Nayeel Imtiaz
Professor Long
CSE 13s
5 December 2021

Write Up for Project 7 - “The Great Firewall of Santa Cruz”

Introduction:

The purpose of this lab was to utilize different data structures ranging from bloom filters, hash tables, and binary search trees, ultimately creating a program that can parse through user input and filter out bad words. The program can change the size of the bloom filter (default size = 2^20) and the hash table (default size = 2^16) from the command line. Changing these two values can drastically change values of other quantities such as the average tree branches traversed and the number of lookups. But on the other hand, certain quantities do not change even after changing the bloom filter size or hash table size.

Discussion:

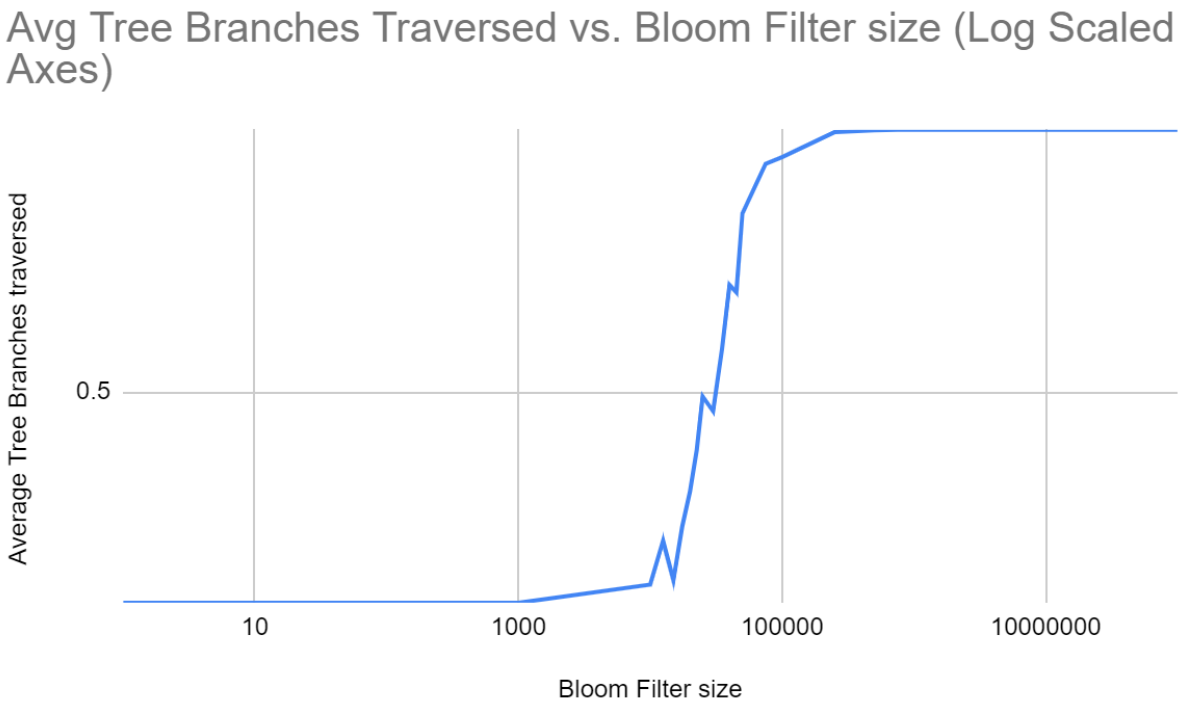
The heights of binary search trees solely depend on the size of the hash table. The smaller the size of the hash table, the greater the average Binary Search Tree height is according to the graph comparing average BST height and hash table sizes. This makes sense since a bigger hash table means the Binary Search Tree has a smaller size, and a BST with a smaller size means the height is lower. However, changing the bloom filter size seems to have no effect on the binary search tree heights. No matter how low or high the bloom filter size is, the average BST height remains constant. This can be observed in the graph below comparing average BST height and bloom filter sizes.

Increasing the bloom filter size decreases the number of lookups performed. The number of lookups changes at the slowest rate around the lowest and highest bloom filter sizes, but on the other hand the number of lookups changes at the fastest rate in between the lowest and highest bloom filter sizes. Changing the hash table size has no effect on the number of lookups in any way though. Also, the number of branches traversed goes up if the bloom filter size is increased, and the number of branches traversed goes down if the hash table size is increased. For both bloom filter size and hash tables size, the magnitude of the rate of change of the branches traversed is at the greatest at average values of bloom filter size and hash table size. On the flip side, the magnitude of the rate of change of the branches traversed is at its lowest when bloom filter size and hash table size are either at a minimum or at a maximum.

