1 Report

a) Universal Hashing Function Family

For the assignment purposes, I utilized universal hashing family functions. Theory analysis provides that i-th hashing function is pairwise independent with (i+1)-th function, where $0 < i \leq N$ and N is the number of hash functions of countMin sketch algorithm. The parameters of this hashing family are the following:

- $h_i = (a_i \cdot x + b_i)\%p$
- a_i, b_i are random numbers where $1 \leq a_i \leq p$ and $0 \leq b_i \leq p$
- p is the first prime number greater than the max value of input stream.

b) Question 1

The countMin sketch algorithm was implemented with Python and it is provided in the zip folder included in my submission. The algorithm is written on *utils.py* scirpt.

c) Question 2

I run my experiments with an artificial dataset which is also provided under the data folder with the real ones. It has 25000 elements. In order to check that my implementation works correctly I run my code with 1000 buckets and 2 hash functions. The theoretical bound of the probability of overestimating with those hyperparameters is 0.25 (since I am using universal hash functions) and the results are the following:

```
 ~/Documents/CS543   python q2.py
Total number of elements in the stream: 25000
Theoretical bound of CountMinSketch with 1000 buckets and 2 hash functions: 0.25
Probability of overestimating elements: 0.0026200873362445414
```

Figure 1: Question 2 script results

d) Question 3

To answer this question, I provide both the probability of overestimating the elements and a plot that shows how much overestimate the elements of my real datasets and how close they are to the bound of $\frac{2 \cdot S}{K}$, where S is the number of total elements and K is the number of buckets. It is easily noticeable that the algorithm works well and the probability of overestimating elements is less than the theoretical bound of 0.25 (for 2 hash functions). To visualize that better, Figure 3 shows that the deviation of the overestimating elements drops as we approach the bound of $\frac{2 \cdot S}{K}$. In particular, more than the 75% of the distinct elements have a deviation value less than 20, when the bound is $\frac{2 \cdot S}{K} = 55.612$.

e) Question 4

To answer this question, I run two types of experiments over a real dataset named LastFM Asia Network (there is an option for artificial one as well). The first experiment is presented at Figure 4, in which I keep

```
~/Documents/CS543   python q3.py
Total number of elements in the stream: 27806
Theoretical bound of CountMinSketch with 1000 buckets and 2 hash functions: 0.25
Probability of overestimating elements: 0.004225352112676056
```

Figure 2: Question 3 script results

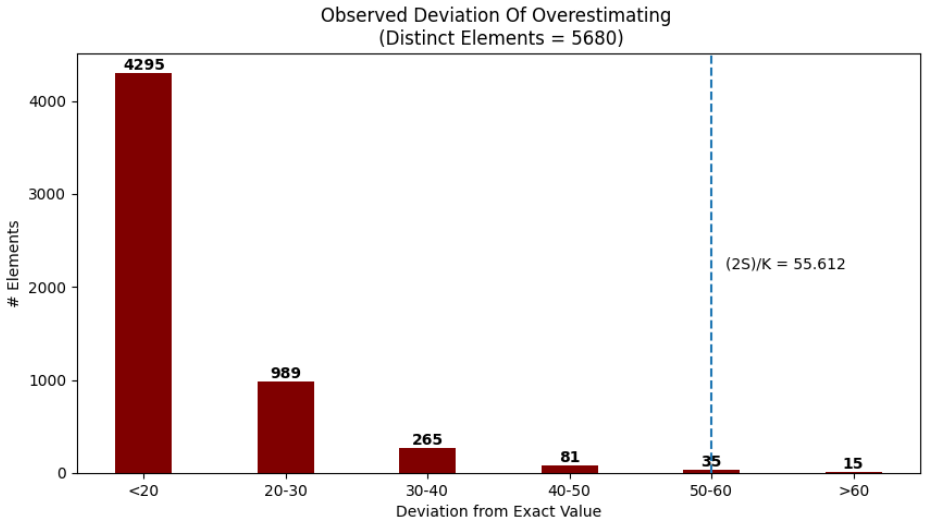

Figure 3: Question 3 Observed Deviation of Overestimating

stable the number of hash functions equal to 4, I vary the number of buckets from 1 to 200 and I count the mean deviation. As the Figure 4 presents, the mean deviation drops as we increase the number of buckets. The second experiments is presented at Figure 5, in which I keep stable the number of buckets equal to 1000, I vary the number of hash functions and I count the overestimating probability. As the Figure 5 presents, the probability drops as we increase the number of hash functions.

The experiments show that the theory match the theory predictions and it is significantly better. In particular, both the experiments from question 3 and 4 show that the probability bound that Markov's inequality provides us is not tight since the experiments we run have an overestimation probability way less than the theoretical analysis.

f) Question 5

To answer the question, I utilized 3 different real datasets (Facebook, LastFM Asia, Wikipedia). I test those 3 to check if their distribution approves the theoretical bound of probability that Markov's inequality provides as. To verify my experiments, I rerun the countMin sketch algorithm 5 times and I took the average of it to compare my probability of overestimating with the thoretical bound. The theoretical bound for a countMin sketch algorithm with $N = 2$ hash functions is $\frac{1}{2^N} = \frac{1}{4}$. As the Figures 6,7 and 8 present, the mean overestimating probability of the 3 datasets are less than the theoretical bound, proving once again the theoretical analysis.

