Randomized Algorithms.
Lecture Notes on Heavy Hitters.
Kasper Green Larsen.

1 Heavy Hitters

In this note, we extend the results in the CountSketch lecture note.

We will only consider the strict turnstile model, where we are maintaining estimates of some underlying frequency vector f. The vector f has coordinates indexed by integers in $[U]$ for some universe size U (think e.g. of U as 2^{32} or 2^{64}). In the strict turnstile streaming model, we receive updates one at a time. An update is given by a pair (i, Δ) and has the effect of updating $f_i \leftarrow f_i + \Delta$ for $i \in [U]$ and $\Delta \in \{-M, -M+1, \dots, M-1, M\}$. We assume for simplicity that M is bounded by a polynomial in n, i.e. $M = n^{O(1)}$ and thus a weight change fits in $O(\log n)$ bits and thus in a machine word. Furthermore, in the *strict* turnstile model, we are promised that at any given time, every frequency satisfies $f_i \geq 0$.

In the CountSketch lecture note, we saw how to use the CountMin sketch to support *frequency estimation queries*, i.e. we saw a data structure using $O(\varepsilon^{-1} \lg(1/\delta))$ words of space that upon receiving an index $i \in [U]$ as query, can return a value $\hat{f}_i$ that with probability at least $1-\delta$ satisfies $f_i \leq \hat{f}_i \leq f_i + \varepsilon \|f\|_1$ where $\|f\|_1 = \sum_i |f_i|$. The update time of the CountMin sketch was $O(\lg(1/\delta))$ and the query time also $O(\lg(1/\delta))$.

In many applications, our main concern is not supporting frequency estimation queries, but rather finding so-called heavy hitters. An ε heavy hitter is an index i such that $f_i \geq \varepsilon \|f\|_1$. In fact, the Count-Min sketch was introduced with the goal of finding heavy hitters but more ideas are needed. This note presents the original solution to finding heavy hitters, due to Cormode and Muthukrishnan [CM05], as well as an improvement due to Larsen, Nelson, Nguyen and Thorup [LNNT16].

Problem Formulation. We first give an exact formulation of the problem of reporting the heavy hitters. In this problem, we still receive frequency updates as above. However, there is just one query. To answer a query, we must produce a list $\mathcal{L}$ satisfying the following:

- All ε heavy hitters are in $\mathcal{L}$, i.e. all indices i with $f_i \geq \varepsilon \|f\|_1$ are in $\mathcal{L}$.
- No index i with $f_i < (\varepsilon/2)\|f\|_1$ is in $\mathcal{L}$.

The sketch/data structure is allowed to fail with probability δ. Note that we have no requirements on indices i with $(\varepsilon/2)\|f\|_1 \leq f_i < \varepsilon\|f\|_1$. This is because an index i with $f_i = \varepsilon\|f\|_1 - 1$ isn't an ε heavy hitter, but if we didn't allow ourselves to put such an i in the output list, we would almost certainly have to keep an exact count for f_i. Thus we have some slack in the definition.

1.1 Solution of Cormode and Muthukrishnan

Assume for simplicity that U is a power of 2. Imagine a complete binary tree on top of the universe, i.e. a complete tree $\mathcal{T}$ having one leaf per index $i \in U$. Such a tree has $\lg U + 1$ levels and we denote the level of the leaves by level 0 and the root's level by $\lg U$. For a node $v \in \mathcal{T}$, let g_v denote the sum of f_j's over all j corresponding to leaves in the subtree rooted at v. Observe that for leaf nodes v, g_v is simply the frequency f_v of v.

We can think of the list of values g_v for the $U/2^i$ nodes at level i as forming another frequency vector on a universe of size $U/2^i$. We use f^i to denote this vector of frequencies for the i'th level of the tree. It is not too hard to convince one self that $f^i_j = \sum_{h=0}^{2^i-1} f_{j2^i+h}$. Of course, the frequency for a node is also just the sum of the frequencies of its two children.

To keep things simple, we focus on the case where we are aiming at an error probability δ of $1/U$, i.e. polynomially small in the universe size. The data structure is as follows: For every level i of $\mathcal{T}$, we store a CountMin sketch on f^i having error probability $\delta' = 1/|U|^2$ and approximation $\varepsilon' = \varepsilon/2$. On an update

(j, Δ), we conceptually add Δ to the frequency of all nodes on the root-to-leaf path leading to the leaf corresponding to j. We see that this can be done by adding Δ to $f^i_{\lfloor j/2^i \rfloor}$ for $i = 0, \dots, \lg U$. Thus an update makes $O(\lg U)$ updates on CountMin data structures, each with error probability $1/U^2$ thus the total update time is $O(\lg^2 U)$. Similarly, we can get a space usage of $O(\varepsilon^{-1} \lg^2 U)$.

