Bloom Filter Project Report

Name: Chris Church

Task 1 Write-up: The goal of task 1 was observe the distribution of hashed values using two hashing functions. For Hash Function 1, when the values were generated, they were made into a histogram using the bucket size 1,000. The minimum and maximum number of occurrences for the buckets were recorded. A scatter plot was also recorded on one of the runs. The function was run 10 times with the random data, generating an average minimum amount of occurrences of 5, an average maximum of occurrences of 39. For the second tests, the numbers 2 through 20,000 in ascending order were given to the hashing function and histograms and scatter plots were generated the same way, except for a bucket size of 99 being used and a smaller hash table for the smaller set (101). The average max occurrences for the ordered data was 124 and the average minimum occurrences was 79. Again the data showed random distribution

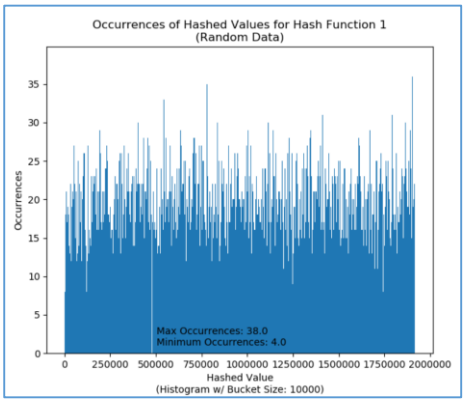
The second function, Hash Function 2, was also examined in the same way as Hash Function 1. For the random data, the distribution of hashed values appeared random on the scatter plot and histogram with an average max occurrence of 38 and an average minimum occurrence of 6. The ordered data showed clear signs of non-random behavior. The histograms had a sinusoidal shape to their distribution and the scatter plot also had a repeating pattern. The average max occurrences was 103 and the average minimum occurrences was 99.

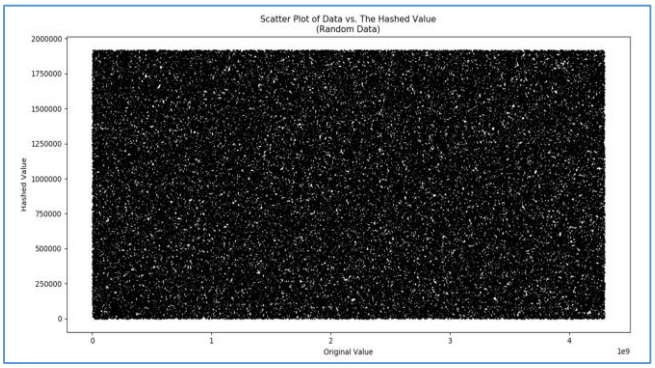
By looking at the difference in the average maximums and average minimums, it can be shown which function is more likely to evenly distribute the hashed values by being less likely to have starved or overloaded values. The second function improved on the difference between the max and min occurrences of the ordered data with a difference of averages of 4 compared to the difference of 45 when using Hash Function 1. But the two appeared to perform the same for the random data with their average differences being 1 apart.

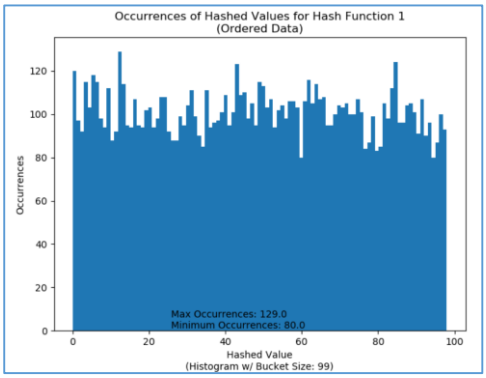
Task 2 Write-up: For the Bloom Filter implementation, an array called *hashtable* of size *n* is created in the bloom filter object to hold the zeroes and ones for the filter. *testBF* calls *add* on the *BloomFilter* object and gives it a number to add, the *BloomFilter* calls *getHashList* from the hash function object it created earlier. This will return a list of *k* hashed values based on either Hash Function 1 or 2. The values in the returned list are then used as indices for the *hashtable* array and the value at each index in is set to 1. After adding values, *testBF* then calls *contains* in the *BloomFilter* object to test if a number that it passes to the function is in the Bloom filter set. To do this *contains* calls the hashing function on the value it was passed just as *add* does, and receives a list of values back. The function then checks the values in *hashtable* where the received list is used as the indices. If at any point during the value checks, the value checked in *hashtable* equals zero, the function returns false. If the function makes it the whole way through finding only ones, it returns true.