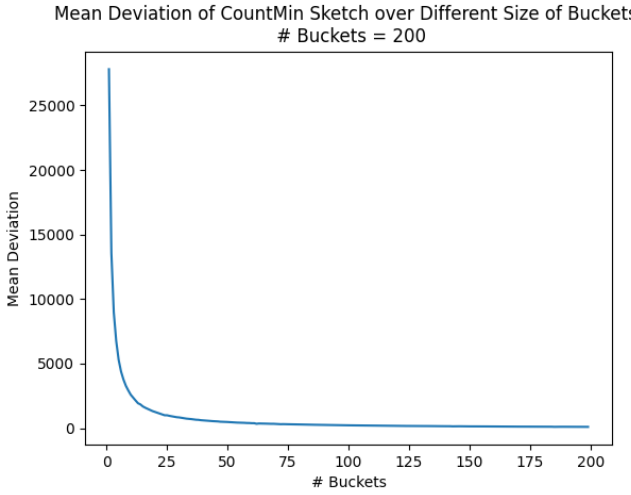

Figure 4: Question 4: use N=4 hash functions and vary the number of buckets (K) from 1 to 200

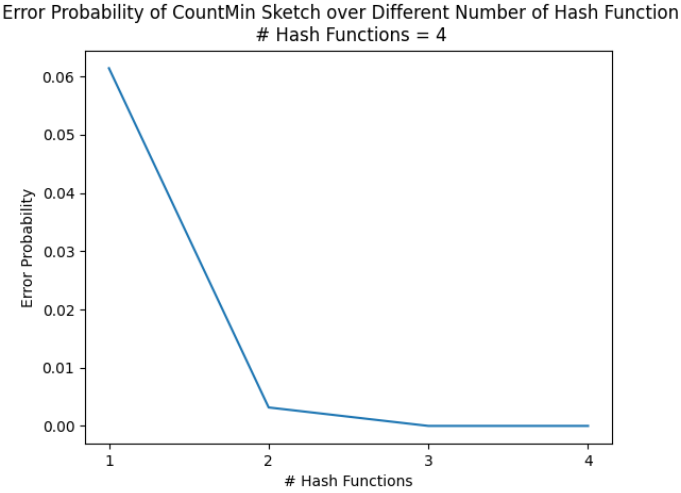

Figure 5: Question 4: use K=1000 buckets and vary the number of hash functions (N) from 1 to 4

```
 ~/Documents/CS543   python q5.py -d f
Facebook dataset was chosen!
Total number of elements in the stream: 88234
Theoretical bound of CountMinSketch with 1000 buckets and 2 hash functions: 0.25
Mean Probability of overestimating elements after 5 runs of countMin sketch algorithm: 0.0024770869457517958
```

Figure 6: Question 5 script results for Facebook script

```
 ~/Documents/CS543   python q5.py -d l
LastFM Asia dataset was chosen!
Total number of elements in the stream: 27806
Theoretical bound of CountMinSketch with 1000 buckets and 2 hash functions: 0.25
Mean Probability of overestimating elements after 5 runs of countMin sketch algorithm: 0.002112676056338028
```

Figure 7: Question 5 script results for LastFM Asia script

```
~/Documents/CS543   python q5.py -d w
Wiki dataset was chosen!
Total number of elements in the stream: 103688
Theoretical bound of CountMinSketch with 1000 buckets and 2 hash functions: 0.25
Mean Probability of overestimating elements after 5 runs of countMin sketch algorithm: 0.009659806803863923
```

Figure 8: Question 5 script results for Wikipedia script

g) Question 6

The exercise took me 3-4 days to implement it. It was not that hard but it was not clear how should I present the results of each sub question, therefore I had to ask the professor and the teaching assistant.

2 Data

a) Datasets

All datasets (artificial or real) are included in my zip file and they are ready to be used. No further action should be made. The datasets used for my experiments are all provided from SNAP: Network datasets site. The link is here. In particular, I utilized 3 graph related datasets taking as "stream" elements the destination nodes of the graphs. In particular:

- **Social circles: Facebook**: This dataset consists of 'circles' (or 'friends lists') from Facebook. Facebook data was collected from survey participants using this Facebook app. Number of edges are 88234. More information about the dataset can be found here.
- **LastFM Asia Social Network**: A social network of LastFM users which was collected from the public API in March 2020. Nodes are LastFM users from Asian countries and edges are mutual follower relationships between them. Number of edges are 27,806. More information about the dataset can be found here.
- **Wikipedia vote network**: Wikipedia is a free encyclopedia written collaboratively by volunteers around the world. A small part of Wikipedia contributors are administrators, who are users with access to additional technical features that aid in maintenance. In order for a user to become an administrator a Request for adminship (RfA) is issued and the Wikipedia community via a public discussion or a vote decides who to promote to adminship. Using the latest complete dump of Wikipedia page edit history (from January 3 2008) we extracted all administrator elections and vote history data. This gave us 2,794 elections with 103,663 total votes and 7,066 users participating in the elections (either casting a vote or being voted on). The number of votes are the edges. More information about the dataset can be found here.

b) Code Reproducibility

The code was written and tested on Ubuntu 20.04 operating system, with Python 3.9 installed. The only python packages required to be installed to run the code are the following: *numpy, pandas, matplotlib, sympy*. To install them under pip you need to follow the command:

- pip install pandas numpy matplotlib sympy

After unzipping the .zip folder and accessing it, the following lines provide you a description of how to run the subquestions of the exercise.

For question 2, please run the *q2* script as follows: **python q2.py**

For question 3, please run the *q3* script as follows: **python q3.py**

For question 4, please run the *q4* script as follows: **python q4.py -d r**
(the parameter d denotes the choice of "r" real dataset. You can choose "a" for articifial dataset as well.)

For question 5, please run the *q5* script as follows: **python q5.py -d f**
(the parameter d denotes the choice of "f" Facebook real dataset. You can choose "l" for LastFM Asia dataset or "w" for Wikipedia dataset as well.)

All the plots will be generated under the plot folder.