The query algorithm is as follows: Start at the root node r. When visiting a node v at level i, if $i = 0$ (i.e. we are at a leaf), add v to the output list. Otherwise, v is an internal node at some level $i > 0$. Query the CountMin sketch on the vector f^{i-1} to obtain estimates of the frequencies stored at the two children of v. Recurse into those children for which the estimate is at least $\varepsilon \|f\|_1$.

To see that this query algorithm works, first define a node v in $\mathcal{T}$ to be heavy if its frequency is at least $(\varepsilon/2)\|f\|_1$. Otherwise, call it light. Since we are in the strict turnstile model and all frequencies in f are non-negative, it follows that $\|f\|_1 = \|f^0\|_1 = \cdots = \|f^{\lg U}\|_1$. Thus we observe the following:

Observation 1. *For any level i in $\mathcal{T}$, there are no more than $2/\varepsilon$ heavy nodes.*

Proof. If we had more than $2/\varepsilon$ heavy nodes at some level i, then we would have that $\|f^i\|_1 > (2/\varepsilon)(\varepsilon/2)\|f\|_1 = \|f\|_1 = \|f^i\|_1$, a contradiction. □

Observation 2. *If i is the index of an ε heavy hitter, then all nodes on the root-to-leaf path leading to the leaf corresponding to i are heavy.*

Proof. Since all frequencies are non-negative, it follows that the frequency of an internal node is always at least as large as any of its children's. □

By Observation 2, the list $\mathcal{L}$ produced by the query algorithm always contains all the heavy hitters. Now let i be an index with $f_i < (\varepsilon/2)\|f\|_1$. Note that the leaf v corresponding to i is light. It follows that if i is added to $\mathcal{L}$, then the query algorithm visited v's parent and made an estimate of v's frequency that was off by more than $(\varepsilon/2)\|f\|_1 = \varepsilon'\|f\|_1$. This happens with probability no more than $1/U^2$ by our setting of parameters for the CountMin sketches. We can thus union bound over all light leaves and conclude that none of them are added to $\mathcal{L}$ with probability at least $1 - 1/U$.

To analyze the query time, observe that if no esimate made by a CountMin sketch fails (which is true with probability about $1 - 1/U$), then we only visit heavy nodes. By Observation 1, there are $O(\varepsilon^{-1} \lg U)$ heavy nodes in $\mathcal{T}$. Thus the query time is $O(\varepsilon^{-1} \lg^2 U)$ since we make two queries to CountMin sketches with error probability $1/U^2$ in each.

Note that if we want the query time to be worst case, we can simply abort with an empty output list in case we visit more than $2\varepsilon^{-1}(\lg U + 1)$ nodes in $\mathcal{T}$ since this means that at least one light node was visited, and this happens with probability no more than about $1/U$.

To summarize, we have a data structure with worst case update time $O(\lg^2 U)$, worst case query time $O(\varepsilon^{-1} \lg^2 U)$ and space $O(\varepsilon^{-1} \lg^2 U)$.

1.2 Solution of Larsen, Nelson, Nguyen and Thorup

This solution improves on the above by a factor $\lg U$ in all parameters at the cost of moving to expected query time. The change is surprisingly simple: Keep the exact same data structure as above, except that all the CountMin sketches stored for levels $i > 0$ are replaced by CountMin sketches with error probability $1/4$ instead of $1/U^2$. The CountMin sketch at the leaves still has error probability $1/U^2$. The query algorithm and update algorithm remains the same.

It is clear that space usage is now $O(\varepsilon^{-1} \lg U)$ as each internal level now uses space only $O(\varepsilon^{-1})$. Similarly, the update time drops to $O(\lg U)$ because each internal level costs $O(1)$ to update. For the correctness probability, note that the argument from the previous section only argues about queries made to the CountMin sketch at level 0, thus the correctness probability still holds. The only thing that might have changed is the query time due to more nodes in $\mathcal{T}$ being visited. In fact, it will be extremely unlikely that we don't visit any light nodes in $\mathcal{T}$. Our task has thus changed to bounding the expected number of light nodes visited.

