SpaceSaving$^\pm$: An Optimal Algorithm for Frequency Estimation and Frequent Items in the Bounded-Deletion Model

Fuheng Zhao
UC Santa Barbara
fuheng_zhao@ucsb.edu

Divyakant Agrawal
UC Santa Barbara
agrawal@cs.ucsb.edu

Amr El Abbadi
UC Santa Barbara
amr@cs.ucsb.edu

Ahmed Metwally
Uber, Inc.
ametwally@uber.com

ABSTRACT

In this paper, we propose the first deterministic algorithms to solve the frequency estimation and frequent item problems in the *bounded-deletion* model. We establish the space lower bound for solving the deterministic frequent items problem in the bounded-deletion model, and propose Lazy SpaceSaving$^\pm$ and SpaceSaving$^\pm$ algorithms with optimal space bound. We develop an efficient implementation of the SpaceSaving$^\pm$ algorithm that minimizes the latency of update operations using novel data structures. The experimental evaluations testify that SpaceSaving$^\pm$ has accurate frequency estimations and achieves very high recall and precision across different data distributions while using minimal space. Our experiments clearly demonstrate that, if allowed the same space, SpaceSaving± is more accurate than the state-of-the-art protocols with up to $\frac{logU-1}{logU}$ of the items deleted, where U is the size of the input universe. Moreover, motivated by prior work, we propose Dyadic SpaceSaving$^\pm$, the first deterministic quantile approximation sketch in the bounded-deletion model.

PVLDB Reference Format:
Fuheng Zhao, Divyakant Agrawal, Amr El Abbadi, and Ahmed Metwally. SpaceSaving$^\pm$: An Optimal Algorithm for Frequency Estimation and Frequent Items in the Bounded-Deletion Model . PVLDB, 15(6): 1215 - 1227, 2022.
doi:10.14778/3514061.3514068

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/ZhaoFuheng/SpaceSavingBoundedDeletionModel.

1 INTRODUCTION

With the development of new technologies and advancements in digital devices, massive amounts of data are generated each day and these data contain crucial information that needs to be analyzed. To make the best use of streaming big data, data sketch[1] algorithms are often leveraged to process the data only once and to provide essential analysis and statistical measures with strong accuracy guarantees while using limited resources. For instance, with limited space and one pass on the dataset, Hyperloglog [23] enables cardinality estimation, the Bloom Filter [8] answers set membership, and KLL [28, 33] provides quantile approximation.

Two fundamental problems in data streams are identifying the most frequently occurring items, a.k.a. frequent items, heavy hitters, Top-K, elephants, iceberg queries, and estimating the frequency of an item, a.k.a the frequency estimation problem. The formal definition of these two problems are included in Section 2.1. Several algorithms [12, 17, 36, 38] have been proposed to solve both problems with tunable accuracy guarantees using small memory footprints. These algorithms can be categorized into *counter* based and *linear sketch* based approaches. The counter based approach [38] tracks a subset of input items and their estimated frequencies. The linear sketch based approach [12, 17, 30] tracks attribute information from the universe. While linear sketches [12, 17] directly solve the frequency estimation problem, they require additional structures such as heaps or need to impose hierarchical structures over the assumed-bounded universe to solve the frequent items problem. The frequency estimation and frequent items problems have important applications, such as click stream analysis [20, 26, 38], distributed caches [45], database management [14, 22, 40, 43], and network monitoring [5, 27, 42]. In addition, if inputs are drawn from a bounded universe, frequency estimation sketches can also solve the quantile approximation problem [17, 44].

Historically, all sketches assumed the *insertion-only* model or the *turnstile* model. The insertion-only model consists only of insert operations, whereas the turnstile model consists of both insert and delete operations such that deletes are always performed on previously inserted items [44]. Supporting both insert and delete operations is harder, e.g., sketches in the turnstile model incur larger space overhead and higher update times compared to sketches in the insertion-only model. Jayaram et al. [29] observed that in practice many turnstile models only incur a fraction of deletions and proposed an intermediate model, the *bounded-deletions* model, in which at most $(1-\frac{1}{\alpha})$ of prior insertions are deleted where $\alpha \geq 1$ and $(1-\frac{1}{\alpha})$ upper bounds the delete:insert ratio. Setting α to 1, the bounded-deletion model becomes the insertion-only model.

The bounded-deletion model is important in many real-world applications such as summarizing product sales in electronic commerce platforms and rankings in standardized testing. Many companies use purchase frequency to check if their customers are satisfied with a product and to identify important groups for advertising

Proceedings of the VLDB Endowment, Vol. 15, No. 6 ISSN 2150-8097.
doi:10.14778/3514061.3514068

[1]The term sketch refers to the algorithms and data structures that can extract valuable information through **one pass** on the entire data.

and marketing campaigns. After customers purchase products, a certain percentage of the purchases may be returned and the frequency estimation should reflect these changes. However, for any financially viable company, it is highly unlikely that all of these customers will return their purchases and hence in most cases the bounded-deletion model can be assumed. In the context of standardized testing such as SAT, ACT, and GRE, frequency estimations are often used to compare and contrast performance among different demographics[2]. Students may request regrades of their exams only once to rectify any machine errors or human errors. Hence, the bounded-deletion model can be used with $\alpha = 2$. Recently, the bounded-deletion model has gained in popularity, and several algorithms have been proposed to discover novel properties of streaming tasks [10, 29, 32, 48].

In this paper, we present the SpaceSaving$^{\pm}$ algorithm that solves both frequency estimation and frequent items problems in the bounded-deletion model with state-of-the-art performance and minimal memory footprint. If the administrator of a large data set knows, a priori, that deletions are not arbitrarily frequent compared to insertions, then SpaceSaving$^{\pm}$ can efficiently capture these changes and identify frequent items with small space, fast update time, and high accuracy. In addition, inspired by quantile summaries [17, 24, 44], we further demonstrate how to leverage SpaceSaving$^{\pm}$ to support deterministic quantile approximation in the bounded-deletion model. In summary, the main contributions of this paper are: (i) we present Lazy SpaceSaving$^{\pm}$ and SpaceSaving$^{\pm}$, two space optimal deterministic algorithms in the bounded-deletion model and establish their space optimality and correctness; (ii) we propose the Dyadic SpaceSaving$^{\pm}$ sketch, the first deterministic quantile approximation sketch in the bounded-deletion model; (iii) we implement SpaceSaving$^{\pm}$ using two heaps to minimize the update time; and (iv) we evaluate SpaceSaving$^{\pm}$ and compare it to the state-of-the-art approaches [12, 17, 29] and achieve 5 orders of magnitude better accuracy on a real-world dataset.

The paper is organized as follows, Section 2 discusses the background of frequency estimation and frequent items problem, and gives an overview of previous algorithms. Section 3 introduces Lazy SpaceSaving$^{\pm}$ and SpaceSaving$^{\pm}$ in the bounded-deletion model, demonstrates that these algorithms are space optimal, and presents an efficient implementation using a min heap and a max heap data structure to minimize update time. Section 4 shows the experimental evaluations conducted using synthetic and real world datasets and compares SpaceSaving$^{\pm}$ to the state-of-the-art sketches that support delete operations. Section 5 introduces the Dyadic SpaceSaving$^{\pm}$ quantile sketch to solve the deterministic quantile approximation problem in the bounded-deletion model. Finally, Section 6 summarizes our contributions and concludes this work.

2 BACKGROUND

2.1 Preliminaries

Given a stream $\sigma = \{item_t\}_{t \in \{1,2,...,N\}}$ of length N and items drawn from *universe* of size U, the frequency of an item x is $f(x) = \sum_{t=1}^{N} \pi(item_t = x)$ where π returns 1 if $item_t$ is x, and returns 0 otherwise. The stream σ implicitly defines a frequency vector $F = \{f(a_1), ..., f(a_U)\}$ for items a_1,...,a_U in the *universe*. Some algorithms assume the *universe* is bounded, such as in the IP network monitoring context [42]. Many algorithms assume unit updates, such as the click stream, while others consider the scenario of weighted updates such as purchasing multiple units of the same item at once on an e-commerce platform. In this paper, we focus on the unit updates model and assume that items cannot be deleted if they were not previously inserted and hence all entries in frequency vector F are non-negative.

The frequency estimation problem takes an accuracy parameter ϵ and estimates the frequency of any item x such that $|\hat{f}(x) - f(x)| \leq \epsilon \cdot |F|_p$, where p can be either 1 or 2 corresponding to l_1 or l_2 norm and respectively provide l_1 or l_2 guarantees, $\hat{f}(x)$ is the estimated frequency and $f(x)$ is the actual frequency. When $p > 2$, providing l_p guarantee requires $poly(U)$ space [4]. In this paper, we focus on the l_1 problem variation. The ϕ frequent items problem is to identify a bag of *heavy items* whose frequency is greater than or equal to the specified threshold $\phi \cdot |F|_1$, where $0 < \phi < 1$. These heavy items are also known as the hot items.[3] In addition, some algorithms solve the (ϵ, ϕ)-approximate frequent items problem, which is to identify a bag of items B, given parameter $0 < \epsilon \leq \phi < 1$, such that B does not contain any element with frequency less then $(\phi - \epsilon)|F|_1$, i.e., $\forall i \in B, f(i) > (\phi - \epsilon)|F|_1$ and B contains all items with frequency greater than $\phi|F|_1$ i.e., $\forall i \notin B, f(i) < \phi|F|_1$.

2.2 Deterministic and Randomized Solutions

Reporting the exact frequent items requires $\Omega(N)$ space [13]. With limited memory, solving the exact frequent items problem is infeasible for large datasets. An alternative and practical approach in the context of big data is to use approximation techniques.

Deterministic solutions for the ϕ frequent items problem guarantee to return all heavy items and potentially some non-heavy items [21, 34, 38, 39]. Randomized solutions for the (ϵ, ϕ)-approximate frequent items problem allow the algorithm to fail with some probability δ [12, 17, 29]. In much of the literature, the failure probability is to set $\delta = O(U^{-c})$ where U is the bounded universe size and c is some constant. From the user perspective, deterministic algorithms provide stronger guarantees as all heavy items are identified. Randomized algorithms, on the other hand, with $1 - \delta$ probability report all heavy items and do not report any light weighted items.

2.3 Algorithms in Insertion-Only Model

The insertion-only model consists only of insert operations and many of the proposed algorithms in the insertion-only model are counter-based algorithms which maintain a fixed number of *item* and *count* pairs, and the underlying maintenance algorithm increments or decrements these counts to capture the frequency of items that are being tracked.

