CSCE 689: Special Topics in Modern Algorithms for Data Science Fall 2023

Lecture 12 — September 18, 2023

Prof. Samson Zhou *Scribe: Chunkai Fu*

1 What is Misra-Gries algorithm and how does it work?

The Misra-Gries algorithm is a streaming algorithm used to approximate frequent items in a data stream. It's a memory-efficient algorithm designed to handle large datasets that cannot fit into memory or in situations where it is infeasible to store all the data on a disk. The algorithm works in the following way:

1. Start with an empty dictionary that will store candidate items and their counts.

2. For each data item from the data stream, do the following

a. If the item is already in the dictionary, increment its counter.

b. If the item is not in the dictionary and there is still space available for a new candidate, add the item to the dictionary with a counter of 1.

c. If the item is not in the dictionary and there is no space available for a new candidate, decrement the counters of all items in the dictionary and remove items with counters reaching zero.

3. After processing the entire data stream, we are left with a dictionary containing candidate frequent items and their corresponding counts.

The Misra-Gries algorithm does not guarantee exact counts for frequent items but provides an approximate count within a specified error bound. The error depends on the amount of available memory and the chosen threshold for removing items from the dictionary. The algorithm is particularly useful when we have a large data stream and limited memory to work with. It allows us to identify potential frequent items without storing the entire stream in memory.

It should be noted that if precise counts are required, and we have enough memory, other algorithms like the Apriori algorithm or the FP-growth algorithm may be more suitable for finding exact frequent items.

2 What is the CountSketch algorithm?

1. **Initialization:** Choose two hash functions $h : [U] \to [B]$ and $g : [U] \to \{-1, 1\}$ uniformly at random, where $[U]$ is the universe of items and $[B]$ is the range of buckets. Initialize a table C of size B with all entries set to 0.

2. **Processing Stream:** For each item i in the stream, update the table as:

$$C[h(i)] \leftarrow C[h(i)] + g(i) \times \text{count}(i)$$

where $\text{count}(i)$ is the occurrence count of item i in the current stream update.

3. **Estimate:** To estimate the frequency $\hat{f}_i$ of an item i, compute:

$$\hat{f}_i = g(i) \times C[h(i)]$$

The algorithm provides an approximation, and the accuracy can be improved by using multiple hash function pairs and averaging the results or by taking the median of the estimates. It is important to note that to retrieve the frequency, we need to multiply the result in the hashed bin by a sign.

3 What's the difference between CountMin and CountSketch?

Count-Min and Count-Sketch are both probabilistic data structures used for estimating frequencies of items in data streams or large datasets with limited memory. While they serve a similar purpose, they have different structures and properties. CountMin is primarily used for approximate frequency counting and is designed to provide accurate estimates of item frequencies in a data stream. CountSketch is a more versatile data structure. It can be used not only for approximate frequency counting but also for solving other problems like finding quantiles and approximating inner products of vectors.

Due to the difference in the data structures of these two algorithms, CountMin provides accurate frequency estimates, while CountSketch is less accurate in estimating frequencies. It is designed for scenarios where we need approximate answers and can tolerate a certain level of error. As we could expect, CountMin typically uses more memory than CountSketch because it maintains a count for each cell in the array, while count sketch uses a hash function to find the corresponding position for the count.

In summary, both algorithms are probabilistic data structures for approximate counting, they differ in their data structures, accuracy, memory usage, and use cases. CountMin is used when accurate frequency estimates are crucial, while CountSketch is used in situations where approximate answers are acceptable and memory efficiency is a priority.

4 What's the difference between Misra-Gries and CountSketch?

The results from the Misra-Gries algorithm is deterministic while the CountSketch algorithm can handle both insertion and deletion of items in the stream. Misra-Gries also gives accurate counts while CountSketch may give less accurate count, it gives the relatively top frequent items in a more efficient manner.

5 What are the relative sizes of the L_1 and L_2 norm?

The L_1 norm is always going to be greater or equal to the L_2 norm. We know that

$||x||_1 = |x_1| + |x_2| + \ldots + |x_n|$

$||x||_2 = \sqrt{x_1^2 + x_2^2 + x_3^2 + \ldots + x_n^2}$

Taking the square of the L_1 and L_2 norm, we have:

$||x||_1^2 = |x_1|^2 + |x_2|^2 + \ldots + |x_n|^2 + \sum |x_i| \, |x_j|$

$||x||_2^2 = x_1^2 + x_2^2 + x_3^2 + \ldots + x_n^2$

We know that $\sum |x_i| \, |x_j| \geq 0$

Therefore $||x||_1 \geq ||x||_2$

6 What is the root cause of CountSketch giving inaccurate results?

It was because of the occurrence of collision in the hash function. We think this inaccuracy is "destined" because in the algorithm, we are trying to map the count of items in the stream to a very small size hash table. So the chance of getting a collision is pretty high. We cannot use a perfect hash because that would require a large size of hash table, which goes against the initial mitice of using CountSketch, which is less memory. Why are we introducing the feature of positive and negative signs? That is because we are trying to cancel out the L_1 norm bounded error with the positive and negative signs so that it can be squeezed close to L_2 norm.

It is worth noting that the selection of whether adding a positive or negative sign needs to be independent for each hashed results. Otherwise, the count results will drift severely.