WRITEUP: Assignment 7

Nelly Sin

December 2021

1 Introduction

This write up is based on the the banhammer.c program for assignment 7 - The Great Firewall of Santa Cruz: Bloom Filters, Binary Trees, and Hash Tables. The banhammer program filters out certain words a user can use by reading in a given file this is from badspeak.txt and newspeak.txt. In general, we want to find the words and notify the user if they are doing a good job of not using the words or warn them if they are using such words.

In addition, in this writeup I have produced six graphs from the statistics from the banhammer program. These graphs are based on the bloom filter size, hash table size, the number of lookups, average branches, average binary search tree size, and average binary search tree height.

When we compare the graphs, we must analyze and observe how the hash table and bloom filter graphs vary. To be specific:

1. How do the heights of the binary search trees change?
2. What are some factors that can change the height of the binary search tree?
3. How does changing the bloom filter size affect the number of lookups performed in the hash table?

2 Abstraction

The height of the hash table for bloom filter should not change; however, the hash table is a factor because the hash table is the index array of the binary search tree. But generally, the binary search tree heights change based on the file we are reading in from.

One of the factors for changing the height of the binary search tree are the hash table size, this should be because the hash table is the index for the binary search tree. The other I can think of is how many words that are in the file. Because the binary search tree rely on how many words there are in order for finding words to happen.

The bloom filter and the hash table are connected. And the hash table is connected with the binary search tree. We check if the word in the bloom filter exists then the total number of lookups will also decrease, since we are relying on if the word exists. To conclude, the bloom filter return more false positives if the filter is smaller than if the bloom filter size is larger.

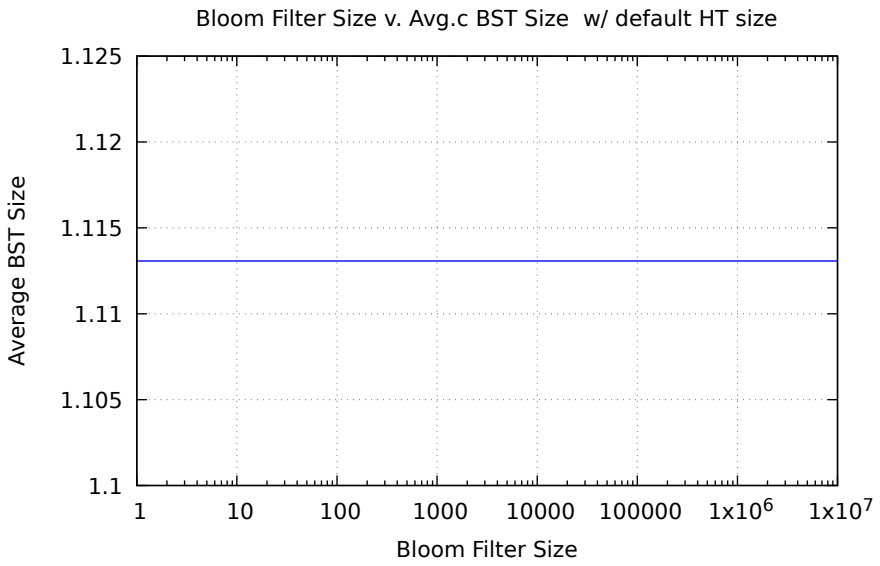

Figure 1: Relationship of Bloom Filter Size and Average Binary Search Tree Size

I notice that the graph is a straight line when the bloom filter increase, so does the average binary search tree size increase along with. This makes sense because the bloom filter size and the average binary search tree size are not directly related. However, the bloom filter is connected with the hash table and hash table is connected to the binary search tree.

Because we cannot allocate space that is already set, thus the binary search tree cannot vary.

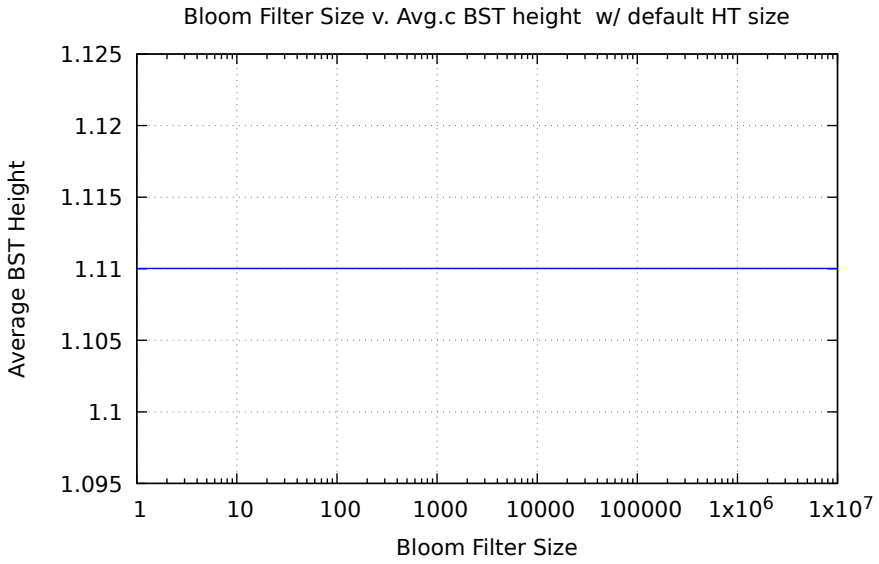

Figure 2: Relationship of Bloom Filter Size and Average Binary Search Tree Height

I notice that the graph is a straight line when the bloom filter increase, so does the average binary search tree height increase along with. This makes sense because the bloom filter size and the average binary search tree height are not directly related. However, the bloom filter is connected with the hash table and hash table is connected to the binary search tree.

