendix D, where we review the results for differential privacy needed in this work.

Hierarchies. Formally, a hierarchical domain is a set $\mathcal{U}$ associated with a partial order $(\succ)$. In streaming literature [8, 1], this partial order is often represented by a function called $\mathsf{Generalise} : \mathcal{U} \to \mathcal{U}$ which maps elements of the universe to other elements in the universe[5]. In this work, it suffices to think of a hierarchical domain as the set of elements that has one to one mapping with the nodes of a rooted tree with finite arity[6]. For any element $x \in \mathcal{U}$, $\mathsf{Generalise}(x)$ refers to the parent or prefix of x. We say an element $*$ is *fully generalised* or the root of the tree if $\mathsf{Generalise}(*) = *$. We say an element e is *fully specified* if there exists no $s \in \mathcal{U}$ such that $e = \mathsf{Generalise}(s)$ (e is a leaf node of the rooted tree representing the hierarchy). We denote by $\mathsf{Generalise}^{(k)}(x)$ as the ancestor of x that is k steps away in the tree representation (element obtained by applying $\mathsf{Generalise}$ k times on x). The pair $(\mathcal{U}, \mathsf{Generalise})$ defines a hierarchical domain $\mathcal{H}$. The height h of the hierarchy represents the maximum number of times *any* fully specified element must be generalised to get a fully generalised element (or more simply, the height of the tree representing the hierarchy). As a concrete example of a single-dimensional hierarchy, one can imagine $\mathcal{U}$ to be the set of prefixes of h-bit strings. When $h = 4$, this universe has 16 fully specified elements: $0000, 0001, \ldots, 1111$. The prefix $000*$ is a generalisation of 0000 and 0001. We use notation $x \succ p$ (read as p is reachable from x) if there exists a $k \in \mathbb{N}$ such that $p = \mathsf{Generalise}^{(k)}(x)$, and $x \succeq p$ if $p = \mathsf{Generalise}^{(k)}(x) \ \vee x = p$.

Remark. *Henceforth, we assume that any dataset X is a multiset of fully specified elements from some hierarchical universe $\mathcal{H}$ of height h.*

Definition 1.1: Unconditional Count Or Absolute Frequency

Given a dataset X, the unconditional frequency of any element $p \in \mathcal{H}$, denoted by $f_X(p)$, is the number of elements in X that generalise to p. Writing $\mathbb{1}[\cdot]$ for the indicator function,

$$f_X(p) = \textstyle\sum_{e \in X} \mathbb{1}[e \succeq p]$$

In Figure 1, the unconditional counts of each node are written next to each node in the tree for the figure on the right.

Definition 1.2: Residual Count/Conditional Frequencies

Given a dataset X, and a set $\mathcal{S} \subseteq \mathcal{H}$, we say $x \not\succ \mathcal{S}$ if $\nexists q \in \mathcal{S}$ such that $x \succeq q$. We define the conditional or residual count $F_{\mathcal{S}}(p)$ of a prefix p with respect to $\mathcal{S}$ as the sum of all fully specified elements who do not have a parent already in $\mathcal{S}$.

$$F_{\mathcal{S}}(p) = \textstyle\sum_{e \in X \wedge e \succeq p \wedge e \not\succ \mathcal{S}} f_X(e)$$

In the left tree in Figure 1, the set $\mathcal{S}$ is shown by nodes in blue, and the conditional count with respect to $\mathcal{S}$ is written by each node.

Definition 1.3: Level Of A Node/Prefix

he level of a prefix $p \in \mathcal{H}$ is the (minimum) number of applications of $\mathsf{Generalise}$ to reach $*$ i.e.

[5] $\mathsf{Generalise}$ encodes partial order binary relation $R : \mathcal{U} \times \mathcal{U} \to \{0,1\}$, such that $R(x,p) = 1 \iff p = \mathsf{Generalise}(x)$

[6] The HHH literature also considers multi-dimensional hierarchies, where the universe is represented by nodes of a lattice rather than a rooted tree. However, in this work, we focus on single dimensional hierarchies which can be represented by a tree.

$\mathsf{Level}(p) = k \iff * = \mathsf{Generalise}^{(k)}(p)$

For example, let $\mathcal{H}$ be the set of 4-bit bistrings. If we generalise $011*$ three times we get to $*$, so the level of the prefix is 3. The fully specified elements or leaf elements are at level $h = 4$ and $*$ is at level 0. With these definitions in place, we can formally define the concept of a heavy hitter and a hierarchical heavy hitter.

Definition 1.4: Exact Heavy Hitters

For dataset X and threshold $\tau \in \mathbb{R}$, we say prefix p is a heavy hitter (HH) if $f_X(p) > \tau$. The set of heavy hitters of X is $\mathcal{HH} = \{e \in X : f_X(e) \geq \tau\}$.

Definition 1.5: Exact Hierarchical Heavy Hitters

The set of exact hierarchical heavy hitters is defined inductively. Let X denote a dataset drawn from a hierarchy of height h, then
(1) $\mathcal{HHH}_h$ denotes the exact heavy hitters in X.
(2) For any prefix p at level $0 \leq l < h$, let $F_{\mathcal{HHH}_{l+1}}(p)$ be the residual count (Defn. 2) of p given $\mathcal{HHH}_{l+1}$. Then $\mathcal{HHH}_l$ is defined as $\mathcal{HHH}_{l+1} \cup \{p \in \mathsf{Level}(l) : F_{\mathcal{HHH}_{l+1}}(p) \geq \tau\}$
(3) $\mathcal{HHH}_0$ is the set of *exact* hierarchical heavy hitters of X.

Figure 1 illustrates this difference between heavy hitters and hierarchical heavy hitters.

Approximate Hierarchical Heavy Hitters. In this paper, the function f that our algorithm M computes is the hierarchical heavy hitters of a dataset X. By definition, differential privacy restricts us from outputting exact answers or using a deterministic algorithm to compute approximate values. As the output *must* be random, we can no longer output exact counts of the hierarchical heavy hitter problem. Thus, keeping in line with the definitions introduced in [8, 1] we define the task of *approximate heavy hitters*, where the estimates are within some approximation error Δ with high confidence. As we now have noise in the system, we relax the threshold by Δ units, where Δ allows the error to grow larger for larger values (i.e., relative error). Clearly the smaller the value of Δ, the closer we are to the definition of exact hierarchical heavy hitters. The coverage constraint says we want to be conservative and not miss out on potential heavy hitters due to DP noise i.e., prevent false negatives[7]. Our goal is to come up with a theoretical bound on the error, and confirm empirically that the error is small enough for practical use cases as well.

Problem 1 (Private Approximate Hierarchical Heavy Hitters)**.** Let M denote an algorithm that receives as input a multi set X of n fully specified elements from some hierarchical domain $\mathcal{H}$. Fix a public threshold $\tau \in \mathbb{R}$, a confidence parameter $\eta \in (0, 1)$, privacy parameters $\varepsilon \in (0, \log n)$ and $\delta = \mathsf{negl}(n)$. We say the algorithm M correctly finds approximate private hierarchical heavy hitters with relative error (τ, Δ) if it outputs $\mathcal{S} \subset \mathcal{H}$ and approximate counts $\widetilde{f_X}(p)$ such that:

1. **Privacy**: M is (ε, δ)-DP.

2. **Simultaneous Relative Error**: With probability $1 - \eta$ we have $\max_{p \in \mathcal{H}} \left| \frac{f_X(p) - \widetilde{f}_X(p)}{f_X(p)} \right| \leq \frac{\Delta}{\tau}$.

3. **Coverage**: For *any* prefix $p \notin \mathcal{S}$, $F_{\mathcal{S}}(p) \leq \tau - \Delta$ with probability $1 - \eta$, with $F_{\mathcal{S}}(p)$ as defined in Definition 2.

We want to show that in the non-streaming setting, the relative error Δ of our algorithm does not grow linearly with the the height of the hierarchy[8], and is independent of the number of heavy hitters in the hierarchy (as discussed in the introduction above). In the streaming setting we will have to deal with error due to privacy and lack of space. Thus, the streaming version of our problem is the exact same problem with limited space.

[7] We constrain on false negatives instead of false positives to stay aligned with the definitions in prior works. Our results still hold if we constrain on false positives.

[8] As we want to bound the worst case simultaneous relative error for *any* element in the hierarchy, logarithmic dependence on height is unavoidable (via the union bound).

Problem 2 (Streaming Private HHH)**.** The streaming problem is to solve Problem 1 with a constant amount of space $\kappa = O(1)$.

Of course, to prevent the problem from being degenerate, we will assume that the amount of space available is significantly smaller than the size of the stream n or the size of the universe $|\mathcal{H}|$.

B Offline Private Hierarchical Heavy Hitters

In this section, we tackle the offline version of the problem, with no space constraints. Before describing the complete protocol and showing how it solves Problem 1, we provide an intuitive explanation of the techniques used to circumvent the dependence the height of the hierarchy and the number of hierarchical heavy hitters by working through a sequence of attempts. Let $\mathcal{H}$ denote a hierarchy with height h and let X denote a dataset of n fully specified elements drawn from $\mathcal{H}$.

Laplace histograms. Observe that the unconditional frequencies for each prefix of the hierarchy can be estimated by computing h histograms for each level of the hierarchy. For two neighbouring datasets X and X' that differ by a single input only, there is *exactly* one node per level for which the unconditional frequency of the nodes differ by one under X and X'. Thus, a first approach to hierarchical heavy hitter estimation is to release h Laplace histograms by adding noise drawn from a Laplace noise distribution with scale $\frac{h}{\varepsilon}$ to every node in the hierarchy. By the privacy of the Laplace histograms (Fact 1), each level is ε/h-private. As any node contributes to at most h counts, by basic composition and the privacy of the Laplace mechanism, the set of realised counts is $(\varepsilon, 0)$-DP. Then, one could compute hierarchical heavy hitters via post-processing, which does not affect the privacy of the algorithm (Fact 3). As we add noise with scale $\frac{h}{\varepsilon}$ to each node in $\mathcal{H}$, the DP error per node scales linearly in the height of the hierarchy. Furthermore, if we wanted to upper bound the simultaneous error over all prefixes in the hierarchy, then we need to apply the union bound over all nodes of the hierarchy. As the number of nodes is exponential in the height of the hierarchy, the relative estimation error scales $\mathtt{poly}(h)$.