Results:

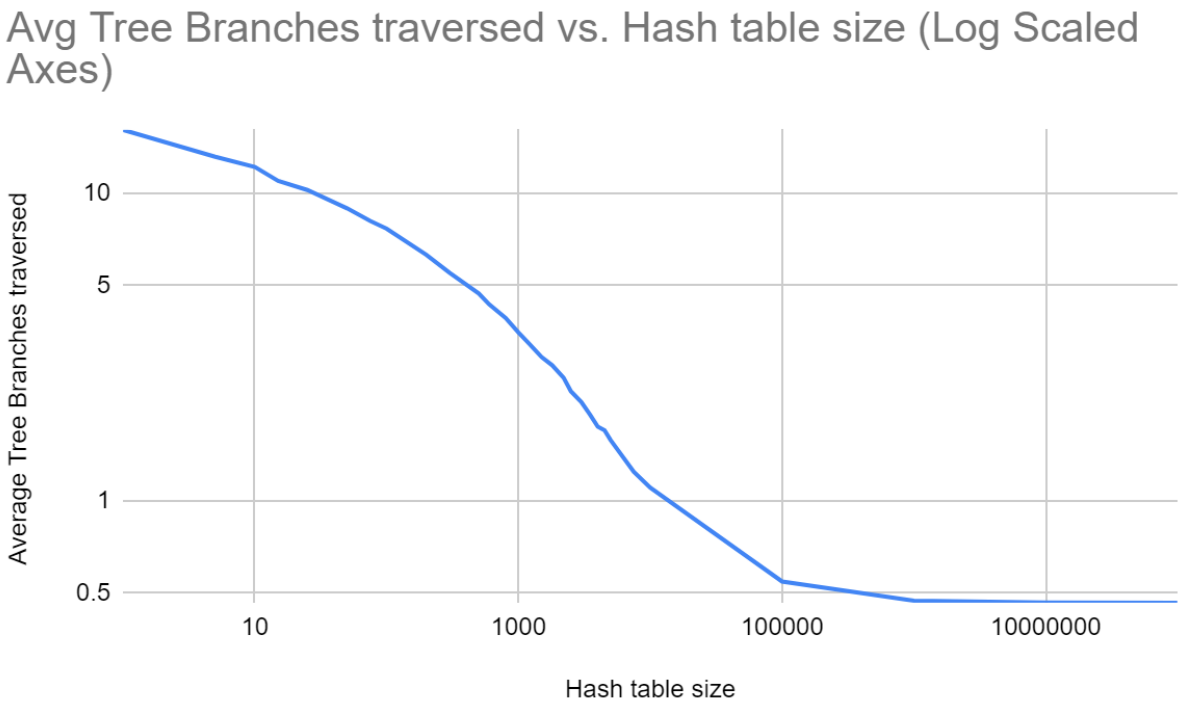
All of these graphs were produced with the Bee Movie script being the input file. Also, all of the graphs below had their axes log scaled in order to show changes in values more easily.

Average Tree Branches Traversed vs. Bloom Filter size (Log Scaled axes)