Binary search tree height is the average of the sum of all nodes, going through all the nodes in the binary search tree. Because we cannot allocate space that is already set, thus the binary search tree cannot vary.

The height of the binary search should be determined by how large the hash table size is because the index array for the binary search tree the binary search tree is stored in the hash table.

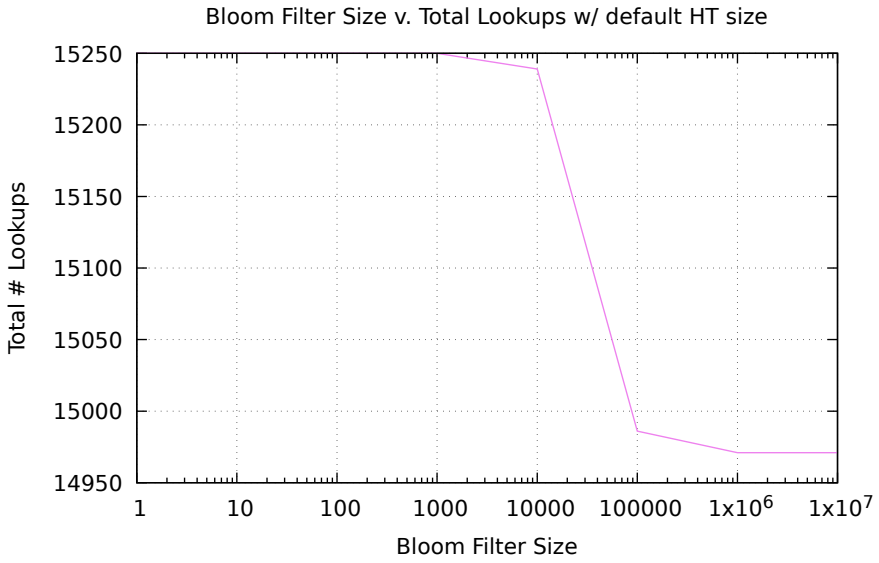

Figure 3: Relationship of Bloom Filter Size and Total Lookups

As the bloom filter size increases and as we take in more words, we have to account for more false positives. Looking back, the bloom filter checks if the word exists – if there are more available spaces in the bloom filter, there will be fewer lookups the program will do.

In the beginning, the bloom filter size returns more false positives; however, as the bloom filter size increases the false positives begin to slow down. So, at a certain point of the graph, the bloom filter's size and the total number of lookups decrease. Because the number of false-positive decreases as well.

This is also because the bloom filter and the hash tables are connected. We check if the word in the bloom filter exists then the total number of lookups will also decrease, since we are relying on if the word exists.

Lastly, the graph becomes stagnant at the end because there are only a limited amount of words we can read from the file. Hence, at that point, it will stay stable because the bloom filter size continues to increase, but there are an "x" amount of words we can read.

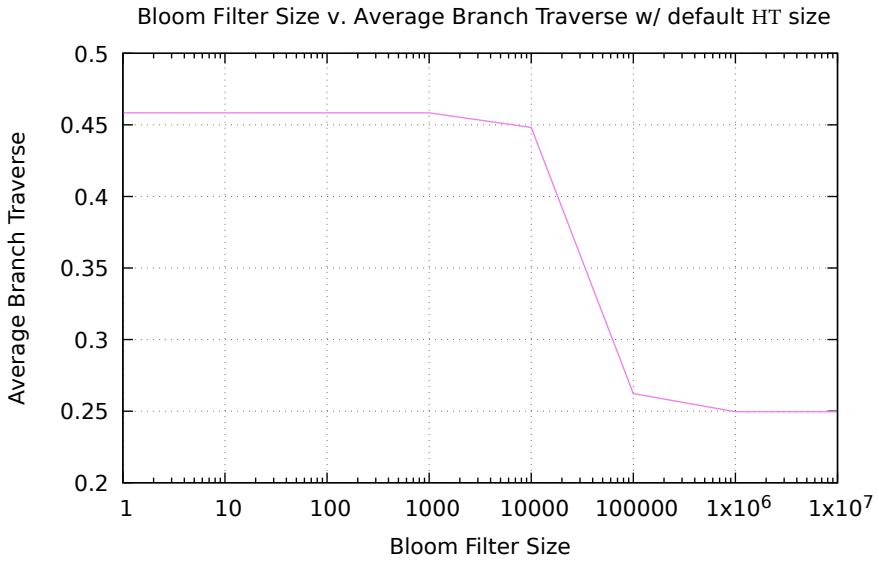

Figure 4: Relationship of Bloom Filter Size and Average Branches Traversed

This is very much similar to the bloom filter size v. the total number of lookups graph. Again, the bloom filter size increases as we take in more words. We have to account for more false positives. Looking back, the bloom filter checks if the word exists – if there are more available spaces in the bloom filter, there will be fewer lookups the program will do.