Stability Histograms. One solution to circumventing such a union bound over all the elements of the universe when the size of the universe is very large is to use stability histograms [21, 12, 22] instead of Laplace histograms. For fixed privacy parameters $\varepsilon \in (0, \log n)$ and $\delta = \mathsf{negl}(n)$, the key intuition behind stability histograms is that if we only released private estimates (using the Laplace mechanism) for nodes with "large enough" non-zero frequency, the simultaneous error bound improves from $\log(|\mathcal{H}|)$ to $\log(n) < \log(1/\delta)$ (as there are at most n elements in X). This is advantageous when $|\mathcal{H}| \gg n$ meaning that many nodes have zero frequency. For such nodes, we incur no error at all (as the algorithm ignores them).

However, this technique cannot achieve pure DP as was possible in the Laplace histogram case. Observe that if X' contains an element x' that is not present in X (we call such an x' isolated), an adversary can perfectly distinguish between X and X' if the count of this element, albeit noisy, appeared in the output (as when processing X, nodes representing generalisations of x' would have frequency 0, and would be ignored). Nevertheless, since stability histograms only output counts for elements with "large" counts, and as the frequency of such an isolated element x' is 1 (from the definition of neighbouring datasets), we can set the frequency threshold for "large enough" such that with probability at least $1 - \delta$ the isolated element will never show up in the final output. This is sufficient to achieve (ε, δ)-DP. Over the whole tree representing the hierarchy, the case where x' distinguishes X and X' can occur at at most h levels of $\mathcal{H}$. Thus, if we output stability histograms with $\delta' = \delta/h$ at each level then, by basic composition, the final output does not contain any isolated elements with probability at least $1 - \delta$. In this way, stability histograms allow us to circumvent the union bound over all the nodes in $\mathcal{H}$, at the cost of going from pure to approximate DP. It does not however, circumvent the issue discussed above where the scale of the DP noise is $\frac{h}{\varepsilon}$.

DP Counting On Trees. One approach to resolving this issue is to make repeated use of the Above-Threshold Algorithm by Dwork et al. [23]. Given a hierarchy $\mathcal{H}$ then, by definition, there can be at most $c = \frac{n}{\tau - \Delta}$ hierarchical heavy hitters. Observe that c is a constant that depends only on τ and Δ, and is independent of the height of the hierarchy h. When the height of the hierarchy h is large and $c \ll h$, we would incur lower per-node DP error if the scale of the noise were $\frac{c}{\varepsilon}$ instead of $\frac{h}{\varepsilon}$. Ghazi et al. [2] observed this fact and proposed an algorithm that traverses the tree bottom up. At each level of the tree, their algorithm inspects only one node, the one with maximal unconditional count for that level. There are at

Algorithm 1: Non-Streaming DP-HHH Detection

```
Data: Data $X$ of size $n$ over hierarchy $\mathcal{H}$ with height $h$, Privacy parameter $\varepsilon \in (0, \log n)$, $\delta = \mathsf{negl}(n)$,
      Threshold $\tau > 0$
Result: Approximate Hierarchical Heavy Hitters $\mathcal{S}$, and, their approximate frequencies $\{\widetilde{f_X}(p)\}_{p \in \mathcal{S}}$.
If $\tau < \frac{8}{\varepsilon}\log(2h/\delta) + 1$, output $(\{\}, 0)$ and end program ;
$\gamma \xleftarrow{\$} \mathsf{Laplace}(\frac{2}{\varepsilon})$;
$\mathcal{S} = \{\}$;
for $i = h, \ldots, 1$ do
    $\mathcal{A}_i = \{p \in \mathcal{H} | \mathsf{Level}(p) = i\}$ ;
    for $p \in \mathcal{A}_i$ do
        if $F_{\mathcal{S}}(p) = 0$ then
            continue to next iteration;
        $w_p \xleftarrow{\$} \mathsf{Laplace}(\frac{4}{\varepsilon})$;
        if $F_{\mathcal{S}}(p) + w_p + \gamma \geq \tau$ then
            $\mathcal{S} = \mathcal{S} \cup \{p\}$;
            $\widetilde{F}_{\mathcal{S}}(p) = F_{\mathcal{S}}(p) + \mathsf{Laplace}(\frac{4}{\varepsilon})$
    Output $\mathcal{S}$ and $\{\widetilde{F}_{\mathcal{S}}(p)\}_{p \in \mathcal{S}}$.
```

most h such nodes (one per level). However, even in the worst case, at most c out of h of these nodes can be conditionally heavy. Thus, we have h counting queries, out of which c are conditionally heavy. Their solution uses the Sparse Vector Technique (SVT) by composing Above-threshold h times, but they pay only for the c conditionally heavy nodes. Once c levels have been identified, the algorithm prunes the tree, and restricts it to just the c levels with at least one conditionally heavy node. Then it estimates the counts of the pruned tree by solving c instances of the private stability histogram estimation problem[9] with privacy budget ε/c. As commented in the introduction, although this is an asymptotic improvement, it is very hard to imagine cases where $c \ll h$. Thus, the simple Laplace stability histograms with composition error h would outperform the algorithm by Ghazi et al. [2] in almost all feasible situations.

Our Approach. Our algorithm is based on the following critical observation. Although it is true that there are at most c conditionally heavy levels in the tree, *only one* of the conditionally heavy nodes is really influenced by x'. For the remainder of the heavy nodes, there is no difference in the conditional frequencies of X and X'. It is this structure that we exploit to improve error our guarantees. In prior work, Kaplan et al. [24] show that there are instances where the composition error of the SVT algorithm is suboptimal, even when dealing with a stream of adaptively chosen queries by an unbounded adversary. Instead, they propose a general algorithm (called Threshold Monitor) where DP error of a query scales linearly by a constant $\frac{75\varepsilon(k+1)}{\log 1/\delta}$, where k represents the number of times any data point can contribute to a counting query being heavy. Note that both the SVT algorithm and Threshold monitor are general and accommodate a stream of adaptively chosen counting queries with no structure. In estimating HHH in the non-streaming setting with central differential privacy, we can pre-select the order of queries to be bottom up (leveraging monotonicity and limited influence of any node), and we do not have to deal with adaptive queries. Thus, in the case of hierarchical heavy hitter estimation, paying a price of c/ε of SVT or the larger constants of Threshold Monitor for adaptivity in composition is wasteful. Instead, we can use a simpler algorithm, and with more specialized privacy analysis we show that DP noise per node with scales $\frac{4}{\varepsilon}$ instead of $\frac{c}{\varepsilon}$, and thus is independent of *both* the height of the hierarchy and the number of heavy hitters in $\mathcal{H}$. Note that this matches the constants for non-hierarchical private algorithm for detecting a single heavy query from a stream [12]. Thus, this is the best one can hope to do. We formally state our claim and provide a sketch of the important steps of the proof below; the full privacy proof is given in Appendix E.

Theorem B.1 (Privacy)**.** The DP-Hierarchical Heavy Hitters algorithm described in Algorithm 1 is (ε, δ)-DP.

[9] While Ghazi et al. do not explicitly refer to stability histograms, their use of bounded truncated Laplace noise is equivalent. In the case of bounded noise, the union bound is affected by the size of the support of the truncated noise distribution, which is upper bounded by $O(1/\delta)$. For stability histograms, it is affected by the number of elements which is upper bounded by $1/\delta$.

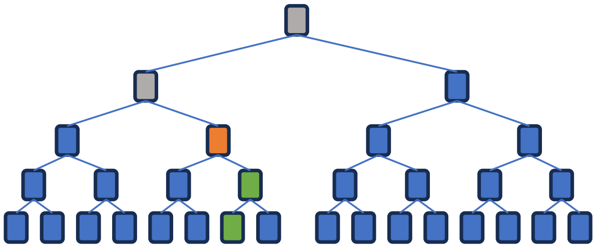

Figure 2: The figure above describes the partitioning of $\mathcal{H}$ assuming the neighbouring element x' is *not* isolated. The nodes in blue ($\mathcal{I}_{\text{Unrelated}}$) denote all the nodes whose residual frequencies are unaffected by the presence of x'. The nodes in green are the prefixes whose conditional frequencies include x', but their noisy conditional frequency (Line 10 of Algorithm 1) is below τ. The node in orange is referred to as the inflection node. It is the first prefix of x' whose noisy count crosses the threshold. Together, the green and orange nodes form ($\mathcal{I}_{\text{Active}}$). For all nodes that are generalisations of the inflection node (shown in grey), their conditional count excludes any influence of x', from the definition of conditional count ($\mathcal{I}_{\text{After}}$).

Proof Sketch. Let X and $X' = X \cup \{x'\}$ denote two neighbouring datasets. There are two cases to consider. The first case is where the neighbouring element x' has a count one. In this case it is easy to show that h instances of stability histograms gives us a $(0, \delta/h)$ private algorithm. In the other case, the hierarchical universe can be partitioned into three categories, which we denote with sets $\mathcal{I}_{\text{Unrelated}}, \mathcal{I}_{\text{Active}}$ and $\mathcal{I}_{\text{After}}$, as described in Figure 2. For nodes in $\mathcal{I}_{\text{Unrelated}}$ we do not pay for privacy as even the unconditional counts in X and X' are exactly the same. For the nodes in $\mathcal{I}_{\text{After}}$, the influence of the neighbouring element x' in calculating residual counts has already been removed from the data (as the node denoted in orange, called the inflection node is in the output already). Thus, we do not need to pay for privacy when estimating residual counts for these nodes. Finally, we show that for nodes in $\mathcal{I}_{\text{Active}}$, we only need to pay once for the inflection node and once for all the remaining nodes (denoted in green). The argument follows from analysis similar to the above-threshold algorithm [23]. Algorithm 1 is a standalone succinct description of the HHH estimation protocol in the non-streaming setting: starting from the leaf level, we compute a noisy estimate of the residual count of each non-zero node, and add it to the output (with fresh noise) if this is above the threshold. See full proof on page 17.