The first counter-based one pass algorithm to find the most frequent items in a large dataset dates back to the deterministic **Majority** Algorithm by Boyer and Moore in 1981 [9]. In 1982, Misra and Gries [39] generalized the majority problem and proposed the deterministic **MG** summary which uses $O(\frac{1}{\epsilon})$ space to solve the frequency estimation and frequent items problems. MG summary is a set of $\frac{1}{\epsilon}$ counters that correspond to monitored items. When a

[2]https://reports.collegeboard.org/pdf/2020-total-group-sat-suite-assessments-annual-report.pdf

[3]The term "Hot Items" was coined in [18]

Table 1: Comparison between different l_1 frequency estimation algorithm.

Sketch	Space	Update Time	Randomization	Model	Note
SpaceSaving [38]	$O(\frac{1}{\epsilon})$	$O(1)$	Deterministic	Insertion-Only	see Lemma 1
Count-Min [17]	$O(\frac{1}{\epsilon}log\frac{1}{\delta})$	$O(log\frac{1}{\delta})$	Randomized	Turnstile	Never Underestimate
Count-Median [12]	$O(\frac{1}{\epsilon}log\frac{1}{\delta})$	$O(log\frac{1}{\delta})$	Randomized	Turnstile	Unbiased Estimation
CSSampSim [29]	$O(\frac{1}{\epsilon}log\frac{1}{\delta}log\frac{\alpha logU}{\epsilon})$ bits	$\Theta(\frac{\alpha logU}{\epsilon U}log\frac{1}{\delta})$	Randomized	Bounded-Deletion	
Lazy-SpaceSaving$^\pm$	$O(\frac{\alpha}{\epsilon})$	$O(log\frac{\alpha}{\epsilon})$	Deterministic	Bounded-Deletion	see Lemma 7
SpaceSaving$^\pm$	$O(\frac{\alpha}{\epsilon})$	$O(log\frac{\alpha}{\epsilon})$	Deterministic	Bounded-Deletion	

Table 2: Frequently used symbols

σ	Data stream
N	Data stream length
universe	All data are drawn from *universe*
U	Size of the *universe*
F	Frequency vector
$f(x)$	x's true frequency
$\hat{f}(x)$	x's estimated frequency
ϵ	Accuracy
δ	Failure probability
I	Number of insertions
D	Number of deletions
α	In the bounded-deletion model, $D \leq (1-\frac{1}{\alpha})I$
minCount	The minimum count inside a sketch
minItem	The item with *minCount*

new item arrives, MG performs the following updates: if the new item is monitored, then increase its count by 1. Else if the summary is not full, monitor the new item. Else decrement all counts by 1 and remove any items with a count of zero. As a result of MG decrementing all counts by 1 when an arriving item is unmonitored, MG always underestimate item's frequency, and a hash-table implementation requires worst case $O(1/\epsilon)$ update time. Two decades later, Manku and Motwani [36] proposed a randomized **StickySampling** algorithm and a deterministic **LossyCounting** algorithm with worst case space $O(\frac{1}{\epsilon}log(\epsilon N))$, which exceeds the memory cost of MG summary. In 2003, Demaine et al. [21] and Karp et al. [34] independently generalized the majority algorithm and proposed the **Frequent** algorithm, which are both a rediscovery of the MG.

Two years later, in 2005, Metwally, Agrawal, and El Abbadi [38] proposed the **SpaceSaving** algorithm that provides highly accurate frequency estimates for frequent items and also presents a very efficient method to process insertions. SpaceSaving uses k counters to store an item's identity, estimated count and estimation error information, i.e., $(item, count_{item}, error_{item})$, and $error_{item}$ is an upper bound on the difference between the item's estimated frequency and its true frequency. When $k=\frac{1}{\epsilon}$, SpaceSaving solves both frequency estimation and frequent items problem. As shown in Algorithm 1, insertions proceed as follows, when a new item (*newItem*) arrives: if *newItem* is monitored, then increment its count; if *newItem* is not monitored and sketch size not full, then monitor *newItem*, and set $count_{newItem}$ to 1 and $error_{newItem}$ to 0; otherwise, SpaceSaving replaces the item (*minItem*) with the minimum count (*minCount*) by *newItem*, sets $error_{newItem}$ to *minCount* and increments $count_{newItem}$. In SpaceSaving [38], $error_{item}$ is only used to show certain properties of the algorithm, while in this work we leverage this information for handling deletions. As shown in Algorithm 2, to estimate the frequency of an item in SpaceSaving, if the item is inside the sketch then report its count value, otherwise report 0. In [2], Agarwal et al. showed that both SpaceSaving and MG are mergeable [4], and a SpaceSaving with k counters can be isomorphically transformed into a MG summary with $k-1$ counters. Although SpaceSaving and MG share similarities, they follow different sets of update rules. When a new inserted item is unmonitored and the sketch is full, SpaceSaving replaces the min item with the new item and increments the count by one, whereas the MG decrements all item counts' by 1. As a result, SpaceSaving maintains an upper bound on the frequency of stored items, while MG always underestimates the frequency. Since SpaceSaving always increments one of the counts by one, the sum of all counts in SpaceSaving is equal to the $|F|_1$. Moreover, the SpaceSaving elegantly handles the case when an unmonitored new item arrives and the sketch is full, and naturally leads to a min-heap implementation such that incrementing any count and replacing the min item have $O(logk)$ update times, where k is the number of counters. SpaceSaving can also be implemented with a linked list data structure by keeping items with equal counts in a group, resulting in an $O(1)$ update time [38].

SpaceSaving satisfies the following properties (the first three properties are proved in [38] while the latter two are proved in [47]):

Lemma 1. *Frequency estimations for monitored items are never underestimated in SpaceSaving.*

Lemma 2. *SpaceSaving with $k=\frac{1}{\epsilon}$ counters ensures that after processing I insertions, the minimum count of all monitored items is no more than $\frac{I}{k}=\epsilon I$, i.e, $minCount < \epsilon I$.*

Lemma 3. *All items with frequency greater than or equal to minCount are inside the SpaceSaving.*

Lemma 4. *The sum of all estimation errors in the sketch is an upper bound on the sum of the frequencies of all unmonitored items.*

Lemma 5. *SpaceSaving with $\frac{1}{\epsilon}$ counters can estimate the frequency of any item with an additive error less than ϵI.*

Lemma 2 and Lemma 3, show that SpaveSaving with $\frac{1}{\epsilon}$ counters reports all items whose frequencies are greater than or equal to

[4] Mergeability is desired for distributed settings and means summaries over datasets can be merged into a single summary as if the single summary processed all datasets.

$\epsilon|F|_1$. Empirically, many studies have demonstrated that SpaceSaving outperforms other deterministic algorithms and it is considered to be the state of the art for finding frequent items [13, 35]. Moreover, due to the superior performance of SpaceSaving, many works use it as a fundamental building block [5, 42, 43, 45, 46]. Recently, a new randomized algorithm **BPtree** was proposed by Braverman et al. [11] to solve the frequent items problem with l_2 guarantees in the insertion-only model using $O(\frac{1}{\epsilon^2}log\frac{1}{\epsilon})$ space.

Algorithm 1: SpaceSaving Insert Algorithm

```
for item from insertions do
    if item ∈ Sketch then
        count_item += 1 ;
    else if Sketch not full then
        Sketch = Sketch ∪ item ;
        count_item = 1 ;
        error_item = 0 ;
    else
        // Sketch is full;
        minItem = min_{minItem∈Sketch} count_minItem;
        error_item = count_minItem ;
        count_item = count_minItem+1 ;
        Replace (minItem, count_minItem, error_minItem) by
          (item, count_item, error_item)
end
```

Algorithm 2: SpaceSaving Query(item)

```
if item ∈ Sketch then
    return count_item
return 0;
```

2.4 Algorithms in Turnstile Model

In the turnstile model, the stream consists of both insert and delete operations such that the deletes are always performed on previously inserted items. The sketches for solving the frequency estimation problem in the turnstile model are known as **linear sketches** [13]. While the counter-based solutions solve both the frequency estimation and frequent items problems, the linear sketch solutions directly answer the frequency estimation problem but need additional information to solve the frequent items problem. When assuming the inputs are from a bounded *universe*, linear sketches can query all items in the universe to identify the frequent items.

In 1999, Alon et al. [3] proposed the randomized **AMS** sketch to approximate the second frequency moment. Charikar et al. [12] improved upon the AMS sketch and proposed the randomized **Count-Median** sketch. The Count-Median sketch provides an unbiased estimator and uses $O(\frac{1}{\epsilon}log\frac{1}{\delta})$ and $O(\frac{1}{\epsilon^2}log\frac{1}{\delta})$ space to solve the l_1 and l_2 frequency estimation problems respectively. Cormode and Muthukrishnan [17] proposed the **Count-Min** sketch that shares a similar algorithm and data structure as the Count-Median sketch. Count-Min sketch never underestimates frequencies, and uses $O(\frac{1}{\epsilon}log\frac{1}{\delta})$ space to solve the l_1 frequency estimation problem.

Although one may exhaustively iterate through the universe to find frequent items, iterating through the universe can be slow and inefficient. As a result, Cormode and Muthukrishnan [17] suggested to imposes a hierarchical structure on the bounded universe, such that there are $logU$ layers and one Count-Min or Count-Median sketch per layer. Then use divide-and-conquer to search for the frequent items from the largest range to an individual item. The required space is $O(\frac{1}{\epsilon}log\frac{1}{\delta}logU)$ and update time is $O(log\frac{1}{\delta}logU)$. Dyadic intervals are in the form of $[i2^j, (i+1)2^j - 1]$ for $j \in log_2U$ and any constant i, such that any ranges can be decomposed into at most log_2U disjoint dyadic ranges [15]. Dyadic intervals over a bounded universe can be integrated with frequency estimation sketches to solve the quantile approximation problem in the turnstile model [17, 24, 44].

2.5 Algorithms in Bounded-Deletion Model

In the bounded-deletion model, the stream consists of both insert and delete operations and a constant $\alpha \geq 1$ is given such that at most $(1 - \frac{1}{\alpha})$ of prior insertions are deleted, i.e., $D \leq (1 - \frac{1}{\alpha})I$ where I is the number of insertions and D is the number of deletions. Jayaram et al. [29] proposed the CSSS (**C**ount-Median **S**ketch **S**ample **S**imulator) to solve the frequency estimation problem in the bounded-deletion model. Assuming $\delta = O(U^{-c})$ for some constant c and the maximum entry of F is $O(U)$, then the Count-Min and the Count-Median sketches require $O(\frac{1}{\epsilon}log^2U)$ bits and achieves the optimal space lower bound in the turnstile model [31]. Jayaram et al. [29] pointed out that in the bounded-deletion model by simulating the Count-Median sketch over $O(\frac{\alpha logU}{\epsilon})$ uniformly sampled items from the stream and then scaling the counts at the end, the *CSSS* sketch can accurately approximate the true frequency of an item with high probability. Hence, by carefully tuning the size of the Count-Median, CSSS requires $O(\frac{1}{\epsilon}log\frac{1}{\delta}log\frac{\alpha logU}{\epsilon})$ bits, improving the overall space compared to sketches in the turnstile model.