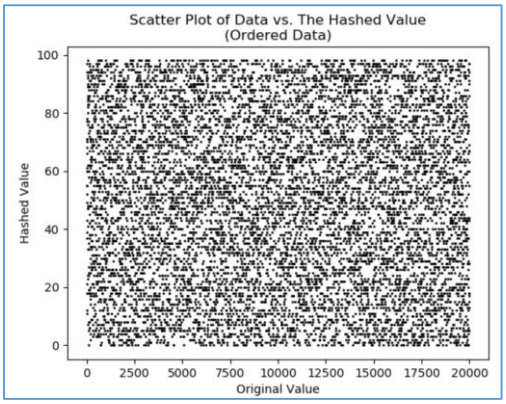
Task 3 Write-up: The four plots are shown on page three with their optimal k values labeled. The theoretical values were calculated at each k with the equation given in the project description. The observed results generated with the program closely matched the theoretical plots more closely for Hash Type 1 and for c = 10. For the c = 10 plots, the same shape was generated between the observed and theoretical plots, and both found the same optimal *k* value (7) as their theoretical counterparts. The c = 15 on the other hand had shapes that were a little more inconsistent with the theoretical plots and their optimal *k* values were one higher than their theoretical plots. Granted, the values observed were pretty close. Also, the larger hash tables when *c* = 15 decreased the false positives by an order of magnitude due decreasing the probability of collisions. Combining these results with experiments using other data sets may have helped make the c = 15 plots closer. More runs of the data set used (the test input provided) would also have benefitted those plots.

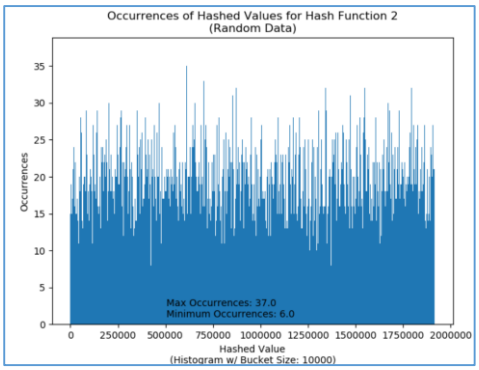
Bloom Filter Project Report
Name: Chris Church
Task 1 Plots:

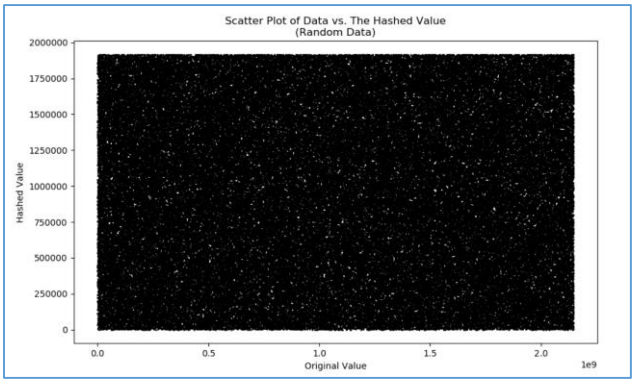


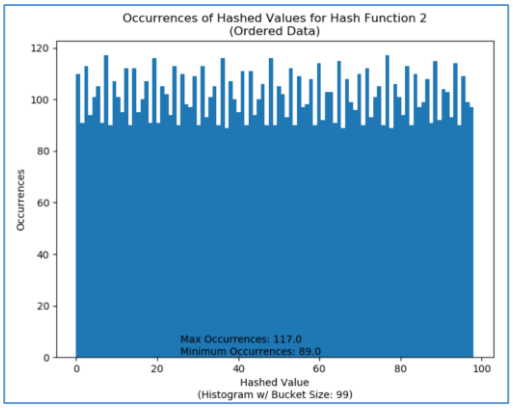


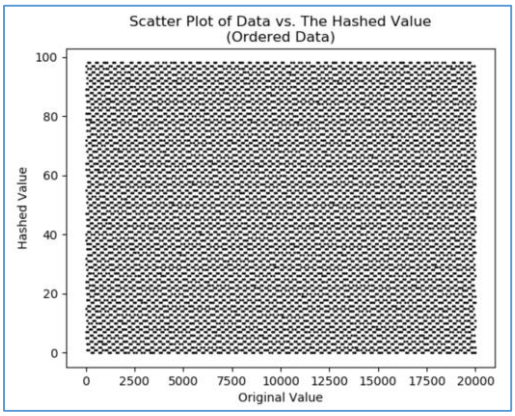










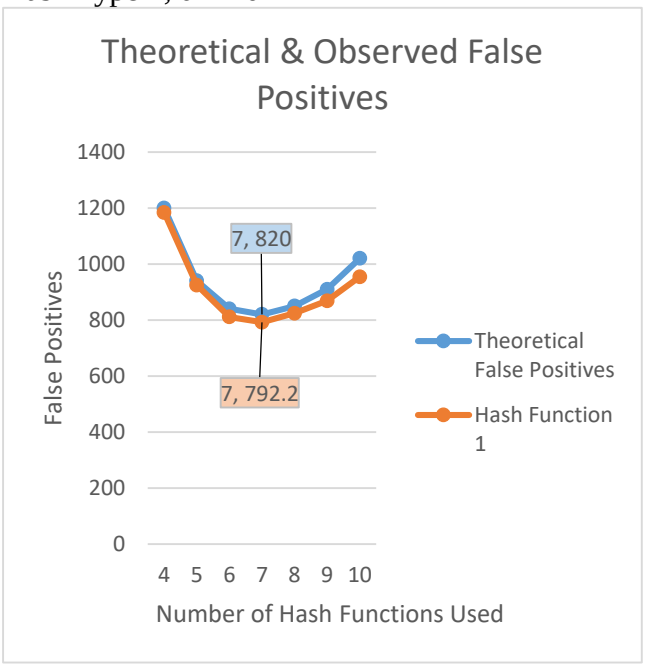


Bloom Filter Project Report

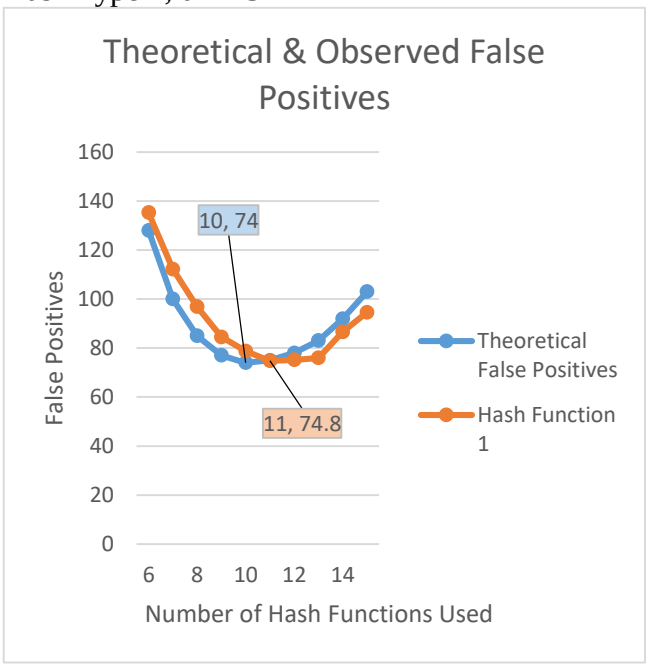
Name: Chris Church

Task 3 Plots:

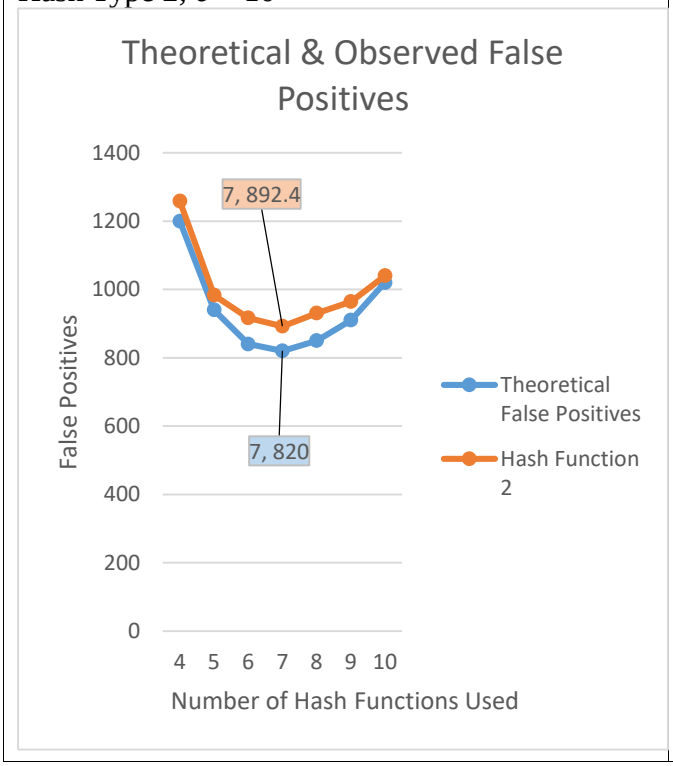
Hash Type 1; $c = 10$



Hash Type 1; $c = 15$



Hash Type 2; $c = 10$



Hash Type 2; $c = 15$