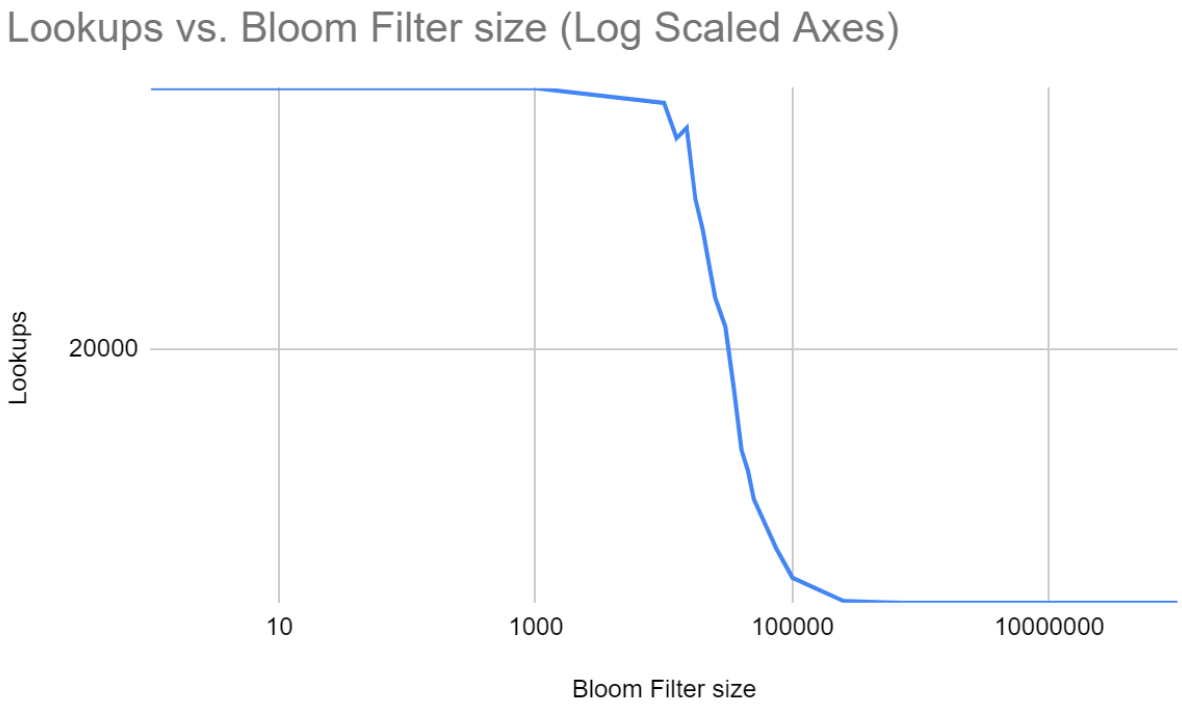
As it can be seen from the graph, the number of average tree branches traversed increases as the bloom filter size also increases. The average number of tree branches traversed increases rapidly somewhere in the middle but then slows down near the top and reaches a maximum. There are some random spikes where average tree branches traversed decreases, but that can most likely be attributed to the randomness of bloom filters. Also, it is important to take note of the resemblance to a logistic equation because of how the number of branch traversals increases rapidly around the middle.

Average Tree Branches Traversed vs. Hash Table size (Log Scaled axes)



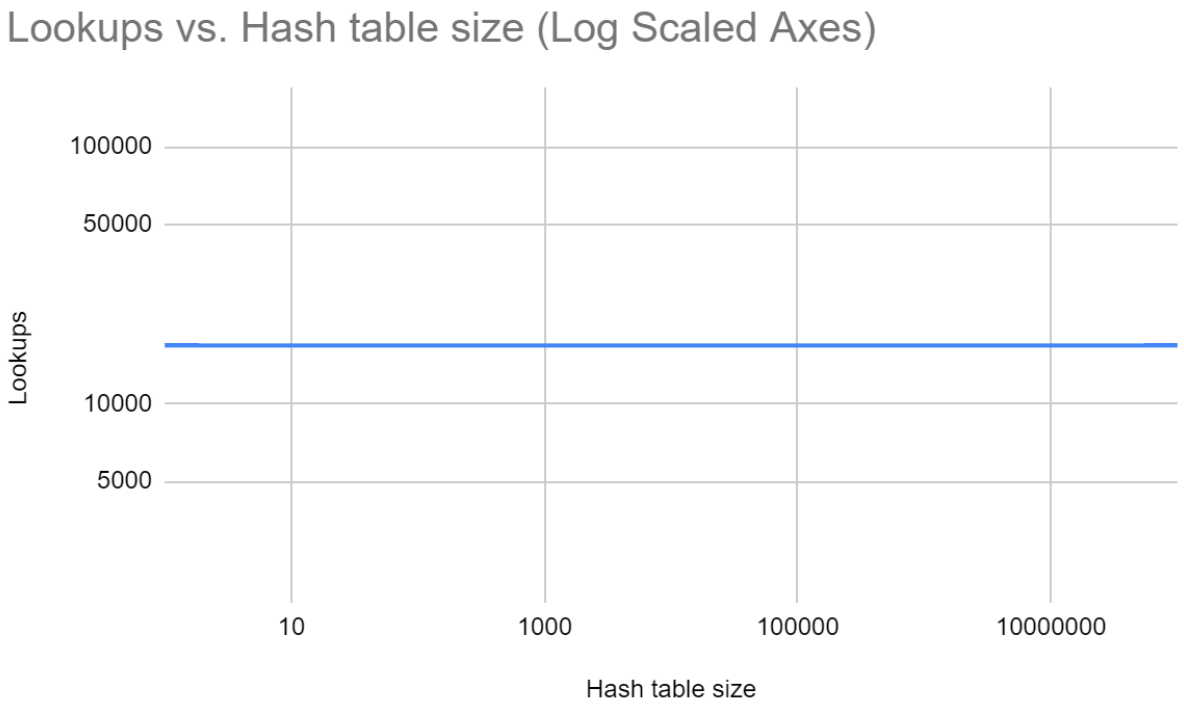
In this graph, the number of average tree branches traversed decreases as the hash table size increases. The average number of tree branches traversed keeps decreasing until it gets to a certain hash table size where the number of tree branch traversals stops changing. This is the opposite of the previous graph where the number of tree branch traversals increases as the bloom filter size increases. Also, this graph does not have random spikes like the previous graph even though hash tables are a little random to some extent too. So to obtain a faster performance time for the banhammer program, one can use larger hash tables, but the trade off would be more storage is used for larger hash table sizes.

