Asmt 5: Streaming Algorithms

Vai Suliafu, u0742607
Wednesday, February 26

1 Misra-Greis

A (40 points): Run the Misra-Gries Algorithm (see **L11.3.1**) with $(k-1) = 9$ counters on streams S1 and S2. Report the output of the counters at the end of the stream. In addition to each counter report the estimated ratio of each label using the estimated count/m.

In each stream, use just the counters to report which characters *might* occur at least 20% of the time, and which must occur at least 20% of the time.

The output of the of the Misra-Gries algorithm on s1 is shown below:

Misra-Gries on s1 with (k -1) = 9

Counts:
[736664, 436662, 1, 1, 1, 1, 197649, 1, 0]

Percentages:
[25%, 15%, 0%, 0%, 0%, 0%, 7%, 0%, 0%]

Labels:
['a', 'b', 'd', 'p', 't', 'n', 'c', 'v', 'k']

Elements in s1 that must be greater than 20% frequency:
['a']

Elements in s1 that might be greater than 20% frequency:
['b']

Similarly, the results for the Misra-Greis algorithm on S2 is shown below:

Misra-Gries on s2 with (k -1) = 9

Counts:
[1, 685133, 1885833, 1, 1, 1, 1, 286358, 1]

Percentages:
[0%, 17%, 47%, 0%, 0%, 0%, 0%, 7%, 0%]

labels:
['h', 'b', 'a', 'f', 'j', 'i', 'o', 'c', 'v']

Elements in s2 that must be greater than 20% frequency:
['a']

Elements in s2 that might be greater than 20% frequency:
['b']

B (40 points): Build a Count-Min Sketch (see **L12.1.1**) with $k = 10$ counters using $t = 5$ hash functions. Run it on streams S1 and S2.

For both streams, report the estimated counts for characters `a`, `b`, and `c`. In addition to each counter report the estimated ratio of each of these labels using the estimated count/m. Just from the output of the sketch, with probably $1 - \delta = 31/32$ (that is assuming the randomness in the algorithm does not do something bad), which objects *might* occur at least 20% of the time, and which objects *must* occur at least 20% of the time.

The output of the Count-Min Sketch Algorithm on s1 is shown below:

Count-Min Sketch on s1 with k=10, t=5:

Character: a
Approximated Count in S1: 899566
Approximated Frequency in S1: 30%

Character: b
Approximated Count in S1: 719573

Approximated Frequency in S1: 24%

Character: c
Approximated Count in S1: 360551
Approximated Frequency in S1: 12%

Elements in s1 that must be greater than 20% frequency
['a']

Elements in s1 that might be greater than 20% frequency
['b']

Similarly, output of the Count-Min Sketch Algorithm on s2 is shown below:

Count-Min Sketch on s2 with k=10, t=5:

Character: a
Approximated Count in S2: 2040066
Approximated Frequency in S2: 51%

Character: b
Approximated Count in S2: 839110
Approximated Frequency in S2: 0.21%

Character: c
Approximated Count in S2: 400625
Approximated Frequency in S2: 10%

Elements in s2 that must be greater than 20% frequency
['a']

Elements in s2 that might be greater than 20% frequency
[]

C (10 points): How would your implementation of these algorithms need to change (to answer the same questions) if each object of the stream was a "word" seen on Twitter, and the stream contained all tweets concatenated together?

If instead of characters, each object of the stream was a 'word' separated by white-space, we could need to update our algorithm to parse 'word' tokens instead of simply iterating through each character of the word.

D (10 points): Describe one advantage of the Count-Min Sketch over the Misra-Gries Algorithm.

One clear advantage of Count-Min Sketch over the Misra-Gries algorithm is the re-usability of the count Matrix data structure that is built in the Count-Min Sketch Algorithm. More specifically, given this Count Matrix data structure, one can easily retrieve the Count-Min Sketch approximations for any element contained in the stream in $O(t)$ time, where t is the number of different hash functions. This feature is obviously preferable when contrasted with Misa-Gries output, which will only give you approximations for the k-1 most frequently observed elements.