Theorem B.2 (Coverage And Error). Let $\mathcal{S}$ denote the set of hierarchical heavy hitters selected by Algorithm 1. Then with probability $1-\eta$, for all $p \notin \mathcal{S}$, $F_{\mathcal{S}}(p) \leq \tau - \Delta$. Additionally, for *all* $p \in \mathcal{H}$ with probability $1-\eta$ we have $\max_{p\in\mathcal{H}} \left|\frac{f_X(p) - \tilde{f}_X(p)}{f_X(p)}\right| \leq \frac{\Delta}{\tau}$, where

$$\Delta = \frac{8}{\varepsilon}\left(\log\left[\frac{1}{\delta}\right] + \log\left[\frac{2h}{\eta}\right]\right)$$

The analysis for coverage and error follows from the tail bounds of the Laplace distribution and the union bound. See proof on page 20.

C Streaming Private Hierarchical Heavy Hitters

In the previous section we proved that the scale of the DP noise per query scaled only by a small constant factor that was independent of the height of the hierarchy and the number of hierarchical heavy hitters in the dataset. A critical factor to being able to bypass composition was that there was no space limitation, and we could store the *exact* conditional count for *every* query in memory. This meant that we could eventually remove the contribution of the neighbouring element x' completely (restricting the privacy loss entirely to the green and orange nodes in Figure 2). Thus for a large majority of queries, neighboring inputs X and X' were treated identically.

In the streaming setting, we do not have the luxury of storing exact counts. Thus, we will need to use some streaming data structure [8, 1, 18] in order to approximate residual frequencies. In this work, we use one

Algorithm 2: Insertion Operation For A Single MG Sketch

```
Data: Next data $x \in \mathcal{H}$, increment $v > 0$.
Result: Inserts $x$ into SS sketch $\mathcal{T}$ and updates its count by $v$.
Parameters: Number of counters $\kappa$
if $x \in \mathcal{T}$ then
    $C[x] = C[x] + v$ ;                                    // $C[x]$ is the count of $x$ in $\mathcal{T}$
else
    if $C[i] \geq v \quad \forall i \in \mathcal{T}$ then
        $C[i] = C[i] - v \quad \forall i \in \mathcal{T}$;
    else
        Let $\tilde{y} = \arg\min_{y \in \mathcal{T}} C[y]$ ;
        $\mathcal{T} = (\mathcal{T} \setminus \{\tilde{y}\}) \cup \{x\}$;
        $C[x] = v$ ;
```

Misra Gries (MG) sketch with κ counters per level of the hierarchy. Thus the total space used is $O(\kappa h)$. We choose the MG sketch for two reasons. Firstly, Agarwal et al. [25] showed that the MG sketch is isomorphic to the Space Saving algorithm, and Mitzenmacher et al. [8] show that the space saving algorithm is optimal for non-private hierarchical heavy hitter estimation. Thus, if we replace SS with MG in the non private HHH estimation problem, we retain the same optimality results. Secondly, with the MG sketch we leverage the structure in the output to circumvent the composition bounds due to approximation. The main bottleneck in approximation algorithms is that the approximated function might have large global sensitivity. Indeed Chan et al. [18] show that the MG sketch has global sensitivity $\Delta_G = \kappa$. This implies that if we naively used noise scaled by the sensitivity of the function, the DP error grows as the streaming error drops. However we are able to leverage the critical observation made by Lebeda and Tetek [17, Lemma 5], who show that if the global sensitivity of the MG sketch is high, then counts of each counter after processing neighbouring streams are highly correlated. Just like in Section B where we made use of monotonicity of residual queries, we will use this correlation to circumvent composition. Our techniques are related to the more general observation that structure has often been used to bypass composition in the privacy community [23, 26, 24, 27]. In summary, to construct our HHH estimation algorithm in the streaming setting, we first modify the Mitzenmacher et al. [8] HHH algorithm to use the modified MG sketch from Lebeda and Tetek [17] instead of the SpaceSaving algorithm. Then we repeatedly leverage the structure of the output to bound the DP error to be independent of the available space. Note, we cannot avoid the approximation error introduced due to lack of space, regardless of privacy. In the streaming setting, our contribution is to show that the DP error for HHH is not affected by this approximation parameter κ. Before describing our algorithm, we first provide an overview of the proof behind how we keep the DP error independent of the number of counters, and why the tricks used in the non-streaming section to make the noise independent of the height of the hierarchy *no longer apply*.

C.1 Technical Overview

Our main insight in the non-streaming setting was that the DP noise could be independent of the height of the hierarchy h. Unfortunately, in the streaming setting, this claim no longer holds true. To fully describe why, we first re-state a lemma by Lebeda and Tetek [17, Lemma 5] which formally describes the observed outcomes when a MG sketch processes neighbouring input streams.

Lemma 1. [17, Lemma 5 (re-stated)] Let $X = X' \cup \{x\}$. Let $(\mathcal{T}, C) \leftarrow \mathsf{MG}(\kappa, X)$ and $(\mathcal{T}', C') \leftarrow \mathsf{MG}(\kappa, X')$ be the output of Algorithm 2 with inputs X and X'. Then, $|\mathcal{T} \cup \mathcal{T}'| \geq \kappa - 2$; for all $x \notin \mathcal{T} \cup \mathcal{T}'$, $C[x] \leq 1$ and $C'[x] \leq 1$; and exactly one of the following is true:

1. $\exists i \in \mathcal{T}$, such that $C[i] = C'[i] + 1$, and $\forall j \neq i : C[j] = C'[j]$.

2. $\forall i \in \mathcal{T}' : C[i] = C'[i] - 1$, and $C'[j] = 0$ for $j \notin \mathcal{T}'$.

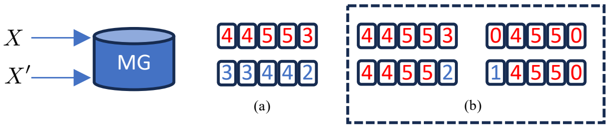

Figure 3: The figure above illustrates the implications of Lemma 1. There are two possible configurations after processing neighbouring streams. One configuration, depicted by figure (a), is that every count in one sketch is different from the count in the other sketch by one unit in the same direction. The other outcome is that either exactly one counter is different for all counters that are non-zero in both sketches. Additionally, when in configurations depicted by Figure(b), there can be at most 2 elements per sketch that are not in the other sketch. The count of these elements when present is always 1. The second outcome, denoted by figure(b) is identical to the configuration discussed for stability histograms in the previous section, where each sketch has at most a single count that is different.

In the lemma above, $(\mathcal{T}, C)$ and $(\mathcal{T}', C')$ denote the output of the MG algorithm (Algorithm 2) on the two neighbouring streams X and X'. The lemma states that there can be at most 2 elements that are in $\mathcal{T}$, but not in $\mathcal{T}'$, and vice versa. We will refer to these as *isolated elements* (this is distinct from the isolated prefixes from the previous section), and if they appear in the final output, an adversary can always tell if an output was generated by X or X'. Furthermore, the count of an isolated element must always be at most 1. Thus once again, we can use the thresholding trick from stability histograms to suppress isolated elements with high probability. The threshold for suppression is set a little higher, as there are two isolated elements instead of one. Along with the above statements, after processing neighbouring data streams, Lemma 1 also states that the resulting sketches could be in one of two scenarios illustrated in Figure 3. In the first scenario, shown in Figure 3b, we have two histograms with κ bins that differ in at most two locations by a count of 1. All other bins have the exact same values across both sketches. We can easily release private versions of these histograms by simply applying the Laplace mechanism with suppression of small values. The other scenario, described by Figure 3a, appears more problematic at first glance. We have that the counts across two outputs C and C' in the two sketches all differ by the *same* amount across *all* the bins. This is undesirable in two ways. First, now the global sensitivity of the approximated counts is κ, so it appears that the DP noise will need to scale linearly in κ (which can be a large constant). Secondly, notice that we can no longer restrict the influence of differing nodes in $\mathcal{T}'$ to a single hierarchical heavy hitter like we did in the illustration given in Figure 2. The neighbouring elements are now spread across all the bins, and they influence *multiple* hierarchical heavy hitters. Furthermore, as the MG sketch often under-approximates the true count of an element (Lemma 2), we cannot guarantee that we have removed the influence of a neighbouring element higher up in the tree by removing a descendant lower in the tree. This means we cannot restrict the influence of neighbouring elements and partition the tree like before to avoid paying the privacy loss due to composition for h levels of the hierarchy. In other words, it seems challenging to circumvent composition bounds due to the height when using the MG sketch.