Lookups vs. Bloom Filter size (Log Scaled axes)



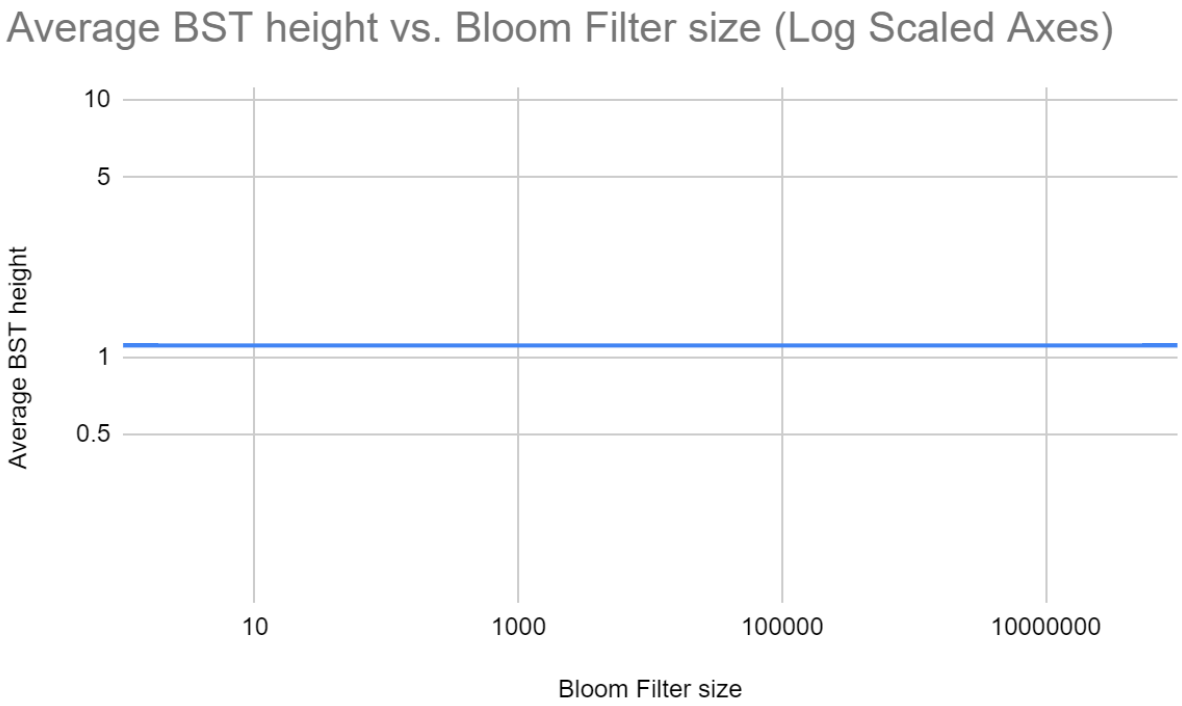
As it can be seen from the graph, the number of lookups decreases as the bloom filter size increases. The number of lookups rapidly decreases in the middle, but then it slows down until it reaches a certain minimum number of lookups. Thus, using a larger bloom filter size can drastically improve the efficiency of the banhammer program, but the trade off once again is the amount of memory needed for the bloom filter. There is a random spike in increase in the middle of the graph, but that can again be attributed to the random nature of bloom filters.