2.6 Summary

In Table 1, we compare the differences and similarities among several different sketches for l_1 frequency estimation. These sketches can also solve l_1 heavy hitters, though some sketches may need additional modifications to the parameters or leverage external data structures. In Table 2, we listed the important symbols used in the paper. Counter-based solutions have many advantages over linear sketches. Counter-based solutions are guaranteed to report all heavy items; they use $O(log\frac{1}{\epsilon})$ update time instead of $O(logU)$ update time where $\frac{1}{\epsilon}$ is often less than the universe size U; and they make no assumptions on the *universe* and thus can be useful in Big Data applications where items are drawn from unbounded domains. In this paper, we present SpaceSaving$^{\pm}$, an optimal counter-based deterministic algorithm with l_1 guarantee to solve both the frequency estimation and frequent items problem in the bounded-deletion model using $O(\frac{\alpha}{\epsilon})$ space.

3 THE SPACESAVING$^{\pm}$ ALGORITHM

In this section, we first show the space lower bound for solving the frequent items problem in the bounded-deletion model. Then, we introduce the Lazy $SpaceSaving^{\pm}$ and $SpaceSaving^{\pm}$ algorithms with optimal space to solve both the frequency estimation and frequent items problems in the bounded-deletion model in which the total number of deletions (D) is less than $(1 - \frac{1}{\alpha})$ of the total insertions (I) where $\alpha \geq 1$. Given a user specified accuracy on the parameter ϵ, a deterministic algorithm for frequency estimation and frequent items problems must:

- Approximate the frequency of all items i with high accuracy such that $\forall\ i : |f(i) - \hat{f}(i)|) \leq \epsilon|F|_1$; and
- Report all the items with frequency greater than or equal to $\epsilon|F|_1$.

We propose Lazy SpaceSaving$^{\pm}$ and SpaceSaving $^{\pm}$. The main difference between the two variants of SpaceSaving$^{\pm}$ arises in the way deletions are handled. Since we assume the strict bounded-deletion model, a delete operation must correspond to a previously inserted item. If the item is being tracked in the sketch, processing such a delete operation is straightforward since the count associated with the item can be decreased by 1. On the other hand, the challenge arises when the sketch maintenance algorithm encounters a delete of an item that is not being tracked in the sketch. We develop different ways of handling such a delete in the two algorithms and the resulting correctness guarantees.

3.1 Space Lower Bound

We first show that there is no counter based algorithm that can solve the deterministic frequent items problem in the bounded-deletion model using less than $\frac{\alpha}{\epsilon}$ counters.

Theorem 1. *In the bounded-deletion model, any counter based algorithm needs at least $\frac{\alpha}{\epsilon}$ counters to solve the deterministic frequent items problem i.e, identify all the items with frequency greater than or equal to $\epsilon|F|_1$.*

Proof. By Contradiction.

Assume that there exists a counter based deterministic solution using $k < \frac{\alpha}{\epsilon}$ counters that can report all the items with frequency greater than or equal to $\epsilon|F|_1$. Consider a stream σ with bounded-deletions that contains I insertions and D deletions where all insertions come before any deletions. Let the I insertions consist exactly of $\frac{\alpha}{\epsilon}$ unique items, each with an exact count of $\frac{\epsilon}{\alpha}I$. After processing all insertions, the optimal algorithm with $k < \frac{\alpha}{\epsilon}$ counters will monitor at most k unique items, and there would be at least one item from the insertions that is left out. Let the set *Missing* contains all such unique items that appeared in I but are not monitored by the optimal algorithm. Now let the $D = (1 - \frac{1}{\alpha})I$ deletions be applied arbitrarily on the monitored items. After all D deletions, all items in *Missing* have frequency of $\frac{\epsilon}{\alpha}I$ in which $\frac{\epsilon}{\alpha}I \geq \epsilon(I - D) \geq \epsilon|F|_1$, and these items are frequent and must be monitored by the optimal algorithm. However, the sketch, with space k, after processing all insertions loses the information regarding *Missing*. Therefore, it is not possible to use less than $\frac{\alpha}{\epsilon}$ counters to solve the deterministic frequent items problem in the bounded-deletion model. □

3.2 Lazy SpaceSaving$^{\pm}$ Approach

Since supporting both insertions and bounded deletions is a much harder task compared to only allowing for insertions, the overall space needs to be increased. From the previous section, we can see that if the goal is to report all the items with frequency more than $\epsilon|F|_1$, where $|F|_1 = I - D$, we need to track more items. Since before any deletions, the sketch has no knowledge regarding which items are going to be deleted, then all elements with frequency higher than $\epsilon(I - D)$ are potential candidates before any deletions. We need an algorithm that can identify these potential candidate items.

By Lemma 2 and Lemma 3, SpaceSaving [38] with space k reports all the items with frequency greater than or equal to $\frac{I}{k}$. Therefore by using $k = \frac{\alpha}{\epsilon}$ space to process I insertions on the SpaceSaving algorithm, it will report all item with frequency greater than or equal to $\frac{\epsilon}{\alpha}I$. Since we know $\frac{1}{\alpha} \leq \frac{(I-D)}{I}$, $\frac{\epsilon}{\alpha}I \leq \epsilon I\frac{(I-D)}{I} = \epsilon(I - D)$. Hence by using $\frac{\alpha}{\epsilon}$ counters, all the items with frequency greater than or equal to $\epsilon(I - D)$ will be identified.

Interestingly, we find that modifying the original SpaceSaving algorithm with $O(\frac{\alpha}{\epsilon})$ space leads to an algorithm that solves the frequency estimation and frequent items problems in the bounded-deletion model. The Lazy SpaceSaving$^{\pm}$ algorithm handles insertions exactly in the same manner as in the original Algorithm 1. For deletions, the Lazy SpaceSaving$^{\pm}$ decreases the monitored item counter, if the deleted item is monitored. Otherwise, the deletions on unmonitored item are ignored, as shown in Algorithm 3. The frequency is still estimated according to Algorithm 2. The rationale for this design is that an unmonitored item has estimated frequency of 0 and deletions of the unmonitored items will not amplify the difference but in fact narrows the difference. Initially, this may seem to be counter-intuitive. Another way to think about it is that the frequency estimations for unmonitored items can only be underestimations. Thus, the decrease in an unmonitored item's exact frequency reduces the underestimation.

Algorithm 3: Lazy SpaceSaving$^{\pm}$: Deletion Handling

```
for item from deletions do
    if item in Sketch then
        count_item -= 1 ;
    else
        //ignore
end
```

We now formally establish that Algorithm 3 solves the frequency estimation problem in the bounded-deletion model. Let $error_{max}$ be the maximum error of frequency estimations in Lazy SpaceSaving$^{\pm}$. We show by induction that $error_{max}$ is always less than $\epsilon(I - D)$. First, we establish an upper bound on *minCount* in Lemma 6.

Lemma 6. *The minimum count, minCount, in Lazy SpaceSaving$^{\pm}$ with k counters is less than or equal to $\frac{I}{k}$.*

Proof. Since deletions never increment any counts, *minCount* is maximized by processing I insertions. With I insertions and no deletions, the sum of all counts is equal to I. *minCount* is largest when all the other counts are *minCount*. Hence, $minCount \leq \frac{I}{k}$. □

Theorem 2. *In the bounded-deletion model where $D \leq (1 - \frac{1}{\alpha})I$, after processing I insertions and D deletions, Lazy SpaceSaving$^{\pm}$ using $O(\frac{\alpha}{\epsilon})$ space solves the frequency estimation problem in which $\forall i, |f(i) - \hat{f}(i)| < \epsilon(I - D)$ where $f(i)$ and $\hat{f}(i)$ are the exact and estimated frequencies of an item i.*

Proof. By Induction.

Base case: After $i' < I$ insertions and 0 deletions with $O(\frac{\alpha}{\epsilon})$ space, we show that $error_{max}$ is less than $\epsilon(I-D)$ as follows: By Lemma 5 (of the insertion-only *SpaceSaving*), $error_{max} < i'\frac{\epsilon}{\alpha} \leq \epsilon\frac{i'(I-D)}{I} < \epsilon(I-D)$.

Induction hypothesis: After $i < I$ insertions and $d < D$ deletions, the maximum error of frequency estimations based on the sketch is $error_{max} < \epsilon(I-D)$.

Induction Step: Consider the case when the $(i+d+1)^{th}$ input item is an insertion. If the newly inserted item x is monitored or the sketch is not full, then no error is introduced. If the newly inserted item x is not monitored and the sketch is full, then x replaces the *minItem* which is the item with minimum count, *minCount*, in all monitored items. Based on Lemma 6, by using $\frac{\alpha}{\epsilon}$ counters in Lazy SpaceSaving$^{\pm}$, $minCount \leq i\frac{\epsilon}{\alpha} < \epsilon(I-D)$. The estimated frequency for x is *minCount*+1 and x is at most overestimated by *minCount*. The frequency estimation for *minItem* becomes 0, and *minItem*'s frequency estimation is off by at most *minCount*. Therefore, $error_{max}$ after processing the newly inserted item is still less than $\epsilon(I-D)$.

Consider the case when the $(i+d+1)^{th}$ input item is a deletion. If the newly deleted item x is monitored, its corresponding counter will be decremented and no extra error is introduced and $error_{max}$ is still less than $\epsilon(I-D)$. If the newly deleted item x is not monitored, then the algorithm ignores this deletion. The frequency estimation errors for monitored items do not change and they are still less than $\epsilon(I-D)$. Moreover, before the arrival of x, $\forall i \notin Sketch, f(i)-\hat{f}(i) = f(i) - 0 < \epsilon(I-D)$. By ignoring the deletion of the unmonitored items, $\forall i \notin Sketch, (f(i)-1)-\hat{f}(i) < f(i)-\hat{f}(i) < \epsilon(I-D)$.