Although we cannot remove the dependence on the height of the hierarchy in the streaming setting, inspired by the observations made by Lebeda and Tetek [17] and Dwork et al. [23], we are still able to remove the dependence on κ. We show that, even in the scenario depicted by Figure 4a, the DP noise is actually independent of the number of counters in the sketch. This might appear unintuitive at first glance, as the global sensitivity is still κ. To gain intuition towards understanding why, we first review the privacy proof for nodes in $\mathcal{I}_{\text{Active}}$ (green and orange nodes in Figure 2) in the non-streaming algorithm. The main observation there was that before we see the inflection node (orange node in Figure 2), we could group all the small valued queries for which the output was $\perp$, and treat them as *one* query, thus paying for them only *once*. We could do so because the event that all of those queries being small is equivalent to saying that the maximum of all the those queries is small. Given two neighbouring datasets, we might have $t-1$ queries, each with sensitivity 1, and thus a total sensitivity of $t-1$, but the max of these queries is still a single query with sensitivity 1. Thus, we have a *single* equivalent query that captures the event described by $t-1$ queries with output $\perp$. A similar reframing also applies to the sketches produced by neighbouring streams. Observe that despite κ bins being different in C and C', the direction in which they are different and the amount by which they are different is the same for *all* bins. Thus, we have a guarantee that, if one of the bins in C and C' is

```
Algorithm 3: Private Release
    Data: Data stream $X$, Privacy parameters $\varepsilon$ and $\delta$
    Result: Privatised MG sketches $(\overline{\mathsf{MG}_1}, \dots, \overline{\mathsf{MG}_h})$
 1  Given a dataset $X$, construct $h$ sketches $(\mathsf{MG}_1, \dots, \mathsf{MG}_h)$ by running Algorithm 2 for each $x \in X$ and
     every generalisation of $x$.
 2  for $l \in [h, h-1, \dots, 1]$ do
 3      $\gamma_l \leftarrow\!\$\ \mathsf{Laplace}(\frac{2h}{\varepsilon})$;
 4      Let $(\mathcal{T}_l, C_l) = \mathsf{MG}_l$;
 5      for $i \in \mathcal{T}_l$ do
 6          $w_i \leftarrow\!\$\ \mathsf{Laplace}(4h/\varepsilon)$;
 7          if $C_l[x] + \gamma_l + w_i > 1 + \frac{6h}{\varepsilon}\log(3h/\delta)$ then
 8              $C_l[x] = C_l[x] + \mathsf{Laplace}(4h/\varepsilon)$ ;
 9          else
10              $C_l[x] = 0$ ;
11      Set $\overline{\mathsf{MG}_l} = (\mathcal{T}_l, C_l)$;
12  Output $(\overline{\mathsf{MG}_1}, \dots, \overline{\mathsf{MG}_h})$
```

different, *all* bins are different, and importantly in the exact *same* way. There is actually just one degree of freedom, despite there being κ different counts to consider. Once one bin differs by one, *all* bins differ by one (i.e. the counts are correlated). In the Above threshold (a single invocation of the SVT algorithm), we looked at the max query, and in this case we can look at any one query (say the first one, which reveals everything about the other ones). The effect is equivalent. In the proof for Theorem C.1 we formally show how we can handle all κ counts being different with just one sample of noise, as if we had just a single query.

Remark. *Lebeda and Tetek [17, Lemma 6] make a similar claim to use one sample to cover all bins. Although the final claim is correct, there is a minor issue in their proof. In their proof, the authors define the function $g : \mathbb{R} \to \mathbb{R}^k$ such that $g(a) = a1^k$. Thus, $g^{-1}(\vec{x})$ is defined only if all coordinates of $\vec{x}$ are the same. Lemma 1 does not guarantee that all the counts are the same, only that two neighbouring differ by the same amount in the same direction. However, their proof [17, Line 4, Page 6] relies on inverting $g^{-1}(\vec{x})$ for general $x \in \mathbb{R}^k$, which is undefined. In our analysis, there is no need to define such a function g and we can obtain our results using the above observations.*

Algorithm 4 describes our algorithm for computing hierarchical private heavy hitters. First, we compute noisy heavy hitters at each level of the hierarchy using Algorithm 3. We show that the output of Algorithm 3 is private. Then we conservatively post-process this private output to get our desired result.

Remark. *For a fixed level ℓ, Algorithm 3 is very close to [17, Algorithm 2], with an essential alteration. In Line 8 of the algorithm, we generate a fresh batch of randomness independent of thresholding operation to release approximate counts. The construction described by Lebeda and Tetek [17] re-uses the thresholding noise and it is therefore not differentially private. The intuition for this is that, in Lines 7-8, we release information about x twice. Once with the threshold, and then again with noisy count. Once the privacy adversary sees a noisy threshold, it gains information about the value of γ_l, and we can no longer use one sample to cover for all bins. More simply put, it violates composition bounds, as two releases require two batches of randomness to cover this leakage. Lyu et al. [28, Algorithm 3 and Figure 2] formally show how not using a fresh batch of randomness is not differentially private. Their proof is based on the analysis of SVT, but as we argue above, the same analysis applies here (as the two follow essentially the same procedure).*

Theorem C.1 (Privacy). Algorithm 3 is (ε, δ)-DP.

See full proof on page 21.

Corollary. *Algorithm 4 is (ε, δ)-DP.*

```
Algorithm 4: Final Algorithm
  Data: Data $X$, Privacy parameter $\varepsilon \in (0, \log n)$, $\delta = o(1/n)$, Threshold $\tau > 0$, Confidence Parameter
        $\eta \in (0, 1/2)$
  Result: Approximate Hierarchical Heavy Hitters $\mathcal{S}$ and their approximate frequencies
          $\widetilde{f_X}(\mathcal{S}) = \{\widetilde{f_X}(p)\}_{p \in \mathcal{S}}$
  Parameters: Number of counters $\kappa$, Height of hierarchy $h$.
  Run Algorithm 3 with input $X$, Threshold $\tau$, and privacy parameters $(\varepsilon, \delta)$ to get outputs
    $(\overline{\mathsf{MG}}_1, \dots, \overline{\mathsf{MG}}_h)$ ;
  $\Delta_1 = \left(1 + \frac{4h \log[6h/\delta]}{\varepsilon}\right) + \frac{n}{\kappa+1} + \left(\frac{8h}{\varepsilon} \log[\frac{2\kappa h}{\eta}]\right)$ ;
  $\Delta_2 = \left(1 + \frac{4h \log[6h/\delta]}{\varepsilon}\right) + \left(\frac{8h}{\varepsilon} \log[\frac{2\kappa h}{\eta}]\right)$ ;
  for $l \in [h, h-1, \dots, 1]$ do
      Let $(\mathcal{T}_l, C_l) = \overline{\mathsf{MG}}_l$;
      for $e \in \mathcal{T}_l$ do
          if $C_l[e] + \Delta_1 > \tau - \Delta_1$ then
              $\mathcal{S} = \mathcal{S} \cup \{e\}$;
              $\widetilde{f_X}(e) = C_l[e]$ ;                                   // Noisy Estimate
              for $p \in \mathsf{Generalise}(e)$ do
                  Let $i = \mathsf{Level}(p)$;
                  Let $(\mathcal{T}_i, C_i) = \mathsf{MG}_i$;
                  if $p \in \mathcal{T}i$ then
                      // conservatively remove residual
                      $C_i[p] = C_i[p] - (C_l[e] - \Delta_2)$
  Output $(\mathcal{S}, \widetilde{f_X}(\mathcal{S}))$
```

Algorithm 4 is just post-processing the (ε, δ)-DP output of Algorithm 3, so by Fact 3 it is also (ε, δ)-DP.

Theorem C.2 (Error)**.** For all $p \in \mathcal{S}$, we have with probability $1 - \eta$,

$$|\widetilde{f_X}(p) - f_X(p)| \leq \Delta$$

where

$$\Delta = \left(1 + \frac{6h \log(3h/\delta)}{\varepsilon}\right) + \frac{n}{\kappa + 1} + \left(\frac{8h}{\varepsilon} \log\left(\frac{2\kappa h}{\eta}\right)\right)$$

See proof on page 22.

We note that the dependence on h here is not optimal. We have used basic composition to show a linear dependence on h, in order to keep the development clear. However, it is possible to show an improved dependence on $\sqrt{h}$, by invoking more advanced composition theorems [19, 29]. Since we assume that h is relatively small in practice, we don't expand on this point in this presentation. Note that as we cannot avoid the composition error due to the height of the hierarchy, we can directly estimate the unconditional frequencies of a node p by looking at the noisy count of $C_{\mathsf{Level}(p)}[p]$. However, this means that we do not obtain relative error guarantees in the streaming case: all prefixes incur noise of the same magnitude. We leave it open to show whether this gap is provably unavoidable or can be surmounted using different techniques.

Theorem C.3 (Coverage)**.** For all $p \notin \mathcal{S}$, with probability $1 - \eta$, $F_{\mathcal{S}}(p) \leq \tau - \Delta$.

See proof on page 23.

D Differential Privacy Definitions

In this section we introduce concepts we need from differential privacy to understand private heavy hitters.

Definition 4.1: Neighbouring Datasets

et X and X' be a multi sets of elements picked from some (possibly hierarchical) domain $\mathcal{H}$. X and X' are said to be neighbouring, (denoted as $X \sim X'$), if they differ by one element only, i.e., $X' = X \cup \{x'\}$, or vice-versa.

Definition 4.2: Differential Privacy (DP)

ix some function f that maps a set of elements from a hierarchical domain $\mathcal{H}$ to some range $\mathcal{Y}$. Fix $n \in \mathbb{N}$. Let $X \in \mathcal{H}^n$ and $X' \in \mathcal{H}^{n+1}$ denote *any* pair of neighbouring datasets. For $\varepsilon > 0$ and $\delta = \mathsf{negl}(n)$, where $\mathsf{negl}(\cdot)$ is a negligible function in n, we say a random algorithm M computes f with (ε, δ)-differential privacy *if and only if* and for *all* $\mathcal{A} \subseteq \mathcal{Y}$,

$$\Pr[\mathsf{M}(X, f) \in \mathcal{A}] \leq \exp(\varepsilon) \Pr[\mathsf{M}(X', f) \in \mathcal{A}] + \delta$$

The special case of (ε, δ)-DP with $\delta = 0$ is referred to as pure DP, whereas $\delta > 0$ is known as approximate DP. A standard approach to obtain DP is to add noise proportional to the global sensitivity of the function being evaluated.