Lookups vs. Hash Filter size (Log Scaled axes)



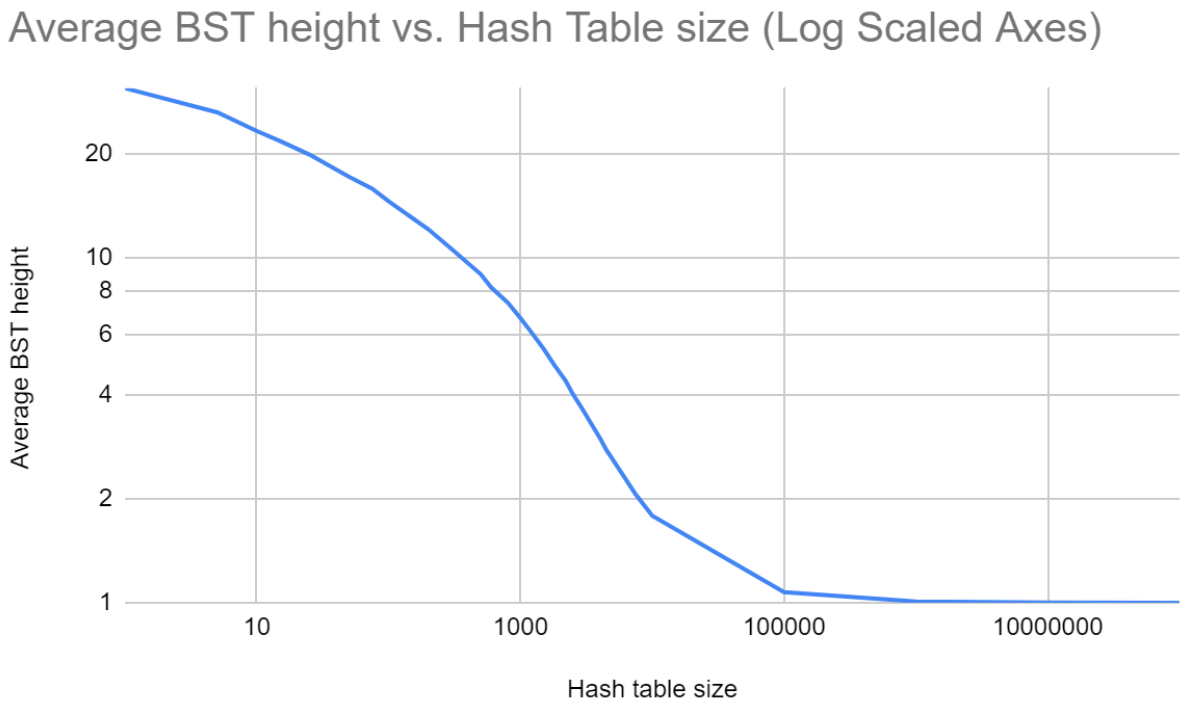
As it can be seen, changing the hash table size has no effect on the total number of lookups. No matter how low or high the hash table size is, the total number of lookups will always remain the same value. This is the opposite of what happened when we changed the bloom filter size as the number of lookups also changed depending on the bloom filter value. So there is no point of changing the size of the hash table to decrease the number of lookups.

Average Binary Search Tree Height vs. Bloom Filter Size (Log Scaled Axes)



Similar to the last graph, changing the bloom filter size has no effect on the average binary search tree height in any way. No matter how low or high the bloom filter size is, the average binary search tree height will always remain the same value. So there is no purpose of changing the bloom filter size to try and make the banhammer program perform faster. However, the same is not true when changing hash table sizes which can be seen in the next graph.

Average Binary Search Tree Height vs. Hash Table Size (Log Scaled Axes)



The average binary search tree height decreases as the hash table size increases. This makes sense as the average number of tree branches traversed also decreases as the hash table size increases. Once again, the average binary search tree height keeps decreasing until it reaches a minimum at a certain hash table size. This is different from the previous graph because changing the bloom filter size has no effect on the average binary search tree height in any way. This graph also shows that it is worth increasing the hash table size to make the banhammer program execute quicker, but the trade is the amount of memory needed for a larger hash table.

Conclusion:

In conclusion, changing the bloom filter size and hash table size has many effects on the banhammer program. Some quantities increase if the bloom filter or hash table size increases while some quantities decrease if the bloom filter or hash table size increases. Then there are some quantities that do not change even if the bloom filter or hash table size is modified. This was a very interesting assignment, and there was so much to learn from it.