Conclusion: By the principle of induction, Lazy SpaceSaving$^{\pm}$ using $O(\frac{\alpha}{\epsilon})$ space solves the frequency estimation problem with bounded error, i.e, $\forall i, |f(i)-\hat{f}(i)| < \epsilon(I-D)$. □

Lazy SpaceSaving$^{\pm}$ also solves the frequent items problem. To prove this, we first show Lazy SpaceSaving$^{\pm}$ never underestimates the frequency of a monitored item.

Lemma 7. *Lazy SpaceSaving$^{\pm}$ never underestimates the frequency of monitored items.*

Proof. Since the handling of insertions is the same as the SpaceSaving and SpaceSaving never underestimates the frequency of monitored items by Lemma 1, it is clear that the insertions can not lead to frequency underestimation for monitored items. When handling deletions, Lazy SpaceSaving$^{\pm}$ only decrements the count when the deleted item is monitored. Since the deletion of a monitored item implies its exact frequency and its estimated frequency both decrease by one, this procedure has no effect on the difference between its exact frequency and estimated frequency. Therefore, Lazy SpaceSaving$^{\pm}$ never underestimates the frequency of monitored items. □

Since Lazy SpaceSaving$^{\pm}$ never underestimates, then report all the items with frequency estimations greater than or equal to $\epsilon(I-D)$, then all frequent items will be reported as shown in Theorem 3.

Theorem 3. *In the bounded-deletion model where $D \leq (1-\frac{1}{\alpha})I$, Lazy SpaceSaving$^{\pm}$ solves the frequent items problem using $O(\frac{\alpha}{\epsilon})$ space.*

Proof. By Contradiction.

Assume a frequent item x is not reported and by definition of frequent items, $f(x) \geq \epsilon(I-D)$. Since it is not reported, its frequency estimation, $\hat{f}(x)$, must be less than $\epsilon(I-D)$. There are two cases where x will not be reported: (i) x is not monitored, or (ii) x is monitored, but its frequency is underestimated, i.e., $\hat{f}(x) < \epsilon(I-D)$.

In the first case where x is not monitored, the estimation frequency of x is 0, i.e, $\hat{f}(x) = 0$. Since x is by assumption a frequent item, the frequency estimation difference for x is $|\hat{f}(x) - f(x)| \geq \epsilon(I-D)$. However, this contradicts Theorem 2 in which, for any items, the difference between its exact frequency and estimated frequency is less than $\epsilon(I-D)$.

In the second case, x is monitored but not reported which implies $\hat{f}(x)$ is less than $\epsilon(I-D)$. Since x is frequent, $\hat{f}(x)$ is an underestimation. However, by Lemma 7, Lazy SpaceSaving$^{\pm}$ never underestimates the frequency of monitored items.

Hence, by contradiction Lazy SpaceSaving$^{\pm}$ solves the deterministic frequent items problem. □

3.3 An illustration of Lazy SpaceSaving$^{\pm}$

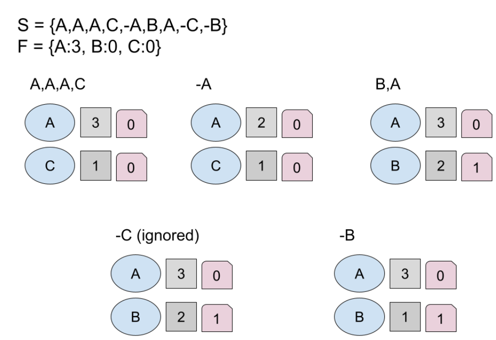

Figure 1: Input Stream consisting of 6 insertions and 3 deletions. Each tuple represents $(item, count_{item}, error_{item})$**.**

Consider an instance of Lazy SpaceSaving$^{\pm}$ with capacity of 2. The input stream σ is $(A, A, A, C, -A, B, A, -C, -B)$ where the minus sign indicate a deletion. The corresponding true frequency of A is 3 while the true frequency of all other items is 0. For the first four insertions and one deletion of the monitored item A, the sketch maintains the exact count with no errors. When the sixth item B arrives, B replaces item C, since C has the minimum count. The following insertion is A and since A is monitored, A's count increases. Then items $-C, -B$ arrive. Since C is not monitored, Lazy SpaceSaving$^{\pm}$ ignores the deletion of C, and the deletion of monitored item B decreases the corresponding count, as shown in Figure 1. After processing all inputs, the lazy-approach does not underestimate the frequency of the items in the sketch. It overestimates the frequency of item B in which $\hat{f}(B) - f(B) = 1$ and $\hat{f}(A) - f(A) = 0$.

3.4 SpaceSaving$^{\pm}$

While Lazy SpaceSaving$^{\pm}$ elegantly satisfies all the necessary requirements, the average frequency error and total frequency error may increase if there are significant deletions of the unmonitored items. Therefore, we propose SpaceSaving$^{\pm}$, a novel algorithm and data structures that accurately handles deletions of the unmonitored items and item's frequency is still estimated according to Algorithm 2. Interestingly, we experimentally show that SpaceSaving$^{\pm}$ performs better than Lazy SpaceSaving$^{\pm}$ when they are both allocated the same sketch space, even though we need more space by a constant factor to establish the correctness of SpaceSaving$^{\pm}$.

Both SpaceSaving and our proposed Lazy SpaceSaving$^{\pm}$ algorithms have the property of never underestimating the frequency of the monitored items. Since the ϵ-approximation requirement is $\forall i, |f(i) - \hat{f}(i)| < \epsilon(I - D)$, there are opportunities to reduce the amount of overestimation for the monitored items, as long as the difference is still within this bound. We observe that an item with a large estimation error indicates that it is unlikely to be a heavy item, as heavy items are often never evicted from the sketch and have small estimation error. In addition, items with large estimation errors are often overestimated due to the aggregation of the frequencies of many less-weighted items. SpaceSaving$^{\pm}$ leverages this intuition. It handles the insertions of all items, and the deletions of the monitored items exactly in the same way as the Lazy SpaceSaving$^{\pm}$. For the deletions of the unmonitored items, SpaceSaving$^{\pm}$ decrements the count of the item that has the maximum estimation error inside the sketch, as shown in Algorithm 4. The estimation error in SpaceSaving$^{\pm}$ is an upper bound on the difference between the item's estimated frequency and its true frequency. With this modification, the estimated frequency of any item reduces either from being replaced or from a deletion of an unmonitored item. In the following proofs, SpaceSaving$^{\pm}$ uses $\frac{2\alpha}{\epsilon}$ to ensure (i) no item can be severely overestimated, and (ii) no item can be severely underestimated. To estimate the frequency of an item, we still use Algorithm 2. Before analyzing the correctness of the algorithm, we first establish three helpful lemmas.

Lemma 8. *The minimum count, minCount, in SpaceSaving$^{\pm}$ with $\frac{2\alpha}{\epsilon}$ counters is less than or equal to $\frac{\epsilon}{2}(I - D)$.*

Proof. Similar to the proof of Lemma 6. Since deletions never increment any counts, *minCount* is maximized by processing I insertions and hence the sum of all the counts is upper bounded by I. *minCount* is largest when all the other counts are also *minCount*. Hence, $minCount \leq \frac{\epsilon I}{2\alpha} \leq \frac{\epsilon(I-D)}{2}$. □

Lemma 9. *The maximum estimation error, $argmax_{j \in Sketch} error_j$, in SpaceSaving$^{\pm}$ with $\frac{2\alpha}{\epsilon}$ counters is less than $\frac{\epsilon}{2}(I - D)$.*

Proof. The estimation error only increases when *minItem* is replaced by a newly inserted item and after the replacement, the estimation error becomes *minCount*. *minCount* is maximized when the input contains I insertions and no deletions. Hence by Lemma 2, SpaceSaving$^{\pm}$ with $\frac{2\alpha}{\epsilon}$ counters has $minCount < \frac{\epsilon}{2}(I - D)$. The estimation error is at most *minCount* and less than $\frac{\epsilon}{2}(I - D)$. □

Lemma 10. *The sum of all estimation errors in SpaceSaving$^{\pm}$, is an upper bound on the sum of frequencies of all unmonitored items and the maximum estimation error is greater than or equal to 0.*

Proof. The deletion of a monitored item has no effect on the sum of the estimation errors, and it has no effect on the sum of the frequencies of the unmonitored items. The deletion of an unmonitored item decreases both the sum of the frequencies of the unmonitored items by 1 and the sum of the estimation errors by 1. From this observation and Lemma 4, we can conclude that in SpaceSaving$^{\pm}$ with k counters, the sum of all estimation errors is an upper bound on the sum of frequencies of all unmonitored items. Since the sum of frequencies of all unmonitored items is greater than or equal to 0 and the sum of all estimation errors is upper bounded by k times the maximum estimation error, the maximum estimation error is greater than or equal to 0. □

Algorithm 4: SpaceSaving$^{\pm}$: Deletion Handling

```
1 for item from deletions do
2     if item in Sketch then
3         count_item -= 1 ;
4     else
5         j = arg max_{j∈Sketch} error_j ;
6         count_j -= 1 ;
7         error_j -= 1 ;
8 end
```

Theorem 4. *In the bounded-deletion model where $D \leq (1 - 1/\alpha)I$, after processing I insertions and D deletions, SpaceSaving$^{\pm}$ using $O(\frac{\alpha}{\epsilon})$ space solves the frequency estimation problem in which $\forall i, |f(i) - \hat{f}(i)| < \epsilon(I - D)$ where $f(i)$ and $\hat{f}(i)$ are the exact and estimated frequencies of an item i.*

Proof. Consider an instance of SpaceSaving$^{\pm}$ with $\frac{2\alpha}{\epsilon}$ counters to process I insertions and D deletions. First, we prove there is no item i such that the frequency estimate of i severely overestimate its true frequency, i.e, $\nexists i, \hat{f}(i) - f(i) > \epsilon(I - D)$. In SpaceSaving$^{\pm}$, the handling of deletions can not lead to any overestimation as counters will only be decremented, and only the replacement of the *minItem* due to a newly inserted item can lead to frequency overestimation of the newly inserted item. From Lemma 8, the *minCount* in SpaceSaving$^{\pm}$ with $\frac{2\alpha}{\epsilon}$ counters is no more than $\frac{\epsilon}{2}(I - D)$. The overestimation of a newly inserted item can be at most *minCount*. Therefore, no item can be overestimated by more than $\frac{\epsilon}{2}(I - D)$.

Second, we prove there is no item that can be severely underestimated i.e, $\nexists i, \hat{f}(i) - f(i) < -\epsilon(I - D)$. Two operations may lead to frequency underestimation: (i) Replacement of *minItem*, or (ii) Deletion of an unmonitored item. For the first case, *minCount* is always less than $\frac{\epsilon}{2}(I - D)$, and the amount of underestimation is less than $\frac{\epsilon}{2}(I - D)$ for any item due to the replacement.

