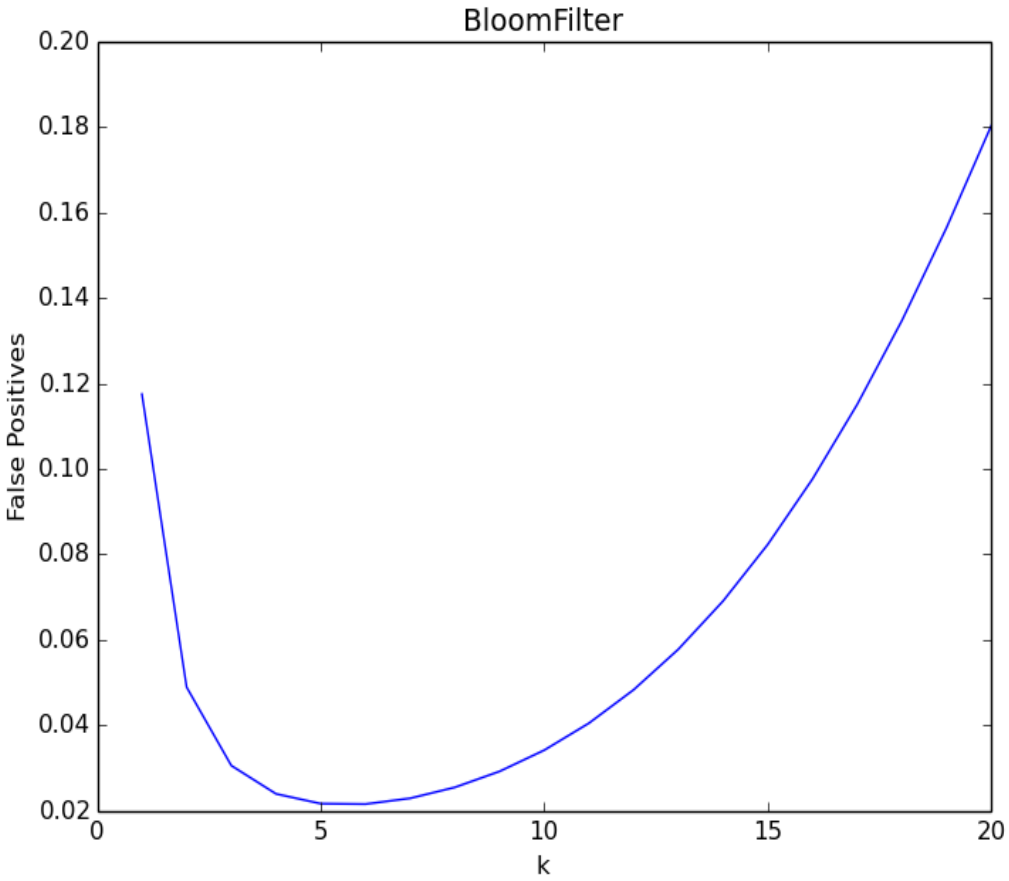
PA:5

Abhishek Yadav,CS12B032

April 19,2016

Question1



Plot of the **False-Positive Rate** of **Bloom-Filter** against k.

(a) It is shown that for large values of n, **False-positive** probability is: $(1 - e^{-km/n})^k$. substituting m/n = 1/8, let's say

$$F(k) = (1 - e^{-k/8})^k$$

,
then

$$\frac{dF}{dk} = (1 - e^{-k/8})^k \left(\frac{k}{8(e^{k/8} - 1)} + \ln(1 - e^{-k/8}) \right)$$

solving this we find that False Positive rate attains minimum at k= 8*ln(2) = 5.54 which can be verified from the plot. $\forall k \in (1, 5.54)$ the derivative is negative and keeps increasing with k (i.e. decreases in magnitude) and changes sign at 5.54(approx). But growth rate slows down after this point.

(b) **Conclusion:** False positive rate of the Bloom-Filter depends on variables m,n and k. From the plot we can see that it is minimum for k={5,6}. False positive rate is lowest when number of hash-Functions(k) = (n/m)*ln(2). So, with increasing space, keeping the stream constant will require larger number of hash-Functions to minimize the False positive rate and less otherwise. Keeping the above ratio constant, as in this case we observe that the False positive rate is low for small values of k which can be attributed to the fact that small number of Hash-Function will set bits less rapidly as compared to higher number of hash functions hence decreased FPR. Similarly, for higher number of hash functions, filter bits are set rapidly and hence increase in FPR due to wrongly identifying the later stream elements as seen. Next, for value of $k \in (3, 7)$ filter bits will be set less rapidly and since k-value is reasonably high this results in decreased FPR for the number of hash functions in this range.

Question2:

(a) **Results:** Sapce-Saving attains 100% accuracy for **y.txt** and almost 0% for x.txt. Results are summarised as following:
y.txt: for all values of **k** acccuracy is 100%.
x.txt: for k=100 accuracy is 0.1%, for 500 accuracy is 0.23%, for 1000 accuracy is 0.37% and for 10000 accuracy is 0.56%.

(b) I did some random search over few top elements in both the datasets and found out that: In dataset y, the elements with actual high frequencies were the top elements given by **space-saving** algorithm. In dataset y.txt elements which were actually frequent occurred regularly Plotting the datasets would reveal that one of the data streams follows a power law distribution while the other follows a uniform distribution. In the space-saving algorithm, inaccuracies creep in when an item is not tracked throughout its existence. The chances of that happening is much higher

in a uniform distribution when compared to that of a power-law distribution. This is due to the factor that a highly frequent item (relative to other items in the stream) would almost always be among the tracked items. This happens in a power law distribution due the distribution being skewed towards a few items.

(c) When the entire stream cannot be stored, we are unable to compute its distribution. Without knowing the distribution, it is hard to analyze the accuracy of the space-saving algorithm. In this situation, we could perform reservoir sampling to approximate the distribution of the two streams. Based on the distributions, we can reason on the accuracy using the same logic as above.