Assignment 4

1 Data Streams

In this problem, we study an approach to approximate the frequency of occurrences of different items in a data stream. Assume $S =< a_1, a_2, ..., a_t >$ is a data stream of items from the set $\{1, 2, ..., n\}$. Assume for any $1 \leq i \leq n$, $F[i]$ is the number of times item i has appeared in S. We would like to have good approximations of the values $F[i](1 \leq i \leq n)$ at all times.

A simple way to do this is to just keep the counts for each item $1 \leq i \leq n$ separately. However, this will require $O(n)$ space, and in many applications (e.g., think online advertising and counts of user's clicks on ads) this can be prohibitively large. We see in this problem that it is possible to approximate these counts using a much smaller amount of space. To do so, we consider the algorithm explained below.

Algorithm. The algorithm has two parameters $\delta, \epsilon > 0$ which are small positive values. It picks $\lceil \log \frac{1}{\delta} \rceil$ independent hash functions where log denotes natural logarithm. For each hash function h_j $(1 \leq j \leq \lceil \log \frac{1}{\delta} \rceil)$, it hashes an integer from $\{1, 2, ..., n\}$ to $\lceil \frac{e}{\epsilon} \rceil$ buckets (as defined below), i.e.,

$$h_j : 1, 2, ..., n \rightarrow \{1, 2, ..., \lceil \frac{e}{\epsilon} \rceil\}$$

Also, the algorithm maintains a count $c_{j,x}$ for all $1 \leq j \leq \lceil \log \frac{1}{\delta} \rceil$ and $1 \leq x \leq \lceil \frac{e}{\epsilon} \rceil$. In the beginning of the stream, all these counts are initialized to 0. Then, upon arrival of each stream element a_k $(1 \leq k \leq t)$, each of the counts $c_{j,h_j(a_k)}$ $(1 \leq j \leq \lceil \log \frac{1}{\delta} \rceil)$ is incremented by 1 (i.e., $\lceil \log \frac{1}{\delta} \rceil$ counts are updated for each arrival element).

For any item $1 \leq i \leq n$, we define $\tilde{F}[i] = min_j\{c_{j,h_j(i)}\}$. Then, $\tilde{F}[i]$ provides a good approximation to $\tilde{F}[i]$.

Memory cost. Note that this algorithm only uses $O(\frac{e}{\epsilon} \log \frac{1}{\delta})$ space (i.e., the number of all counts $c_{j,x}$).

What to do. Implement the above algorithm and run it on the dataset below with parameters $\delta = e^{-5}$, $\epsilon = e * 10^{-4}$. (Note: with this choice of δ you will be using exactly 5 hash functions. The hash function is defined below, and the 5 pairs (a, b) that you will need for the hash functions are in **hash_params.txt**). The algorithm will output count approximations $\tilde{F}[i]$ for all i's. Then, for each distinct item i in the dataset, compute the relative error $err[i] = \frac{\tilde{F}[i]-F[i]}{F[i]}$ and plot these values as a function of the exact word frequency $\frac{F[i]}{t}$.
The plot should use a logarithm scale both for the x and the y axes, and there should be ticks to allow reading the powers of 10 (e.g. 10^{-1}, 10^0, 10^1 etc.). The plot should have the x and y axes. The exact frequencies $F[i]$ should be read from the counts file *counts.txt*. For different items with the same frequency, draw them as different points with the same x value in the plot. Note that items of low frequency can have a very large relative error. That is not a bug in your implementation.

You can use the python code implemented in the IPython Notebook to draw the plot.

Assignment 4

Hash functions. You should use the following hash function (see following pseudo-code), with $p = 123457$, a and b values provided in file **hash_params.txt** and $n\_buckets = \lceil \frac{e}{\epsilon} \rceil = 10^4$. In the provided file **hash_params.txt**, each line gives you a, b values to create one hash function.

```
# Returns hash(x) for hash function given by parameters a, b, p and n_buckets
def hash_fun(a, b, p, n_buckets, x)
{
y = x [modulo] p
hash_val = (a*y + b) [modulo] p
return hash_val [modulo] n_buckets
}
```

Dataset. The dataset contains following files:

- **data_stream.txt**: Each line of this file is a number, corresponding to the ID of an item in the stream.
- **counts.txt**: Each line is a pair of numbers separated by a tab. The first number is an ID of an item and the second number is its associated exact frequency count in the stream.
- **hash_params.txt**: Each line is a pair of numbers separated by a tab, corresponding to parameters a and b which you should use to define your hash functions.

What to submit:

- Log-log plot of the relative error as a function of the frequency.
- The source code file.
- Print the top 10 most frequent IDs with highest exact counts and the corresponding approximate counts. Each line is a triplet, i.e., (ID, approximate count, exact count).
- Print the top 10 least frequent IDs with lowest exact counts and the corresponding approximate counts. Each line is a triplet, i.e., (ID, approximate count, exact count).