We show that the deletion of an unmonitored item can lead to at most $\frac{\epsilon}{2}(I - D)$ frequency underestimation. Based on Lemma 9 and Lemma 10, the maximum estimation error must be less than $\frac{\epsilon}{2}(I - D)$ and greater than or equal to 0. In Algorithm 4, lines 6 and 7, the deletion of an unmonitored item decreases both the count and the estimation error of the item with the maximum estimation error. Call this item x. Since x's counter decreases by 1, the difference between x's frequency estimation and x's true frequency, $\hat{f}(x) - f(x)$, also decreases by 1. Once an item becomes monitored, its estimation error can only decrease. The number of decrements due to an unmonitored item is at most $\frac{\epsilon}{2}(I-D)$. Hence for any item, its frequency is underestimated by at most $\frac{\epsilon}{2}(I-D)$ due to the deletion of unmonitored items. As a result, for any item, its frequency can be underestimated by at most $\epsilon(I-D)$ by replacing the $minItem$ and the deletions of the unmonitored items. □

In Theorem 4, we proved that SpaceSaving$^{\pm}$ guarantees that all items' estimated frequencies are off by no more than $\epsilon(I-D)$, i.e., $\forall i, |\hat{f}(i)-f(i)| \leq \epsilon(I-D)$. By reporting all the items with estimated frequency greater than 0, all frequent items must be reported, which is established in Theorem 5.

THEOREM 5. *In the bounded-deletion model, where* $D \leq (1-\frac{1}{\alpha})I$, *SpaceSaving*$^{\pm}$ *solves the frequent items problem using* $O(\frac{\alpha}{\epsilon})$ *space.*

PROOF. Proof by contradiction:
Assume SpaceSaving$^{\pm}$ does not report all frequent items. There must exist a frequent item x that is not reported. Since SpaceSaving$^{\pm}$ reports all the items with estimated frequency greater than 0 as frequent items (recall unmonitored items have estimated frequency of 0), x's estimated frequency must be less than or equal to 0, i.e., $\hat{f}(x) \leq 0$. Moreover, since x is a frequent item, then the exact frequency of x must be greater than or equal to $\epsilon(I-D)$, i.e., $f(x) \geq \epsilon(I-D)$. The difference between x estimated frequency and x exact frequency is greater than or equal to $\epsilon(I-D)$, i.e., $|f(x)-\hat{f}(x)| \geq \epsilon(I-D)$. This leads to a contradiction since it violates the frequency approximation guarantee proved in Theorem 4. □

3.5 An illustration of SpaceSaving$^{\pm}$

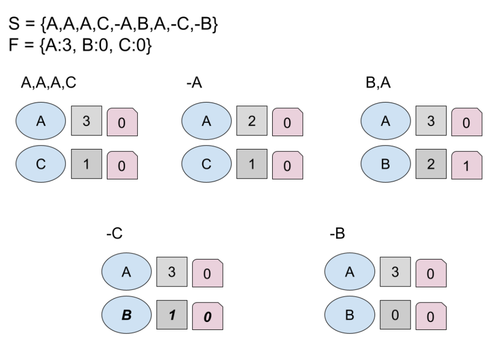

Figure 2: Input Stream consists of 6 insertions and 3 deletions. Each tuple represents $(item, count_{item}, error_{item})$**.**

Consider the same stream illustrated in Section 3.3 in which the stream σ is $(A, A, A, C, -A, B, A, -C, -B)$ where the minus sign indicate a deletion. The corresponding exact frequency of A is 3, while the true frequency of all other items is 0. Consider an instance of SpaceSaving$^{\pm}$ with capacity of 2. The SpaceSaving$^{\pm}$ image after digesting the first 7 items are exactly the same as in the previous example. When the deletion of item C arrives, SpaceSaving$^{\pm}$ does not ignore the deletion of unmonitored item C, and since item B has the largest estimation error, both B's count and B's estimation error are decreased. The final deletion of A decreased A's corresponding count. After processing the stream, the estimated frequency for A and B are 3 and 0 respectively, as shown in Figure 2. The frequency estimations are exact in which $|\hat{f}(A)-f(A)| = 0$ and $|\hat{f}(B)-f(B)| = 0$. With the same bounded-deletion stream and sketch space, Lazy SpaceSaving$^{\pm}$ overestimated the frequency of item B by 1 (Section 3.3), while SpaceSaving$^{\pm}$ is able to further reduce the estimation error to 0. By judiciously handling the deletion of the unmonitored items, SpaceSaving$^{\pm}$ reduces the impact of overestimation and achieves better accuracy.

3.6 Min Heap and Max Heap

SpaceSaving algorithm is usually implemented with a standard min-heap data structure such that the operations that increase the item counts and that remove the minimum item can be performed in logarithmic time [6]. To support the deletion of the unmonitored items, SpaceSaving$^{\pm}$ further needs to find the item with the maximum estimation error and modify the estimation errors efficiently. From these observations, we use two heaps on both the estimated counts and the estimation errors, as underlying data structures. The estimated counts are stored in a min heap, the estimation errors are stored in a max heap, and a dictionary maps each item to the corresponding nodes in these two heaps. Using two heaps and a dictionary with $O(k)$ space, both the minimum count and maximum estimation error can be found in $O(1)$ time; while insertions and deletions can be done in $O(logk)$ time. For example, if the sketch needs to delete an unmonitored item, then the procedure would be as follows: (1) use the dictionary to ascertain that the deletion is performed on an unmonitored item; (2) use the max heap to find the item with maximum estimation error; (3) use the dictionary to find the location of the item with maximum estimation error in the min heap; (4) decrease both its count (min heap) and its estimation error (max heap); (5) percolate it up in min heap and percolate it down in max heap;

4 EVALUATION

This section evaluates the performance of Lazy SpaceSaving$^{\pm}$ and SpaceSaving$^{\pm}$. They are the first deterministic frequency estimation and frequent item algorithms in the bounded-deletion model and make no assumptions on the universe. The experiments aim to identify advantages and disadvantages of Lazy SpaceSaving$^{\pm}$ and SpaceSaving$^{\pm}$ compared to other state-of-the-art sketches such as:

- **CSSS** [29] : The CSSS sketch is the first theoretical algorithm to solve the frequency estimation and frequent item problems in the bounded-deletion model.
- **Count-Min** [5] [17]: Count-Min Sketch operates in the turnstile model and never underestimate frequencies.
- **Count-Median** [6] [12]: Count-Median Sketch operates in the turnstile model and its frequency estimation is unbiased.

4.1 Experimental Setup

We implemented SpaceSaving$^{\pm}$ using the min and max heap data structures described in Section 3.6 in Python. The main distinction from the original SpaceSaving [38] are: (i) support of delete

[5]See https://github.com/rafacarrascosa/countminsketch for implementation detail
[6]See [19] for implementation detail

operations using the Algorithm 3 or Algorithm 4; (ii) the use of a max heap on the estimation errors; and (iii) the overall space complexity is $\frac{\alpha}{\epsilon}$ and the update time complexity is $O(log\frac{\alpha}{\epsilon})$. We also implemented the CSSS sketch as described in [29]. All the experimental metrics are averaged over 5 independent runs. In all experiments, Lazy SpaceSaving$^{\pm}$ and SpaceSaving$^{\pm}$ use the same amount of space, and to align the experiments with the theoretical literature [7, 29], we set the universe size $U = 2^{16}$ and $\delta = U^{-1}$.

4.2 Data Sets

The experimental evaluation is conducted using both synthetic and real world data sets consisting of items that are inserted and deleted. For the synthetic data, we consider three different distributions:

- **Zipf Distribution**: The elements are drawn from a bounded universe and the frequencies of elements follow the Zipf Law [49], in which the frequency of an element with rank R: $f(R,s) = \frac{constant}{R^s}$ where s indicates skewness. Deletions are uniformly chosen from the insertions.
- **Binomial Distribution**: The elements are generated according to the binomial distribution with parameters n and p where p is the probability of success in n independent Bernoulli trials.

In addition to the synthetic data sets, we used the following real world CAIDA Anonymized Internet Trace 2015 Dataset [1].

- **2015 CAIDA Dataset**: The CAIDA dataset is collected from the 'equinixchicago' high-speed monitor. In the experiment, we use 5 disjoint batches of 2 million TCP packets where insertions are the destination IP addresses and deletions are randomly chosen from insertions.

We also conducted experiments by exploring two additional patterns of the data sets:

- **Shuffled**: The insertions are randomly shuffled and the deletions are randomly and uniformly chosen from insertions.
- **Targeted**: The insertions are randomly shuffled and the deletions delete the item with the least frequency.

The metrics used in the experiments are:

- **Mean Squared Error**: The mean squared error (MSE) is the average of the squares of the frequency estimation errors. MSE is a measurement widely used to judge the accuracy of an estimation and serves as an empirical estimation of the variance [16].
- **Recall**: Recall is defined as $\frac{TP}{TP+FN}$ where TP (true positive) is the number of items that are estimated to be frequent and are indeed frequent and FN (false negative) is the number of items that are frequent but not included in the estimations.
- **Precision**: Precision is defined as $\frac{TP}{TP+FP}$ where FP (false positive) is the number of items that are estimated to be frequent but are not frequent.

The experiments are presented in the following two subsections: frequency estimation and frequent item experiments.

4.3 Frequency Estimation Evaluation

In this section, we compare Lazy SpaceSaving$^{\pm}$ and SpaceSaving$^{\pm}$ with state-of-the-art frequency estimation sketches. Our proposed algorithms use $O(\frac{\alpha}{\epsilon})$ space while the Count-Min and Count-Median use $O(\frac{1}{\epsilon}logU)$ space. When $\alpha = logU$, they share the same space and the delete:insert ratio becomes $\frac{logU-1}{logU}$. In addition, these experiments evaluate the accuracy of each sketch using the mean square error (MSE). In MSE figures the x-axis denotes the sketch size while the y-axis depicts the average of the mean square errors. Since the mean square error is strictly positive, the lower y-axis values indicate better accuracy. In the following experiments, we assume all insertions arrive before any deletions into the sketch which is an adversarial pattern as spatial locality is minimized.

4.3.1 Sketch Size. In this experiment, the input data has I insertions and D deletions. The delete:insert ratio is 0.5 and $alpha$ equals to 2. The deletion pattern is either shuffled, randomly chosen from insertions, or targeted delete of the least frequent items. The Zipf and Binomial distributions have $|F|_1 = 10^5$ and the CAIDA dataset has $|F|_1 = 10^6$. This experiment explores the effect of distribution skewness and the space size effect of sketches operating in both the bounded-deletion model and in the turnstile model.