And the branches count every time we traverse through the binary search tree – or on the nodes of the binary search tree. Lookups count every time we look through the hashtable and each time we insert it into the hashtable.

To add to this, the bloom filter is directly correlated to the hashtable and the hashtable is the index array of the binary search tree. While the bloom filter increases the fewer false positives we get, which is why there is a major decrease at some point on the graph. In addition, it becomes stable at the end because there are "x" amount of words we can only read in. Another thing to

mention is that it is very much similar to the bloom filter size v. average branches traverse.

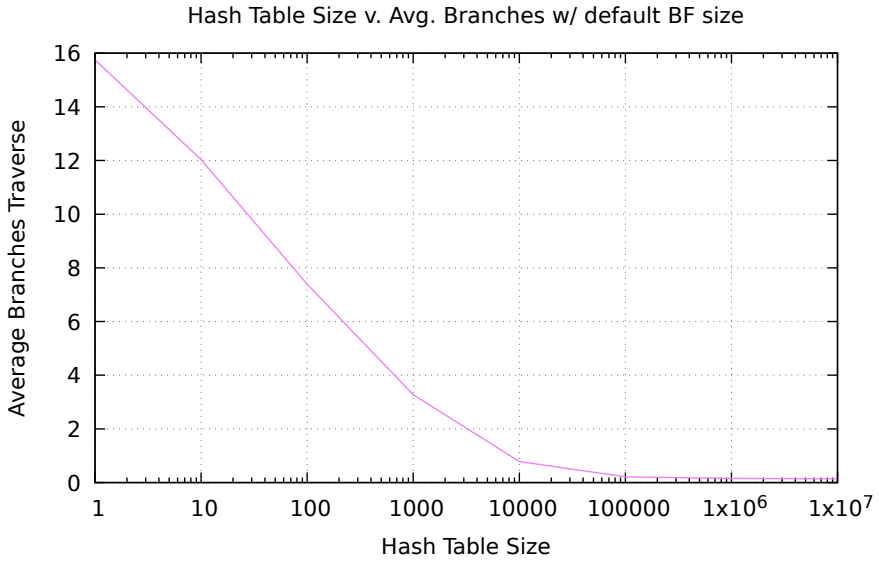

Figure 5: Relationship of Hash Table Size and Average Branches Traversed

As the graph depicts, the relationship between the hash table and the average branches traversed is decreasing in a somewhat exponential manner. This makes sense since the hash table is storing the roots. If the hash table has a smaller size it will traverse the binary tree more. Assuming the file has a lot of words to store in the hash table; therefore, there will be more lookups because the binary search tree is very large at that amount of index.

Since the graph is decreasing, this means that going through the binary search tree takes less time because the binary search trees are stored in the index at how much the hash table grows.

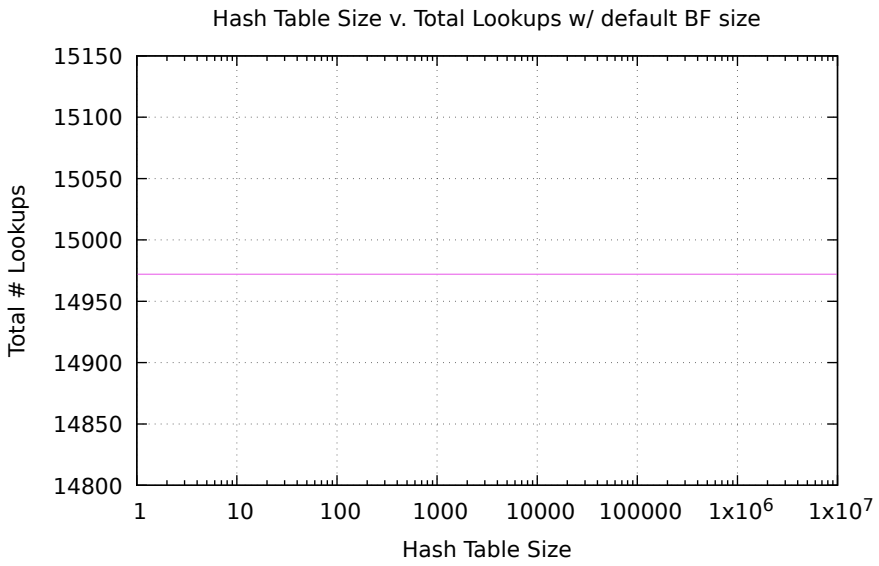

Figure 6: Relationship of Hash Table Size and Total Lookups

As the graph depicts the output is a stable line across the x-axis. Observing this, I can conclude that this graph makes sense to me, as when the hash table size increases, the number of lookups calculated does not change. The total number of lookups will not vary and will remain stable with the varying hash table size. In conclusion, the size of the hash table does not determine how the number of look ups that are computed.