Definition 4.3: Global Sensitivity

Given any function f that maps a set of elements from a hierarchical domain $\mathcal{H}$ to $\mathbb{R}$, we define the global sensitivity $\Delta_G(f)$ of f as $\Delta_G(f) := \max_{(X,\, X')} ||f(X) - f(X')||_1$, where the maximum is taken over *any* pair of neighbouring datasets X, X'.

It is well known that the global sensitivity of a counting query, such as the queries defined in Definitions 1 and 2 is 1. The following facts about differential privacy can be found in any introductory textbook on differential privacy [19].

Fact 1 (Laplace Histograms)**.** Let f be a function that maps elements from some domain to a subset of $\mathbb{R}^d$, that has global sensitivity Δ_G. Then the Laplace mechanism M defined as $\mathsf{M}(X) = f(X) + (Y_1, \ldots, Y_d)$ where each $Y_1, \ldots, Y_d \leftarrow\!\!\$ \ \mathsf{Laplace}(\Delta_G/\varepsilon)$ is ε-DP.

Fact 2 (Basic Composition)**.** Let M_1 and M_2 be a $(\varepsilon_1, \delta_1)$-DP and $(\varepsilon_2, \delta_2)$-DP algorithm respectively. $\mathsf{M}(X) = (\mathsf{M}_1(X), \mathsf{M}_2(X))$ is $\Big((\varepsilon_1 + \varepsilon_2), (\delta_1 + \delta_2)\Big)$-DP.

Fact 3 (Post Processing)**.** Let A_1 be an (ε, δ)-DP algorithm and A_2 be a (possibly randomised) post-processing algorithm. Then the algorithm $\mathsf{A}(x) = \mathsf{A}_2(\mathsf{A}_1(x))$ is still an (ε, δ)-DP algorithm.

E Deferred Proofs

Theorem B.1 (Privacy)**.** The DP-Hierarchical Heavy Hitters algorithm described in Algorithm 1 is (ε, δ)-DP.

Proof of Theorem B.1. Let X be a database of n fully specified elements from some hierarchical domain $\mathcal{H}$ with height h. Let $m = |\mathcal{H}|$ denote the number of elements in the hierarchy. Let $X' = X \cup \{x'\}$. Fix privacy parameters $\varepsilon \in (0, \log n)$, $\delta \in o(n)$, and let τ denote the public threshold. Let $\mathcal{S}$ be any general output of the algorithm. As $\mathcal{S}$ is general, our goal is to show that with probability $1-\delta$, $\Pr[\mathsf{M}(X) = \mathcal{S}] \leq e^{\varepsilon} \Pr[\mathsf{M}(X') = \mathcal{S}]$. Our proof proceeds by breaking this probability over a large output space into smaller events. We can condition that the outputs on X and X' are the same prior to each event we analyze — if not, then the outputs already differ, and this contribution to the probability has already been accounted for. We denote each element in $\mathcal{H}$ with integers $\{1, 2, \ldots, m\}$ based on the order[10] in which Algorithm 1 processes them

[10] The algorithm output be the same if we traverse the tree in post order. The key point is we always process all the descendants of an element before processing any element.

(we traverse the tree bottom up in level order). For an element $i \in \mathcal{H}$, let $a_i \in \mathbb{R}$ denote i's noisy residual count if $i \in \mathcal{S}$ (Line 12). If $i \notin \mathcal{S}$, then set $a_i = \bot$. Subsequently, we use the shorthand $a_i = \top$ to mean $a_i \neq \bot$. Thus the vector $\vec{a} = (a_1, \ldots, a_m)$ completely describes the output of the algorithm for any general $\mathcal{S}$. Let β_i denote the random variable that describes our algorithm's output for element i when X is used as the input. Similarly, let β'_i denote the random variable that describes the algorithm output when X' is used as the input. Then using the notation described above $\Pr[\mathsf{M}(X) = \mathcal{S}] = \prod_{i=1}^{m} \Pr\left[\beta_i = a_i | \vec{\beta}_{-i} = \vec{a}_{-i}\right]$, where $\vec{\beta}_{-i} = (\beta_1 \ldots, \beta_{i-1})$ and $\vec{a}_{-i} = (a_1, \ldots, a_{i-1})$. We define the *inflection node* $u^\star \in \mathcal{H}$, to be the first generalisation of x' that is included in the output as a hierarchical heavy hitter. That is $\nexists q \in \mathcal{S}$ such that $q \succ u^\star$. We denote with $i^\star = \mathsf{Level}(u^\star)$. In Figure 2 such a node is depicted in orange. If such a node does not exist i.e., no generalisation of x' is included in $\mathcal{S}$, then set $i^\star = 0$ (an imaginary level above the root of the tree). Given $u^\star$, observe that we can partition $\mathcal{H}$ into 3 sets.

1. $\mathcal{I}_{\text{Unrelated}} = \{v \in \mathcal{H} | x' \nsucceq v\}$ (shown in blue in Figure 2)
2. $\mathcal{I}_{\text{After}} = \{v \in \mathcal{H} | u^\star \succ v\}$ (shown in grey in Figure 2).
3. $\mathcal{I}_{\text{Active}} = \mathcal{H} \setminus (\mathcal{I}_{\text{Unrelated}} \cup \mathcal{I}_{\text{After}})$ (denoted by all nodes that are green and orange in Figure 2).

If $i^\star = 0$, we set $\mathcal{I}_{\text{Active}} = \{q : x' \succeq q\}$, and $\mathcal{I}_{\text{After}} = \{\}$. Note that every output $\mathcal{S}$ entails some $u^\star$, which defines the above partitioning of the $\mathcal{H}$. Let $p(\beta_i, a_i)$ denote shorthand for $\Pr\left[\beta_i = a_i | \vec{\beta}_{-i} = \vec{a}_{-i}\right]$. Then, given $\mathcal{S}$, we have

$$\prod_{i=1}^{m} p(\beta_i, a_i) = \prod_{i \in \mathcal{I}_{\text{Unrelated}}} p(\beta_i, a_i) \prod_{i \in \mathcal{I}_{\text{Active}}} p(\beta_i, a_i) \prod_{i \in \mathcal{I}_{\text{After}}} p(\beta_i, a_i) \tag{1}$$

Unrelated Nodes. Observe that for any $p \in \mathcal{I}_{\text{Unrelated}}$, we have $f_X(p) = f_{X'}(p)$. Also observe for any $q \succ p$, we also have $q \in \mathcal{I}_{\text{Unrelated}}$. This implies that the residual counts (Definition 2) of p are the same under X and X' i.e., $F_{\mathcal{S}}(p) = F'_{\mathcal{S}}(p)$. In other words, if we restrict to just these nodes X and X' are exactly the same. Thus for any $i \in \mathcal{I}_{\text{Unrelated}}$,

$$\begin{aligned}\Pr\left[\beta_i = \top | \vec{\beta}_{-i} = \vec{a}_{-i}\right] &= \Pr\left[F_{\mathcal{S}}(i) + w_i + \gamma \geq \tau\right] \\ &= \Pr\left[F'_{\mathcal{S}}(i) + w_i + \gamma \geq \tau\right] \\ &= \Pr\left[\beta'_i = \top | \vec{\beta}'_{-i} = \vec{a}_{-i}\right]\end{aligned}$$

After Nodes. A similar observation can be made about any $i \in \mathcal{I}_{\text{After}}$. First observe that given any $\mathcal{S}$, if $i^\star = 0$, then $|\mathcal{I}_{\text{After}}| = 0$, and we do not have to deal with this partition at all. Assume that $|\mathcal{I}_{\text{After}}| \geq 1$. This implies there is some node $u^\star \in \mathcal{S}$ that generalises to nodes in $\mathcal{I}_{\text{After}}$. By its definition, for all $p \in \mathcal{I}_{\text{After}}$, $F'_{\mathcal{S}}(p)$ does not contain the count for x'. This is true because $u^\star \in \mathcal{S}$, and $u^\star$ is the first generalisation of x' that is included in $\mathcal{S}$. So $F'_{\mathcal{S}}(u^\star) = f_{X'}(u^\star)$. This implies that when computing residual counts for a node $i \in \mathcal{I}_{\text{After}}$, it is the same as if x' never existed at all. Thus once again we have for all $i \in \mathcal{I}_{\text{After}}$, $F_{\mathcal{S}}(i) = F'_{\mathcal{S}}(i)$, and using the argument above, $\Pr\left[\beta_i = \top | \vec{\beta}_{-i} = \vec{a}_{-i}\right] = \Pr\left[\beta'_i = \top | \vec{\beta}'_{-i} = \vec{a}_{-i}\right]$.

Active Nodes. The above arguments have shown that the mechanism output distribution is identical for $\mathcal{I}_{\text{Unrelated}}$ and $\mathcal{I}_{\text{After}}$. This means that the entire loss in privacy comes from $\mathcal{I}_{\text{Active}}$, which is a subset of nodes on the path from x' to the root. Notice that $|\mathcal{I}_{\text{Active}}|$ could be as large as h, so it might appear that we will have to pay for composition h times. However, we show that this is not true: with probability $1 - \delta$, we only need to pay twice, once for all nodes below $u^\star$, and once for $u^\star$. Define $t = |\mathcal{I}_{\text{Active}}| = h - i^\star$. Observe that for any prefix $p \in \mathcal{I}_{\text{Active}}$, it may be in one of the two scenarios.