As expected, all sketches share the same pattern: increasing the sketch size leads to decrease in the MSE, shown in Figure 3. All experiments show SpaceSaving$^{\pm}$ has the lowest MSE and best accuracy as the sketch size grows. For the skewed Zipf distribution and CAIDA dataset, SpaceSaving$^{\pm}$ is the clear winner for all sketch sizes, as shown in Figure 3. For the lesser skewed binomial distribution, Count-Median performs competitively compared to SpaceSaving$^{\pm}$; however, SpaceSaving$^{\pm}$ eventually has better accuracy as the sketch size increases, as shown in Figure 3(b,e). The CSSS sketch has accuracy between Count-Median and Count-Min sketches. The Count-Min sketch often overestimates an item's frequency and thus has higher mean square error across all distribution.

The targeted deletion pattern, when the least frequent items are targeted for deletions, leads to a slight decrease in MSE across all distributions for Count-Min. The targeted delete pattern decreases the cardinality of F, increases the overall skewness, and hence heavy hitter items become more dominant and all sketches are able to capture the overall change and have less mean square error.

4.3.2 Delete:Insert Ratio and α. Sketches in the bounded-deletion model have space complexity dependent on parameter α, which upper bounds the delete:insert ratio. With higher delete:insert ratio, these sketches need to increase their sketch space to tolerate the increase in deletions in order to deliver the same guarantee. In this experiment, we fixed the sketch space to $10^3 logU$ bits and fixed the input stream length to one million. The x-axis represents different delete:insert ratio, and the y-axis is the mean squared error averaged over 5 independent runs, as shown in Figure 4.

As expected, the accuracy of Lazy SpaceSaving$^{\pm}$ and CSSS depends on α and their MSE increases as the delete:insert ratio increases. The more interesting result is that SpaceSaving$^{\pm}$'s MSE decreases when the delete:insert ratio is less or equal to 0.9. Moreover, for a universe of size 2^{16}, SpaceSaving$^{\pm}$ provides MSE less than CSSS, Count-Min, and Count-Median even if the delete:insert ratio is as high as 0.9375, which is $\frac{logU-1}{logU}$, while using the same amount of space, as shown by the right most values in Figure 4. By handling the deletion of unmonitored items judiciously, SpaceSaving$^{\pm}$'s frequency estimation is more robust to the increase in deletions than other algorithms in the bounded-deletion model. For sketches

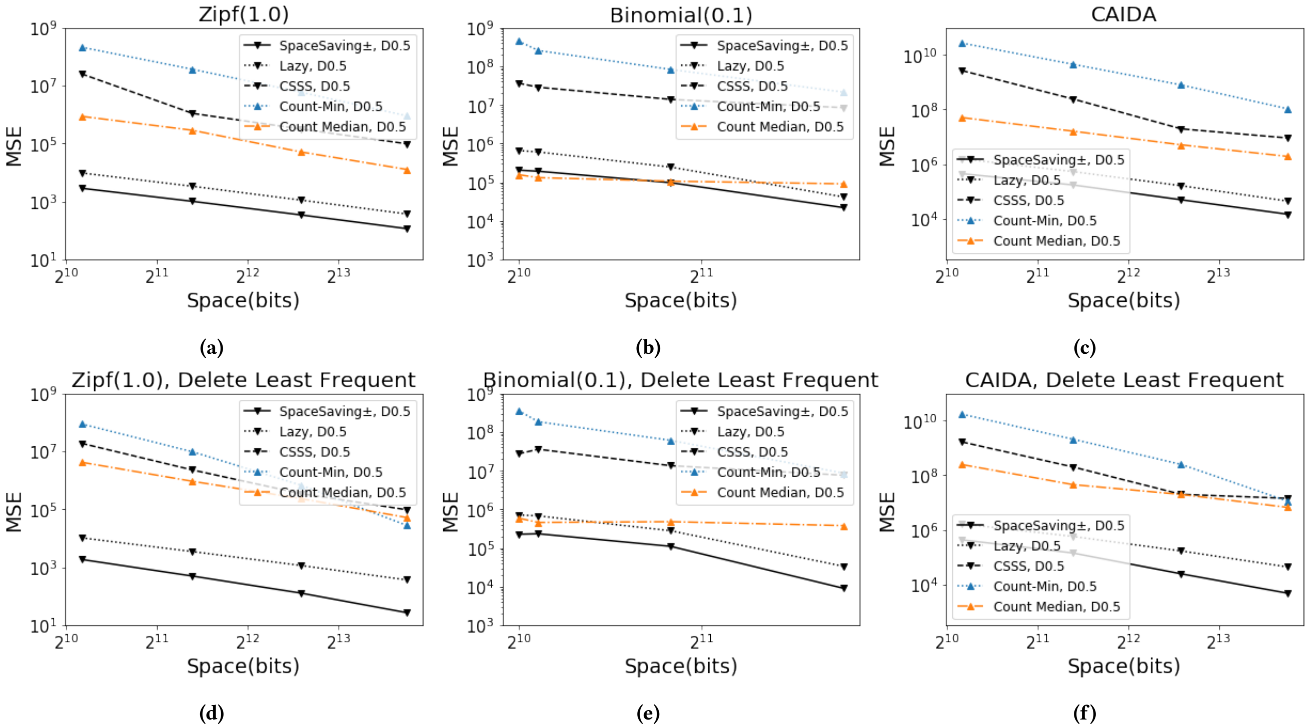

Figure 3: Trade-off between space and accuracy on various data distributions and different patterns

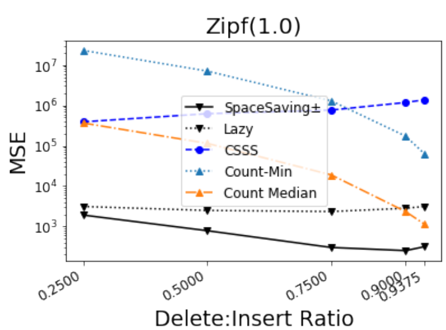

Figure 4: Varying delete:insert ratio.

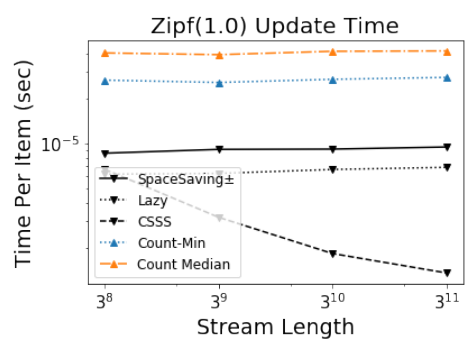

Figure 5: Update times for Sketches

that operate in the turnstile model, the MSE of Count-Min and Count-Median decreases as the delete:insert ratio increases, since more deletions reduce the number of hash collisions and reduce the amount of over counting in each bucket. If the universe size increases, the performance of linear sketches will further decrease, whereas the data-driven SpaceSaving$^{\pm}$ has no dependency on the universe, and can provide accurate estimations even in the extreme case of unbounded universe.

4.3.3 Update Time. In Figure 5 , the x-axis is the stream length and the y-axis is the average latency in second per item over 5 independent runs. The input is a shuffled Zipf distribution and the delete:insert ratio is 0.5. All sketches use $10^3 logU$ bits. As shown in Figure 5, Lazy SpaceSaving$^{\pm}$ has slightly less update time than SpaceSaving$^{\pm}$. The lazy approach ignores deletions of unmonitored items and achieves better latency. CSSS sketch update time decreases as the stream length grows because it performs sampling to obtain $O(\frac{\alpha logU}{\epsilon})$ samples and runs Count-Median sketch on the samples. As the stream length increases the sample size increases at a slower pace, and the average update time per item decreases. Count-Min and Count-Median have update times depend on the universe size where a larger universe size will further increase the update time. Since Count-Median performs more hashes than Count-Min, Count-Median requires more update time than Count-Min. While randomized CSSS has fast update time, our algorithms are deterministic and provide very accurate approximations.