To this end, define for a light node v its light depth $d(v)$ as the distance to its nearest heavy ancestor in $\mathcal{T}$ (recall that being light and heavy has nothing to do with the hash functions etc. used by the CountMin sketches, it only depends on the vector f). Consider first all light nodes at levels $i \geq 2$. For notational convenience, let V be the set of such nodes. If we visit such a node, we spend $O(1)$ time making two queries to a CountMin sketch. Thus if we define an indicator random variable X_v for each such node $v \in V$, taking the value 1 if v is visited by the query algorithm, and 0 otherwise, then the expected time spend on visiting all light nodes $v \in V$ is $O(\mathbb{E}[\sum_{v \in V} X_v]) = O(\sum_{v \in V} \mathbb{E}[X_v]) = O(\sum_{v \in V} \Pr[v \text{ is visited}])$.

Thus we need to bound $\Pr[v \text{ is visited}]$. For this, the trick is to use the light depth. More precisely, let $V_i \subseteq V$ be the set of nodes in V with light depth i for $i = 1, \dots, \lg U$. Then we have $O(\sum_{v \in V} \Pr[v \text{ is visited}]) = O(\sum_{i=1}^{\lg U} \sum_{v \in V_i} \Pr[v \text{ is visited}])$. So let $v \in V_i$ and consider $\Pr[v \text{ is visited}]$. For v to be visited, it must be the case that the query made in every single ancestor up to (and including) its nearest heavy ancestor failed, i.e. returned an estimate that was off by at least $\varepsilon' \|f\|_1$. Since these are queries to $d(v) = i$ independent CountMin data structures with failure probability $1/4$ each, the probability they all fail is at most $1/4^i$. Hence we have that $O(\sum_{i=1}^{\lg U} \sum_{v \in V_i} \Pr[v \text{ is visited}]) = O(\sum_{i=1}^{\lg U} \sum_{v \in V_i} 4^{-i})$. Next we bound $|V_i|$. The trick here is to examine a heavy node. Any heavy node can be the nearest heavy ancestor of at most 2^i light nodes with light depth i. This is simply because any node has no more than 2^i descendants i levels down. By Observation 1, there are no more than $O(\varepsilon^{-1} \lg U)$ heavy nodes in $\mathcal{T}$ and therefore we have $|V_i| \leq O(2^i \varepsilon^{-1} \lg U)$. We thus conclude that: $O(\mathbb{E}[\sum_{v \in V} X_v]) = O(\sum_{i=1}^{\lg U} \sum_{v \in V_i} 4^{-i}) = O(\sum_{i=1}^{\lg U} 2^i 4^{-i} \varepsilon^{-1} \lg U) = O(\varepsilon^{-1} \lg U \sum_{i=1}^{\lg U} 2^{-i}) = O(\varepsilon^{-1} \lg U)$ as claimed.

What remains is to bound the cost of the queries made in light nodes at level $i = 1$. Recall that these query the data structure at level 0 which had an error probability of $1/U^2$, and thus the cost of visiting such a light node is $O(\lg U)$ rather than $O(1)$. If we this time define V_i as the set of light nodes at level 1 that have a light depth of i, then by argument similar to above, the expected time spend visiting nodes $v \in V_i$ is $O(\sum_{i=1}^{\lg U} |V_i| 4^{-i} \lg U)$. Fortunately, the new sets V_i are a factor $\lg U$ smaller than before. To see this, note that any node in V_i must have its nearest heavy ancestor at level exactly $1 + i$. But Observation 1 tells us that there are only $O(\varepsilon^{-1})$ heavy nodes at level $i + 1$, and each such node has only 2^i descendants at level 1. Therefore we have $|V_i| = O(\varepsilon^{-1} 2^i)$. Inserting this, we conclude that $O(\sum_{i=1}^{\lg U} |V_i| 4^{-i} \lg U) = O(\sum_{i=1}^{\lg U} \varepsilon^{-1} 2^i 4^{-i} \lg U) = O(\varepsilon^{-1} \lg U)$.

To summarize, we have improved the space to $O(\varepsilon^{-1} \lg U)$ (which can be proved to be optimal), the update time is worst case $O(\lg U)$ and the expected query time is $O(\varepsilon^{-1} \lg U)$.

References

[CM05] Graham Cormode and S. Muthukrishnan. An improved data stream summary: the count-min sketch and its applications. *J. Algorithms*, 55(1):58–75, 2005.

[LNNT16] Kasper Green Larsen, Jelani Nelson, Huy L. Nguyen, and Mikkel Thorup. Heavy hitters via cluster-preserving clustering. In *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS*, pages 61–70, 2016.