Case I (Isolated Prefix): The prefix p is isolated i.e., $f_X(p) = 0$ but $f_{X'}(p) = 1$ (shown by the bottom image of Figure 4a). Such an event can happen at most h times at each level of the tree (as illustrated in the right of Figure 4b). Now in this regime, if $\beta'_p \neq \bot$, then one could perfectly distinguish between X and X' (as $\beta_p = \bot$ always, by Line 8 of Algorithm 1). In order for $\beta'_p = a_p$, where $a_p \neq \bot$, we need the noise introduced for privacy (Line 10 in Algorithm 1) to push the residual count of p above $\tau > 1 + \frac{8}{\varepsilon}\log(2h/\delta)$

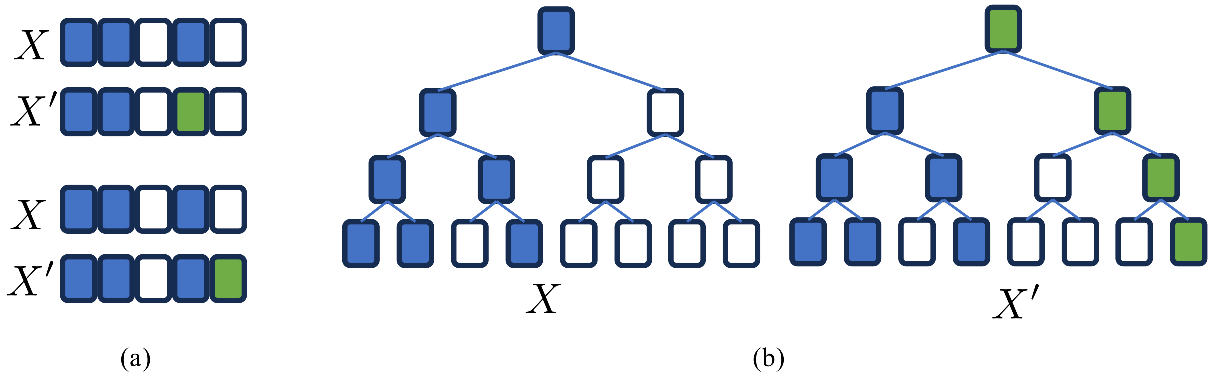

Figure 4: The figure on the left (a) illustrates the two possibilities when dealing with neighbouring inputs at any *fixed* level of the hierarchy. Here X and $X' = X \cup x'$ denote two arbitrary neighbouring datasets. The location of x' is denoted in green. In the case of the top left figure, the unconditional frequency of x' is greater than 0 for both datasets (we refer to such a x' as a merged element). In the bottom left figure, x' has a frequency of 1 in X' and 0 in X (which we refer to as being isolated). The figure (b) on the right denotes two neighbouring datasets X and X', where at every level of the hierarchy x' and its generalisations are isolated.

(from the assumption on Line 1). Let $\alpha = \frac{8}{\varepsilon}\log(2h/\delta)$. Let E_1 denote the event that $|\gamma| \leq \alpha/2$ and E_2 denote the event that $|w_p| \leq \alpha/2$. If $\Pr[E_1 \cap E_2] \geq 1-\delta$, then with probability $1-\delta$, $\beta'_p = \perp$, and p is not included in $\mathcal{S}$ (as the total randomness is not enough to push the noisy count over τ). From the tail bounds of the Laplace distribution we have that if $\alpha \geq \frac{8}{\varepsilon}\log(2h/\delta)$, then $\Pr_{\gamma \leftarrow_{\$} \mathsf{Laplace}(2/\varepsilon)}\left[|\gamma| > \alpha/2\right] \leq \frac{\delta}{2h}$, and $\Pr_{w_p \leftarrow_{\$} \mathsf{Laplace}(4/\varepsilon)}\left[|w_p| > \alpha/2\right] \leq \frac{\delta}{2h}$. Hence, at any level, the probability of an isolated prefix showing up in the output is $\Pr[\overline{(E_1 \cap E_2)}] \leq \frac{\delta}{h}$. There are at most h isolated prefixes, in the worst case (as shown in Figure 4b), so taking the union bound over h levels of the tree, we get that any isolated prefix shows up with probability at most δ. Thus the output of the Algorithm 1 when the neighbouring element x' remains isolated throughout is $(0,\delta)$-private. Note that we might have fewer than t levels of the tree when any prefix p is isolated. Setting $\tau > 1 + \frac{8}{\varepsilon}\log(2h/\delta)$ we cover the worst case, thereby all other cases.

Case II (Merged Neighbour): In this regime, $f_X(p) > 0$ and $f_{X'}(p) > 0$. In other words, p is *not* isolated. Observe that for any $p \succ q$ we have q is also not isolated i.e., $f_X(q) > 0$ and $f_{X'}(q) > 0$. This is true because going up one level p merges into a node q that is a generalisation of more elements than p. As p was not isolated, q cannot be isolated. In other words, let level $j = \mathsf{Level}(p)$. For all $1 \leq i < j$, the algorithm stays in the non-isolated regime. Let $t = |\mathcal{I}_{\text{Active}}|$, and label the elements in $\mathcal{I}_{\text{Active}}$ with $(p_1, p_2, \ldots, p_{t-1}, u^\star)$, where $p_i \succ p_j$ if $i < j$. Given $\mathcal{S}$, we have that for all $i \in [t-1]$, $p_i \notin \mathcal{S}$ and therefore both $F'_{\mathcal{S}'}(p) = f_{X'}(p)$, and $F_{\mathcal{S}}(p) = f_X(p)$. Let $\vec{w} = w_{p_1}, \ldots, w_{p_{t-1}}$. We have

$$\prod_{i \in \mathcal{I}_{\text{After}}} p(\beta_i, a_i) \tag{2}$$

$$= \Pr\left[\beta_{p_1}, \ldots, \beta_{p_{t-1}}, \beta_{u^\star} = (\perp)^{t-1}\top\right] \tag{3}$$

$$= \prod_{i=1}^{t-1} \Pr_{\gamma,\vec{w}}\left[f_X(p_i) + w_{p_i} + \gamma < \tau\right] \cdot \Pr_{\gamma, w_{u^\star}}\left[f_X(u^\star) + w_{u^\star} + \gamma \geq \tau\right] \tag{4}$$

Writing $i* = \arg\max_{i=1,\ldots,t-1} f_X(p_i) + w_{p_i} + \gamma$, we analyse the first product term:

$$\prod_{i=1}^{t-1} \Pr_{\gamma,\vec{w}}\left[f_X(p_i) + w_{p_i} + \gamma < \tau\right] \tag{5}$$

$$= \prod_{i=1}^{t-1} \Pr_{\gamma}\left[f_X(p_i) + w_{p_i} + \gamma < \tau | w_{p_i}\right] \Pr\left[w_{p_i}\right] \tag{6}$$

$$= \prod_{i=1}^{t-1} \Pr\left[w_{p_i}\right] \prod_{i=1}^{t-1} \Pr_{\gamma}\left[f_X(p_i) + w_{p_i} + \gamma < \tau | w_{p_i}\right] \tag{7}$$

$$= \Pr_{\gamma}\left[f_X(p_{i*}) + w_{p_{i*}} + \gamma < \tau | w_{p_{i*}}\right] \prod_{i=1}^{t-1} \Pr\left[w_{p_i}\right] \tag{8}$$

$$\leq e^{\varepsilon/2} \Pr_{\gamma}\left[f_X(p_{i*}) + w_{p_{i*}} + \gamma + 1 < \tau | w_{p_{i*}}\right] \prod_{i=1}^{t-1} \Pr\left[w_{p_i}\right] \tag{9}$$

$$= e^{\varepsilon/2} \prod_{i=1}^{t-1} \Pr\left[w_{p_i}\right] \prod_{i=1}^{t-1} \Pr_{\gamma}\left[f_{X'}(p_i) + w_{p_i} + \gamma < \tau | w_{p_i}\right] \tag{10}$$

$$= e^{\varepsilon/2} \prod_{i=1}^{t-1} \Pr_{\gamma, \vec{w}}\left[f_{X'}(p_i) + w_{p_i} + \gamma < \tau\right] \tag{11}$$

Equation (8) comes from the fact that as we are conditioning on w_{p_i}, for all $i \in [t-1]$, and thus we can treat $f_X(p_i) + w_{p_i}$ as a constant. Then the the statement $f_X(p_i) + w_{p_i} + \gamma < \tau$, for all $t-1$ queries is equivalent to the statement that $f_X(p_{i*}) + w_{p_{i*}} + \gamma < \tau$ (as the *same* randomness i.e., γ is added to all constant queries). Thus we have found a *single* query which covers all $t-1$ queries, and thus the actual degree of variability is 1, despite there being $(t-1)$ one-sensitive queries[11]. Finally, by the privacy of the Laplace Mechanism we have,

$$\Pr_{w_{u^\star} \leftarrow\$ \mathsf{Laplace}(4/\varepsilon)}\left[f_X(u^\star) + w_{u^\star} + \gamma \geq \tau | \gamma\right] \Pr\left[\gamma\right]$$
$$\leq e^{\varepsilon/4} \Pr_{w_{u^\star} \leftarrow\$ \mathsf{Laplace}(4/\varepsilon)}\left[f_{X'}(u^\star) + w_{u^\star} + \gamma \geq \tau | \gamma\right] \Pr\left[\gamma\right]$$

We "pay" $\varepsilon/4$ to estimate the residual frequency of $u^\star$ in Line 12. Note, it is important that we use fresh random noise at this point, as otherwise the output is not guaranteed to be meet the DP requirements, similar to the sparse vector technique [28]. Then, applying basic composition, $\Pr\left[\beta_{u^\star} = a_{u^\star}\right] \leq e^{\varepsilon/2} \Pr\left[\beta'_{u^\star} = a_{u^\star}\right]$. Combining this with (11), and the fact that there is no privacy loss in $\mathcal{I}_{\text{After}}$ and $\mathcal{I}_{\text{Unrelated}}$, we get that $\Pr\left[\mathsf{M}(X) = \mathcal{S}\right] \leq e^{\varepsilon} \Pr\left[\mathsf{M}(X') = \mathcal{S}\right]$ with probability $1 - \delta$. □