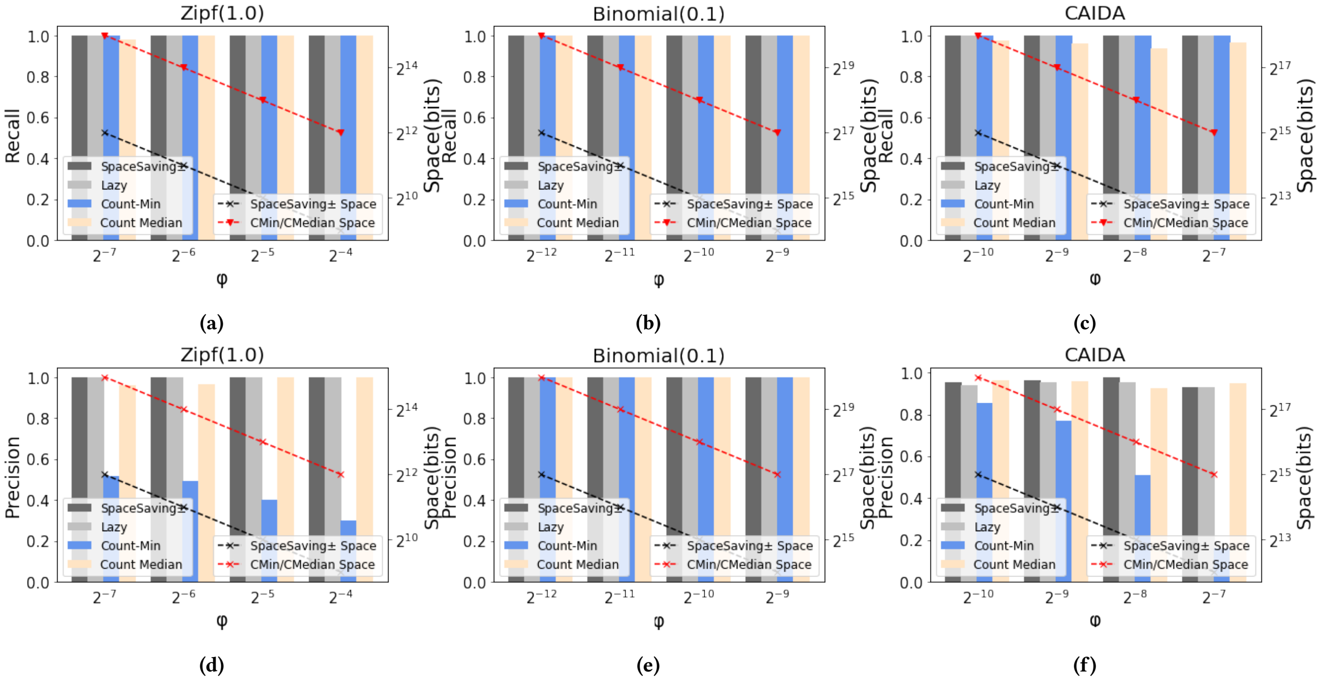

Figure 6: Recall and Precision Comparison

4.4 Frequent Items Evaluation

In this section, we compare the recall and precision of our proposed algorithms with state-of-the-art sketches for identifying frequent items. All experiments in this section have delete:insert ratio of 0.5, $\alpha = 2$, and all insertions arrive before any deletions. The left y-axis depicts either the average recall or average precision over 5 independent runs: higher y-axis values indicate better recall or precision. The right y-axis denotes the space used for each sketch where Lazy SpaceSaving$^{\pm}$ and SpaceSaving$^{\pm}$ use $\frac{\alpha}{\epsilon} logU$ bits; Count-Min and Count-Median use $\frac{1}{\epsilon} log^2 U$ bits. Each sketch queries all potential items and then reports items with estimated frequency greater than or equal to $\phi|F|_1$ as the frequent items. In addition, the following experiments do not compare with the CSSS sketch. Although CSSS can solve the frequent item problem, CSSS is more of theoretical interest since it reduces the size of each counter from $O(logU)$ bits to $O(log(\alpha))$ bits but in practice, it requires a lot more space to solve the frequent item problem. More specifically, the sketch size increases by 192 times, which implies the universe is powered by 192 times. The space increase is more significant than the space saved by reducing the number of bits per counter.

4.4.1 Recall. In these experiments, we compare the recall among Lazy SpaceSaveing$^{\pm}$, SpaceSaving$^{\pm}$, Count-Min and Count-Median. In Figure 6 (a), (b), and (c), the x-axis represents different frequent items threshold ϕ in which frequent items have frequency greater than or equal to $\phi|F|_1$. The right y-axis denotes the space used for each sketches in which Lazy SpaceSaving$^{\pm}$ and SpaceSaving$^{\pm}$ use $\frac{\alpha}{\epsilon} logU$ bits; Count-Min and Count-Median use $\frac{1}{\epsilon} log^2 U$ bits. The sketch space increases as ϕ decreases. The left y-axis is the recall ratio. As expected, Lazy SpaceSaving$^{\pm}$ and Count-Min sketches have 100% recall across all distributions, since they never underestimate the frequent item's frequency. The Count-Median sketch may sometimes underestimate the frequency and thus does not always achieve 100% recall, as shown in Figure 6 (c). In the proof of Theorem 5, SpaceSaving$^{\pm}$ needs to report all items with frequency greater than 0 to identify all frequent items and achieve 100% recall. In this experiment, SpaceSaving$^{\pm}$ reports items with frequency larger than $\phi|F|_1$. Since it might underestimate an item's frequency, the recall rate might not be 100%. However, in these experiments, SpaceSaving$^{\pm}$ still achieves 100% recall across all distributions and thus indicates SpaceSaving$^{\pm}$ rarely underestimates.

4.4.2 Precision. In this subsection, we compare the precision among Lazy SpaceSaveing$^{\pm}$, SpaceSaving$^{\pm}$, Count-Min and Count-Median. In Figure 6 (d), (e), and (f), the x-axis represents the different frequent items threshold ϕ. The right y-axis denotes the space budget in which Lazy SpaceSaving$^{\pm}$ and SpaceSaving$^{\pm}$ use $\frac{\alpha}{\epsilon} logU$ bits; Count-Min and Count-Median use $\frac{1}{\epsilon} log^2 U$ bits. The sketch space increases as ϕ decreases. The left y-axis is the precision ratio. Lazy SpaceSaving$^{\pm}$, SpaceSaving$^{\pm}$ and Count-Median have above 90% precision for all ϕ and distributions. Since Lazy SpaceSaving$^{\pm}$ sometimes overestimates an item's frequency, a few items' frequency are overestimated and hence they may be falsely classified as frequent items. SpaceSaving$^{\pm}$ judiciously handles the deletion and achieves very high precision across all distributions using minimal space. Count-Min often overestimates items' frequencies and many items are incorrectly classified as frequent.

5 QUANTILE SKETCH

In this section, we demonstrate that SpaceSaving$^\pm$ can be easily integrated with prior protocols [17, 44] to solve the quantile approximation problem. We propose Dyadic SpaceSaving$^\pm$ (DSS$^\pm$), the first deterministic quantile sketch in the bounded-deletion model. Dyadic SpaceSaving$^\pm$ sketch is a universe-driven algorithm that accurately approximates quantiles with strong guarantees.

5.1 The Quantiles Problem

The rank of an element x is the total number of elements that are less than or equal to x, denoted as $R(x)$. The quantile of an element x is defined as $R(x)/|F|_1$ where F is the frequency vector. The most familiar quantile value is 0.5 also known as *median*. *Deterministic* ϵ approximation quantile algorithms [25, 41] take as input a precision value ϵ and an item such that the approximated rank has at most $\epsilon|F|_1$ additive error. The *randomized* quantile algorithms provide a weaker guarantee in which the approximated rank of an item has at most $\epsilon|F|_1$ additive error with high probability [28, 33, 37].

Recently, Zhao et al. [48] proposed the first randomized quantile sketch KLL$^\pm$ using $O(\frac{\alpha^{1.5}}{\epsilon}\ log^2 log\frac{1}{\epsilon})$ space in the bounded deletion model by generalizing the KLL [33] from the insertion-only model. The first sketch to summarize quantiles in the turnstile model is the Random Subset Sums (RSS) proposed by Gilbert et al. [24]. RSS is a universe driven algorithm, which assume input are drawn from a bounded universe and maintain attributes over the bounded universe [14]. RSS breaks down the bounded universe into dyadic intervals and maintains frequency estimations for each interval. Recall, dyadic intervals are in the form of $[i2^j, (i+1)2^j - 1]$ for $j \in log_2 U$ and any constant i, such that any ranges can be decomposed into at most $log_2 U$ disjoint dyadic ranges [15]. Cormode et al. [17] proposed the Dyadic Count-Min (DCM) which replaces the frequency estimation sketch for each dyadic interval with a Count-Min, and hence improves the overall space complexity to $O(\frac{1}{\epsilon}\log^2 U \log(\frac{\log U}{\epsilon})))$ and update time to $O(\log U \log(\frac{\log U}{\epsilon}))$. Then, Wang et al. [44] proposed the Dyadic Count-Median (DCS) which replaces Count-Min with Count-Median [12] to further improve the space complexity to $O(\frac{1}{\epsilon}\log^{1.5} U \log^{1.5}(\frac{\log U}{\epsilon})))$, while using the same update time complexity as DCM.

5.2 DSS$^\pm$: A Deterministic Quantile Sketch

We propose the Dyadic SpaceSaving$^\pm$ (DSS$^\pm$) to solve deterministic quantile approximation in the bounded-deletion model. Inspired by the previous algorithms, we observe that by replacing the frequency estimation sketch in each dyadic layer with a SpaceSaving$^\pm$ of space $O(\frac{\alpha}{\epsilon} logU)$ solves the quantile approximation in the bounded-deletion model. Any range can be decomposed into at most $logU$ dyadic intervals [15]. Since SpaceSaving$^\pm$ with $O(\frac{\alpha}{\epsilon} logU)$ space ensures that the frequency estimation has at most $\frac{\epsilon(I-D)}{logU}$ additive error and by summing up at most $logU$ frequencies, the approximated rank has at most $\epsilon(I-D)$ additive error and the approximated quantile has at most ϵ error. To update the DSS$^\pm$ quantile sketch with an item x: for each $logU$ layers, x is mapped to an element in that layer and increments the corresponding element's frequency, as shown in Algorithm 5. The rank information of an item can be calculated by summing $O(logU)$ number of subset sums, as shown in Algorithm 6. Therefore, the Dyadic SpaceSaving$^\pm$ sketch requires $O(\frac{\alpha}{\epsilon} log^2 U)$ space with update time $O(logU log\frac{\alpha logU}{\epsilon})$. The quantile experiments comparing DSS$^\pm$, KLL$^\pm$ and DCS are shown in [47].

Algorithm 5: DSS$^\pm$ Update(x,1)

```
for h from 0 to logU do
    DSS±[h].update(x, 1);
    x= x/2;
end
```

Algorithm 6: DSS$^\pm$ Query(x)

```
Rank = 0;
for h from 0 to logU do
    if x is odd then
        Rank = Rank + DSS±[h].query(x);
    x= x/2;
end
return Rank;
```

6 CONCLUSION

Frequency estimation and frequent items are two important problems in data stream research, and have significant impact for real world systems. Over the past decades of research, many algorithms have been proposed for the insertion-only and the turnstile models. In this work, we propose data-driven deterministic SpaceSaving$^\pm$ sketches to accurately approximate item frequency and report heavy hitter items in the bounded-deletion model. To our knowledge, Lazy SpaceSaving$^\pm$ and SpaceSaving$^\pm$ are the first deterministic algorithms to solve these two problems in the bounded-deletion model and they make no assumption on the universe. The experimental evaluations of SpaceSaving$^\pm$ highlight that it has the best frequency estimation accuracy among other state-of-the-art sketches, and requires the least space to provide strong guarantees. We also demonstrate that implementing SpaceSaving$^\pm$ with the min and max heap approach provides fast update time. Furthermore, the experiments showcase that SpaceSaving$^\pm$ has very high recall and precision rates across a range of data distributions. These characteristics of SpaceSaving$^\pm$ make it a practical choice for real world applications. Finally, by leveraging SpaceSaving$^\pm$ and dyadic intervals over bounded universe, we proposed the first deterministic quantile sketch in the bounded-deletion model. Our analysis clearly demonstrates that overall, for an unbounded universe or for practical delete:insert ratios below $\frac{logU-1}{logU}$ (e.g., for a realistic universe size of U=2^{16}, a ratio of .93 and for U=2^{32}, a ratio of .96), SpaceSaving$^\pm$ is the best algorithm to use and solves several major problems with strong guarantees in a unified algorithm.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their valuable feedback and also thank Rajesh Jayaram and Dan Qiao for helpful discussions. This work is funded in part by NSF grants CNS-1703560 and CNS-1815733.

REFERENCES

[1] [n.d.]. Anonymized Internet Traces 2015. https://catalog.caida.org/details/dataset/passive_2015_pcap. Accessed: 2021-11-5.