Theorem B.2 (Coverage And Error)**.** Let $\mathcal{S}$ denote the set of hierarchical heavy hitters selected by Algorithm 1. Then with probability $1 - \eta$, for all $p \notin \mathcal{S}$, $F_{\mathcal{S}}(p) \leq \tau - \Delta$. Additionally, for *all* $p \in \mathcal{H}$ with probability $1 - \eta$ we have $\max_{p \in \mathcal{H}} \left|\frac{f_X(p) - \tilde{f}_X(p)}{f_X(p)}\right| \leq \frac{\Delta}{\tau}$, where

$$\Delta = \frac{8}{\varepsilon}\left(\log\left[\frac{1}{\delta}\right] + \log\left[\frac{2h}{\eta}\right]\right)$$

Proof of Theorem B.2. **Coverage:** We give exact answers for any node p with $F_{\mathcal{S}}(p) = \widetilde{F}_{\mathcal{S}}(p) = 0$. For each level of the tree we have $n+1$ Laplace variables. From the tail bounds of the Laplace distribution we have with probability $1 - \eta/h$, the error for any node is upper bounded by $O\left(\frac{1}{\varepsilon}\log\left(\frac{nh}{\eta}\right)\right) \leq O\left(\frac{1}{\varepsilon}\log\left(\frac{h}{\delta\eta}\right)\right) = O\left(\frac{1}{\varepsilon}\left(\log\left[\frac{1}{\delta}\right] + \log\left[\frac{h}{\eta}\right]\right)\right)$. So if we set $\Delta = \Omega\left(\frac{1}{\varepsilon}\left(\log\left[\frac{1}{\delta}\right] + \log\left[\frac{h}{\eta}\right]\right)\right)$, then take the union bound over h levels of the tree, we have with probability $1 - \eta$ that the DP noise is upper bounded by $\left|w_p + \gamma \leq \Delta\right|$. As $p \in \mathcal{S}$ only if $\widetilde{F}_{\mathcal{S}}(p) + w_p + \gamma \geq \tau$, we have with probability $1 - \eta$ that $F_{\mathcal{S}}(p) \geq \tau - \Delta$, as required.

Relative Error: Notice that even without any DP noise, if we used just the hierarchical heavy hitters to estimate the frequency of a prefix $p \notin \mathcal{S}$, in the worst case, with probability $1 - \eta$ we underestimate the unconditional frequency of p by a factor of $\tau - \Delta$. For any $p \notin \mathcal{S}$,

$$f_X(p) - \sum_{q \in \mathcal{S} \wedge q \succeq p} F_{\mathcal{S}}(q) \leq \tau - \Delta$$

[11] This section of the analysis is closely related to the analysis of the Above Threshold Algorithm by Dwork et al. [30].

where c_p denotes the number of hierarchical heavy hitters used to estimate the frequency of p i.e., $c_p = |\{q \in \mathcal{S} : q \geq p\}|$. Of course if $p \in \mathcal{S}$, then using residual frequencies allow us to perfectly estimate the unconditional frequency of p. Also observe that if we need c_p nodes to estimate the unconditional frequency of p, then we must have $f_X(p) \geq c_p \cdot \tau$. Thus we can easily upper bound the relative error by a small constant independent of the height of the hierarchy or the number of heavy hitters in the hierarchy, as follows:

$$\left|f_X(p) - \widetilde{f}_X(p)\right| \tag{12}$$

$$= \left|f_X(p) - \sum_{q \in \mathcal{S} \wedge q \geq p} \widetilde{F}_{\mathcal{S}}(q)\right| \tag{13}$$

$$\leq \left|f_X(p) - \sum_{q \in \mathcal{S} \wedge q \geq p} F_{\mathcal{S}}(q)\right| + \sum_{q \in \mathcal{S} \wedge q \geq p} \left|F_{\mathcal{S}}(q) - \widetilde{F}_{\mathcal{S}}(q)\right| \tag{14}$$

$$\leq (\tau - \Delta) + c_p \cdot \Delta \tag{15}$$

Now since $f_X(p) \geq c_p \cdot \tau$, we get that for any $p \notin \mathcal{S}$

$$\left|\frac{f_X(p) - \widetilde{f}_X(p)}{f_X(p)}\right| \leq \frac{1}{c_p} + \frac{\Delta}{\tau} \tag{16}$$

Also for any $p \in \mathcal{S}$, we have

$$\left|\frac{f_X(p) - \widetilde{f}_X(p)}{f_X(p)}\right| \leq \frac{\Delta}{\tau} \tag{17}$$

□

Theorem C.1 (Privacy)**.** Algorithm 3 is (ε, δ)-DP.

Proof of Theorem C.1. Fix some level l. At each level of the tree, we have at most $\kappa + 1$ independent samples from a Laplace distribution, denoted by $w_1, \ldots, w_\kappa$ and γ_l in Figure 3. We also process each level of the data structure independently. The resulting output $(\mathcal{T}_l, C_l)$ or $(\mathcal{T}_l', C_l')$ is in one of two conditions stated in Lemma 1 and as illustrated in Figure 3. We will use the first κ independent samples to handle the second condition (Figure 3b), which is essentially the stability histogram problem with Laplace noise and global sensitivity 2.

Case 1. First we need to handle the isolated counters in stability histograms. Remember that in isolated counter events there is an element $x \in \mathcal{T}_l$ but $x \notin \mathcal{T}_l'$, or vice versa. From Lemma 1, as $|\mathcal{T} \cap \mathcal{T}'| \geq \kappa - 2$, there can be at most 2 isolated elements included in the output of the algorithm at any given level l. Let $a, b \in \mathcal{T}_l$ denote the possible isolated elements at level l after processing X. If a or b were to ever show up in the output, the privacy adversary could perfectly distinguish X and X'. For convenience, define $\upsilon = \frac{2h}{\varepsilon} \log(\frac{3h}{\delta})$. For a to show up in the final output, we need $w_a + \gamma_l$ to push the noisy count of a over $1 + 3\upsilon$. Once again from the tail bounds of the Laplace distribution, we have that the probability $\Pr[w_a > 2\upsilon] \leq \frac{\delta}{3h}$. A similar argument bounds the probability of either of γ and w_b exceeding υ and 2υ, respectively. Using the same union bound across both events, we get with probability at least $1 - \delta/h$ that both $w_a + \gamma_l \leq 3\upsilon$ and $w_b + \gamma_l \leq 3\upsilon$. For all counters, we add noise drawn from a Laplace distribution, using privacy parameter $\varepsilon/4h$ once during thresholding, and once during public release. Taking the union over the height of the hierarchy, we get that the first case is handled with (ε, δ)-privacy.

Case 2. Next we show that the other case (Figure 3a), where all counters in C and C' are off by the *same* amount in the *same* direction, is also handled with $\varepsilon/2h$-DP. This is case (2) of Lemma 1, where $C[i] = C'[i] - 1$ for all $i \in \mathcal{T}'$. Remember that our analysis considered the noise in $w_1, \ldots, w_\kappa$ already for stability histograms. Thus we want γ_l to cover for this other case entirely. Here we use notation $a1^\kappa$ to denote the vector $[a, a, \ldots, a]$ of size κ for any $a \in \mathbb{R}$, $\vec{w}$ succinctly denotes all the w_i's at level l. The analysis below is for any fixed $l \in [h]$, so we drop the subscript l to avoid an overload of notation. We consider any possible output vector $\vec{y}$: for a given $\vec{y}$ and once the noise $\vec{w}$ is fixed, the only freedom is in the choice of γ.

$$\Pr_{\vec{w}, \gamma}[C + \gamma 1^\kappa + \vec{w} = \vec{y}] \tag{18}$$

$$= \Pr_{\gamma}\left[C + \gamma 1^{\kappa} + \vec{w} = \vec{y} | \vec{w}\right] \Pr[\vec{w}] \tag{19}$$

$$= \Pr[\vec{w}] \prod_{i=1}^{\kappa} \Pr_{\gamma}\left[C[i] + \gamma + w_i = y_i | w_i\right] \tag{20}$$

$$= \Pr[\vec{w}] \prod_{i=1}^{\kappa} \Pr_{\gamma}\left[(C[i] - 1) + (1 + \gamma) = y_i | w_i\right] \tag{21}$$

$$= \Pr[\vec{w}] e^{\varepsilon/2h} \prod_{i=1}^{\kappa} \Pr_{\gamma}\left[C'[i] + \gamma = y_i | w_i\right] \tag{22}$$

$$= e^{\varepsilon/2h} \Pr[\vec{w}] \prod_{i=1}^{\kappa} \Pr_{\gamma}\left[C'[i] + \gamma + w_i = y_i | w_i\right] \tag{23}$$

$$= e^{\varepsilon/2h} \Pr_{\gamma}\left[C' + \gamma 1^{\kappa} + \vec{w} = \vec{y} | \vec{w}\right] \Pr[\vec{w}] \tag{24}$$

$$= e^{\varepsilon/2h} \Pr_{\vec{w},\gamma}\left[C' + \gamma 1^{\kappa} + \vec{w} = \vec{y}\right] \tag{25}$$

Equation (19) comes from Bayes' Rule. Equation (21) and (22) are the critical steps, where we make the switch from C to C' by adding one to γ, using our assumption that $C[i] = C'[i] - 1$ for all κ of the counter values in $\mathcal{T}$. Meanwhile, we hold the vectors $\vec{y}$ and $\vec{w}$ steady, and so the event is the same. We go from Equation (21) to Equation (22) using the privacy of the Laplace mechanism: the probability of seeing $(1 + \gamma)$ instead of γ is at most a factor of $e^{\epsilon/2h}$ higher. So despite there being κ counts, we *only* have to look at the dependence on γ. Finally, applying basic composition over h levels of the tree, we get that the entire algorithm restricted to the scenario depicted in Figure 3a is $(\varepsilon/2, 0)$-DP. Hence, regardless of which case we happen to be in, we obtain that the whole protocol is (ε, δ)-DP □

Theorem C.2 (Error)**.** For all $p \in \mathcal{S}$, we have with probability $1 - \eta$,

$$|\widetilde{f}_X(p) - f_X(p)| \leq \Delta$$

where

$$\Delta = \left(1 + \frac{6h \log(3h/\delta)}{\varepsilon}\right) + \frac{n}{\kappa + 1} + \left(\frac{8h}{\varepsilon} \log\left(\frac{2\kappa h}{\eta}\right)\right)$$

Proof of Theorem C.2. Next we state the well known estimation error due to space constraints for the Misra Gries algorithm. A proof for this lemma can be found in [31, Section 2].

Lemma 2 (MG Error)**.** Let $C[x]$ denote the frequency estimate for $x \in X$, given by a MG sketch with κ on input stream X. Then $C[x] \in \left[f_X(x) - \frac{|X|}{\kappa+1}, f_X(x)\right]$

For any fixed level of the hierarchy, with probability $1 - \eta/h$ we have three sources of error. The first source of $1 + \frac{6h \log(3h/\delta)}{\varepsilon}$ comes from suppressing isolated counts. The second error term $\frac{n}{\kappa+1}$ comes from the error of the deterministic MG counter (Lemma 2). For the third error term, for some α to be defined later, at each level $l \in [h]$, we want with probability at most $\eta/2h$ that $|\gamma_l| > \alpha/2$ and with probability at most $\eta/2h$ we want for $i \in [\kappa]$, $|w_i| \leq \alpha/2$. This guarantees, with probability $1 - \eta/h$, that $\max_{i \in \kappa} |w_i| + |\gamma_l| \leq \alpha$. Using the tail bounds of the Laplace distribution, and taking union bounds over all the counters we get

$$\Pr[\max_{i \in \kappa} w_i > \alpha/2] \leq \kappa e^{-\frac{\varepsilon}{8h}\alpha}$$

Setting $\eta/2h = e^{-\frac{\varepsilon}{8h}\alpha}$, we get

$$\alpha \geq \frac{8h}{\varepsilon} \log \frac{2\kappa h}{\eta}$$

We can also bound the error from γ_l using a similar analysis, and it turns out the above value for α suffices to bound the error due to γ_l. Finally, we take the union bound over all h levels, and get with probability $1 - \eta$, the total error incurred by *any* node is at most

$$\Delta = \left(1 + \frac{6h \log(3h/\delta)}{\varepsilon}\right) + \frac{n}{\kappa + 1} + \left(\frac{8h}{\varepsilon} \log\left(\frac{2\kappa h}{\eta}\right)\right)$$

□

Theorem C.3 (Coverage)**.** For all $p \notin \mathcal{S}$, with probability $1-\eta$, $F_{\mathcal{S}}(p) \leq \tau - \Delta$.

Proof of Theorem C.3. Let $\mathcal{S}$ denote the output of Algorithm 4. Fix $p \in \mathcal{S}$, define $\mathcal{A}_p = \{h \in \mathcal{S} : h \succ p \wedge \nexists h' \text{ s.t } h \prec h' \prec p\}$. Let s_p denote the amount of weight removed from the unconditional count of p in step 12 of Algorithm 4 for prefix p. Note that $s_p \leq \sum_{h\in\mathcal{A}_p} f_X(h)$ by definition, as MG counts cannot ever overestimate $f_X(e)$. We need to account any overestimation due to differential privacy. With probability $1-\eta$, we have that the noise per node due to DP is Δ_2. Thus, with probability $1-\eta$, $C_l[e] - \Delta_2$ in Line 12 is strictly less than $f_X(e)$ (as we have removed the DP contribution and $C_l[e] \leq f_X(e)$). So we always take off less than we could from the count of p. Any prefix p is included in $\mathcal{S}$ only if $C_l[p] + \Delta_1 > \tau - \Delta_1$. So for all $p \notin \mathcal{S}$, we must have $C_l[p] + \Delta_1 - s_p \leq \tau - \Delta_1$. Therefore, with probability $1-\eta$,

$$F_{\mathcal{S}}(p) = f_X(p) - \sum_{h\in\mathcal{A}_p} f_X(h) \tag{26}$$

$$\leq (C_l[p] + \Delta_1) - s_p \tag{27}$$

$$\leq \tau - \Delta_1 \tag{28}$$

Equation (26) comes from the definition of residual count. Equation (27) comes from the fact that $s_p \leq \sum_{h\in\mathcal{A}_p} f_X(h)$, and $f_X(p) \leq C_l[p] + \Delta_1$ with probability $1-\eta$ (from Theorem C.2). □

F Composition And Structure

Differential Privacy was initially motivated by the study of counting queries, and heavy hitter estimation can be seen as post processing of private release of counting queries. The privacy community has studied extensively the composition of privacy parameters when dealing with *arbitrary* counting queries. From basic composition we know that the maximum DP error of q invocations of the Laplace mechanism scales $O(1/\varepsilon \cdot q \log[q])$. Steinke and Ullman [32] show that we can save the $\log q$ factor by exploiting the structure of correlated noise used for DP, and have error scale by $O(q/\varepsilon)$. Of course we could also use advanced composition [26] on top of this to further reduce the error to $O\left(1/\varepsilon \cdot \sqrt{q}\log[1/\delta]\right)$. Kairouz et al. [33] show an exact characterisation of the best privacy parameters that can be guaranteed when composing many (ε, δ)-differentially private mechanisms. Unfortunately computing these parameters for arbitrary queries with different privacy parameters is #P-complete [34]. While the early work was focused on arbitrary counting queries, there has been considerable work in exploiting structure in queries to obtain better error than basic or advanced composition. Dwork et al. [23] proposed the Above Threshold algorithm and showed that one can group similar queries together, and replace them with a single query to pay for many queries just once. This insight has led to multiple constructions of highly accurate DP algorithms that would have been impractical if we considered basic or advanced composition [35, 36, 37, 38]. Kaplan et al. [24] identified that for certain distribution of queries, the composition of the above threshold algorithm could be further improved, by observing that not all large queries include the neighbouring element. Dong et al. [39] show how to bypass composition for special class of conjunctive queries. We direct the reader to the chapter by Steinke [40] for a detailed survey on the role of composition in differential privacy. Despite an enormous body of work on counting queries and composition, the question of hierarchical counting with privacy has remained unexplored. In this work, we show that one can use similar tricks to the above work to exploit the structure of a hierarchy, and get highly accurate algorithms, that are not possible with general purpose techniques.

G Greater Privacy For Larger Groups

As discussed earlier, we only bound the relative error of approximating the unconditional frequency of any prefix in Equation (16). As pointed out by Ghazi et al. [2, Page 2], if we instead wanted to upper bound the worst case absolute error, then it is known that the error *must* scale linearly with the height of the hierarchy. The problem of estimating private counts in a tree can be reformulated as releasing a *linear query* of the form $A\vec{x}$ privately, where $\vec{x}$ is a vector of unconditional counts for all leaves in the hierarchy, and then finding heavy hitters post processing. The matrix $A \in \{0,1\}^{|\mathcal{H}| \times |\mathcal{H}|}$ adjacency matrix representation of the tree that represents if one node is a parent of another or not. Releasing linear queries privately has been exhaustively studied in the privacy community [41, 20, 30, 27]. Edmonds et al. [20] provide lower bounds for the absolute error of linear queries under pure and approximate differential privacy. They show that even when considering (ε, δ)-DP, the absolute error for any private linear query algorithm scales $\|A\|_\infty = \Theta(h)$ (which is the same as just releasing the entire tree privately with Laplace noise). Remember, we are able to circumvent composition by leveraging the structure of a hierarchy. Requiring the error of every estimate at every level to be bounded by the same constant destroys this structure. We cannot use information gained lower down the hierarchy to make useful claims about elements higher in the hierarchy. We need to treat each level independently as we want the noise for each node to be independent. Thus the advantage of our algorithm, and that of [2, Algorithm 1] is that we can re-use information from earlier queries while still preserving privacy, paying for absolute error instead. A second advantage to considering relative error, is that it is the more practical notion of error. This was also observed by Pujol and Desfontaines [42], who posted as an open problem the task of finding algorithms that allow larger groups to have more privacy than smaller groups. When dealing with hierarchies, nodes higher up in the tree by definition can be estimated by counts lower in the tree (via partitions or hierarchical heavy hitters) - and this information is public knowledge. So we would like to utitlise this when designing algorithms. Consider the situation where the exact count of a node is 10^6. If we incur an absolute error of 100 units when estimating the count of a node, we do not expect that to affect the final social decision associated with this private statistic. However, when the exact answer is say 99, and we incur an error of 100 units, such a large error in estimation is unacceptable.

Relationship To Counting Over Trees: The estimation error from Theorem B.2 can be used to obtain better results in practice for [2, Definition 1.4] for the simpler problem of estimating unconditional frequencies with constant relative error. Note the main difference between our result and that of Ghazi et al. [2], is that they add noise with scale $O(c/\varepsilon)$ to estimate the count of any node, regardless of the distribution of heavy hitters[12]. In our case, as we show in Equation (16), the relative error is independent of both the height and the number of hierarchical heavy hitters. Furthermore, our algorithm is simpler to define. In Ghazi et al. [2, Algorithm 3], the authors use a geometric progression of decreasing τ's, and repeatedly apply [2, Algorithm 1] on these thresholds to get the relative error guarantee. It is not clear how to set these thresholds for practical algorithms on real-world datasets, and the constants are larger than the height of any hierarchical domain in practice.

[12] Remember c is an upper bound on the total number of hierarchical heavy hitters.