[2] Pankaj K Agarwal, Graham Cormode, Zengfeng Huang, Jeff Phillips, Zhewei Wei, and Ke Yi. 2012. Mergeable summaries. In *Proceedings of the 31st ACM SIGMOD-SIGACT-SIGAI symposium on Principles of Database Systems*. 23–34.

[3] Noga Alon, Yossi Matias, and Mario Szegedy. 1999. The space complexity of approximating the frequency moments. *Journal of Computer and system sciences* 58, 1 (1999), 137–147.

[4] Ziv Bar-Yossef, Thathachar S Jayram, Ravi Kumar, and D Sivakumar. 2004. An information statistics approach to data stream and communication complexity. *J. Comput. System Sci.* 68, 4 (2004), 702–732.

[5] Ran Ben Basat, Xiaoqi Chen, Gil Einziger, and Ori Rottenstreich. 2020. Designing heavy-hitter detection algorithms for programmable switches. *IEEE/ACM Transactions on Networking* 28, 3 (2020), 1172–1185.

[6] Radu Berinde, Piotr Indyk, Graham Cormode, and Martin J Strauss. 2010. Space-optimal heavy hitters with strong error bounds. *ACM Transactions on Database Systems (TODS)* 35, 4 (2010), 1–28.

[7] Arnab Bhattacharyya, Palash Dey, and David P Woodruff. 2018. An optimal algorithm for 1-heavy hitters in insertion streams and related problems. *ACM Transactions on Algorithms (TALG)* 15, 1 (2018), 1–27.

[8] Burton H Bloom. 1970. Space/time trade-offs in hash coding with allowable errors. *Commun. ACM* 13, 7 (1970), 422–426.

[9] Robert S Boyer and J Strother Moore. 1991. MJRTY—a fast majority vote algorithm. In *Automated Reasoning*. Springer, 105–117.

[10] Mark Braverman, Sumegha Garg, and David P Woodruff. 2020. The coin problem with applications to data streams. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, 318–329.

[11] Vladimir Braverman, Stephen R Chestnut, Nikita Ivkin, Jelani Nelson, Zhengyu Wang, and David P Woodruff. 2017. BPTree: an 2 heavy hitters algorithm using constant memory. In *Proceedings of the 36th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*. 361–376.

[12] Moses Charikar, Kevin Chen, and Martin Farach-Colton. 2002. Finding frequent items in data streams. In *International Colloquium on Automata, Languages, and Programming*. Springer, 693–703.

[13] Graham Cormode and Marios Hadjieleftheriou. 2008. Finding frequent items in data streams. *Proceedings of the VLDB Endowment* 1, 2 (2008), 1530–1541.

[14] Graham Cormode, Theodore Johnson, Flip Korn, Shan Muthukrishnan, Oliver Spatscheck, and Divesh Srivastava. 2004. Holistic UDAFs at streaming speeds. In *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*. 35–46.

[15] Graham Cormode, Tejas Kulkarni, and Divesh Srivastava. 2019. Answering range queries under local differential privacy. *Proceedings of the VLDB Endowment* 12, 10 (2019), 1126–1138.

[16] Graham Cormode, Samuel Maddock, and Carsten Maple. 2021. Frequency Estimation under Local Differential Privacy [Experiments, Analysis and Benchmarks]. *arXiv preprint arXiv:2103.16640* (2021).

[17] Graham Cormode and Shan Muthukrishnan. 2005. An improved data stream summary: the count-min sketch and its applications. *Journal of Algorithms* 55, 1 (2005), 58–75.

[18] Graham Cormode and Shan Muthukrishnan. 2005. What's hot and what's not: tracking most frequent items dynamically. *ACM Transactions on Database Systems (TODS)* 30, 1 (2005), 249–278.

[19] Graham Cormode and Ke Yi. 2020. *Small Summaries for Big Data*. Cambridge University Press.

[20] Sudipto Das, Shyam Antony, Divyakant Agrawal, and Amr El Abbadi. 2009. Cots: A scalable framework for parallelizing frequency counting over data streams. In *2009 IEEE 25th International Conference on Data Engineering*. IEEE, 1323–1326.

[21] Erik D Demaine, Alejandro López-Ortiz, and J Ian Munro. 2002. Frequency estimation of internet packet streams with limited space. In *European Symposium on Algorithms*. Springer, 348–360.

[22] Min Fang, Narayanan Shivakumar, Hector Garcia-Molina, Rajeev Motwani, and Jeffrey D Ullman. 1999. Computing Iceberg Queries Efficiently.. In *Internaational Conference on Very Large Databases (VLDB'98), New York, August 1998*. Stanford InfoLab.

[23] Philippe Flajolet, Éric Fusy, Olivier Gandouet, and Frédéric Meunier. 2007. Hyperloglog: the analysis of a near-optimal cardinality estimation algorithm.

[24] Anna C Gilbert, Yannis Kotidis, S Muthukrishnan, and Martin J Strauss. 2002. How to summarize the universe: Dynamic maintenance of quantiles. In *VLDB'02: Proceedings of the 28th International Conference on Very Large Databases*. Elsevier, 454–465.

[25] Michael Greenwald and Sanjeev Khanna. 2001. Space-efficient online computation of quantile summaries. *ACM SIGMOD Record* 30, 2 (2001), 58–66.

[26] Şule Gündüz and M Tamer Özsu. 2003. A web page prediction model based on click-stream tree representation of user behavior. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*. 535–540.

[27] Rob Harrison, Shir Landau Feibish, Arpit Gupta, Ross Teixeira, S Muthukrishnan, and Jennifer Rexford. 2020. Carpe elephants: Seize the global heavy hitters. In *Proceedings of the Workshop on Secure Programmable Network Infrastructure*. 15–21.

[28] Nikita Ivkin, Edo Liberty, Kevin Lang, Zohar Karnin, and Vladimir Braverman. 2019. Streaming Quantiles Algorithms with Small Space and Update Time. *arXiv preprint arXiv:1907.00236* (2019).

[29] Rajesh Jayaram and David P Woodruff. 2018. Data streams with bounded deletions. In *Proceedings of the 37th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*. 341–354.

[30] Cheqing Jin, Weining Qian, Chaofeng Sha, Jeffrey X Yu, and Aoying Zhou. 2003. Dynamically maintaining frequent items over a data stream. In *Proceedings of the twelfth international conference on Information and knowledge management*. 287–294.

[31] Hossein Jowhari, Mert Sağlam, and Gábor Tardos. 2011. Tight bounds for lp samplers, finding duplicates in streams, and related problems. In *Proceedings of the thirtieth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*. 49–58.

[32] John Kallaugher and Eric Price. 2020. Separations and equivalences between turnstile streaming and linear sketching. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*. 1223–1236.

[33] Zohar Karnin, Kevin Lang, and Edo Liberty. 2016. Optimal quantile approximation in streams. In *2016 ieee 57th annual symposium on foundations of computer science (focs)*. IEEE, 71–78.

[34] Richard M Karp, Scott Shenker, and Christos H Papadimitriou. 2003. A simple algorithm for finding frequent elements in streams and bags. *ACM Transactions on Database Systems (TODS)* 28, 1 (2003), 51–55.

[35] Nishad Manerikar and Themis Palpanas. 2009. Frequent items in streaming data: An experimental evaluation of the state-of-the-art. *Data & Knowledge Engineering* 68, 4 (2009), 415–430.

[36] Gurmeet Singh Manku and Rajeev Motwani. 2002. Approximate frequency counts over data streams. In *VLDB'02: Proceedings of the 28th International Conference on Very Large Databases*. Elsevier, 346–357.

[37] Gurmeet Singh Manku, Sridhar Rajagopalan, and Bruce G Lindsay. 1998. Approximate medians and other quantiles in one pass and with limited memory. *ACM SIGMOD Record* 27, 2 (1998), 426–435.

[38] Ahmed Metwally, Divyakant Agrawal, and Amr El Abbadi. 2005. Efficient computation of frequent and top-k elements in data streams. In *International conference on database theory*. Springer, 398–412.

[39] Jayadev Misra and David Gries. 1982. Finding repeated elements. *Science of computer programming* 2, 2 (1982), 143–152.

[40] Rob Pike, Sean Dorward, Robert Griesemer, and Sean Quinlan. 2005. Interpreting the data: Parallel analysis with Sawzall. *Scientific Programming* 13, 4 (2005), 277–298.

[41] Nisheeth Shrivastava, Chiranjeeb Buragohain, Divyakant Agrawal, and Subhash Suri. 2004. Medians and beyond: new aggregation techniques for sensor networks. In *Proceedings of the 2nd international conference on Embedded networked sensor systems*. 239–249.

[42] Vibhaalakshmi Sivaraman, Srinivas Narayana, Ori Rottenstreich, Shan Muthukrishnan, and Jennifer Rexford. 2017. Heavy-hitter detection entirely in the data plane. In *Proceedings of the Symposium on SDN Research*. 164–176.

[43] Daniel Ting. 2018. Data sketches for disaggregated subset sum and frequent item estimation. In *Proceedings of the 2018 International Conference on Management of Data*. 1129–1140.

[44] Lu Wang, Ge Luo, Ke Yi, and Graham Cormode. 2013. Quantiles over data streams: an experimental study. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*. 737–748.

[45] Victor Zakhary, Lawrence Lim, Divyakant Agrawal, and Amr El Abbadi. 2020. CoT: Decentralized elastic caches for cloud environments. *arXiv preprint arXiv:2006.08067* (2020).

[46] Yinda Zhang, Zaoxing Liu, Ruixin Wang, Tong Yang, Jizhou Li, Ruijie Miao, Peng Liu, Ruwen Zhang, and Junchen Jiang. 2021. CocoSketch: high-performance sketch-based measurement over arbitrary partial key query. In *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*. 207–222.

[47] Fuheng Zhao, Divyakant Agrawal, Amr El Abbadi, and Ahmed Metwally. 2021. SpaceSaving$^{\pm}$: An Optimal Algorithm for Frequency Estimation and Frequent items in the Bounded Deletion Model. arXiv:2112.03462 [cs.DB]

[48] Fuheng Zhao, Sujaya Maiyya, Ryan Wiener, Divyakant Agrawal, and Amr El Abbadi. 2021. KLL±: Approximate quantile sketches over dynamic datasets. *Proceedings of the VLDB Endowment* 14, 7 (2021), 1215–1227.

[49] George Kingsley Zipf. 2016. *Human behavior and the principle of least effort: An introduction to human ecology*. Ravenio Books.