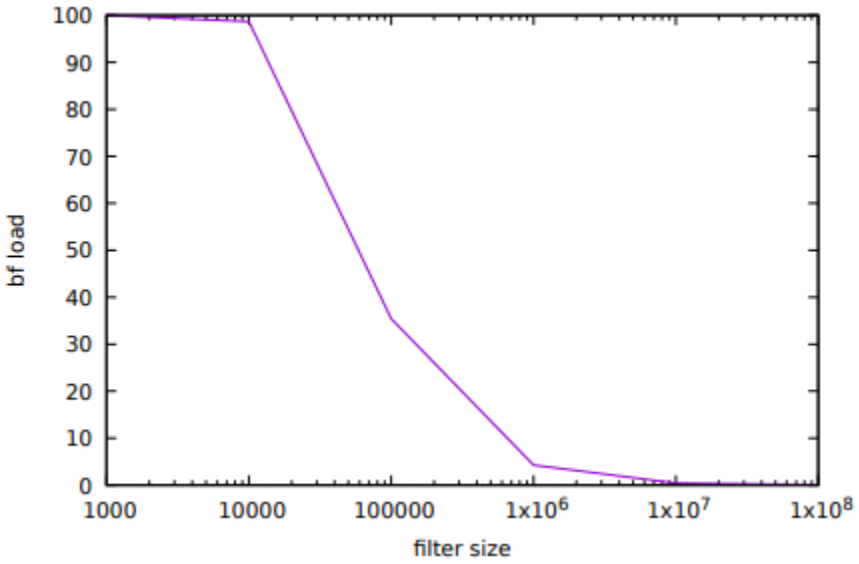
Kelly Liu
Professor Long
December 5, 2021
CSE 13S

Write Up Assignment 7
Assignment 7: The Great Firewall of Santa Cruz

In this writeup, I will be looking at the effect of increasing the bloom filter size that it has as well as the effects of increasing the hash table size. The first six graphs compare bloom filter sizes with everything else and the last six graphs compare the hash table size with everything else.

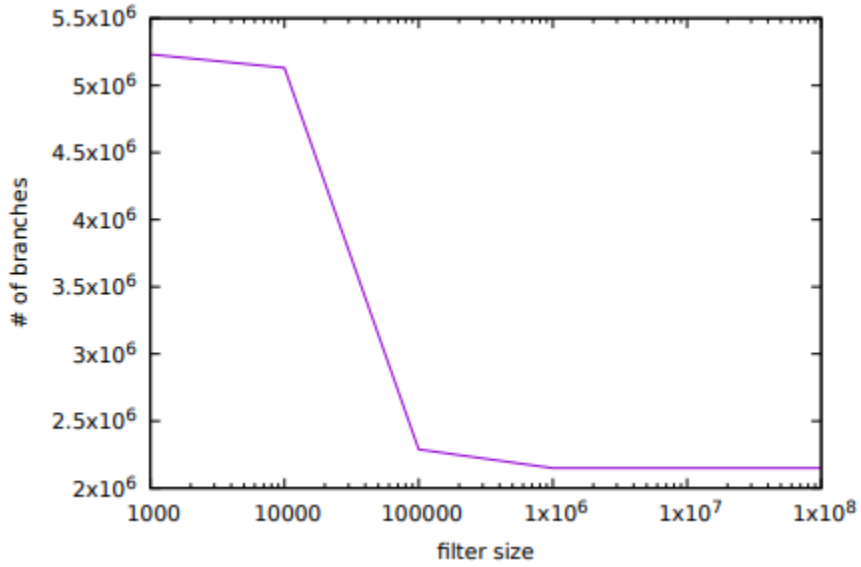
GRAPHS:

Bf Size vs Bf load



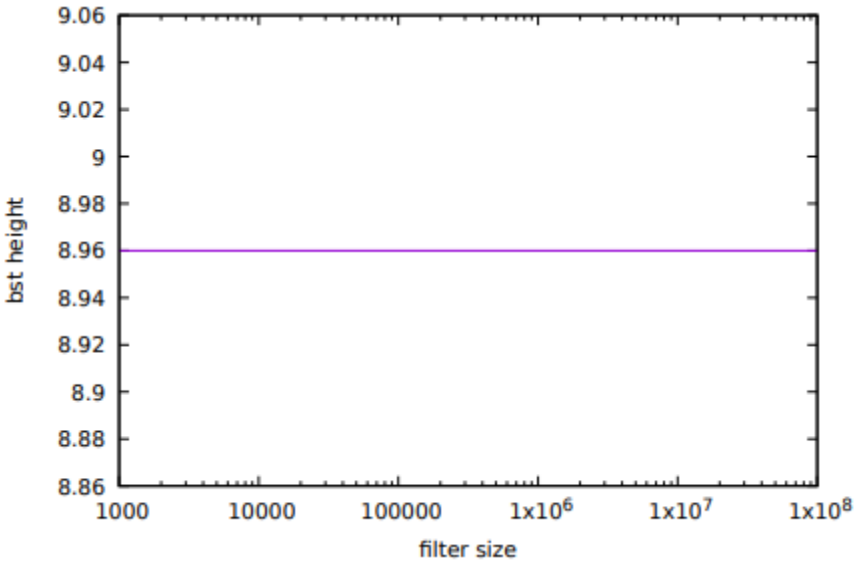
What we can tell from this graph is that, as the bloom filter size increases, our bloom filter load decreases because as bf load is bf count over bf size. This inherently tells us that as our denominator, or our bf size increases, our bloom filter load will substantially decrease at a fast rate.

Bf Size vs Branches



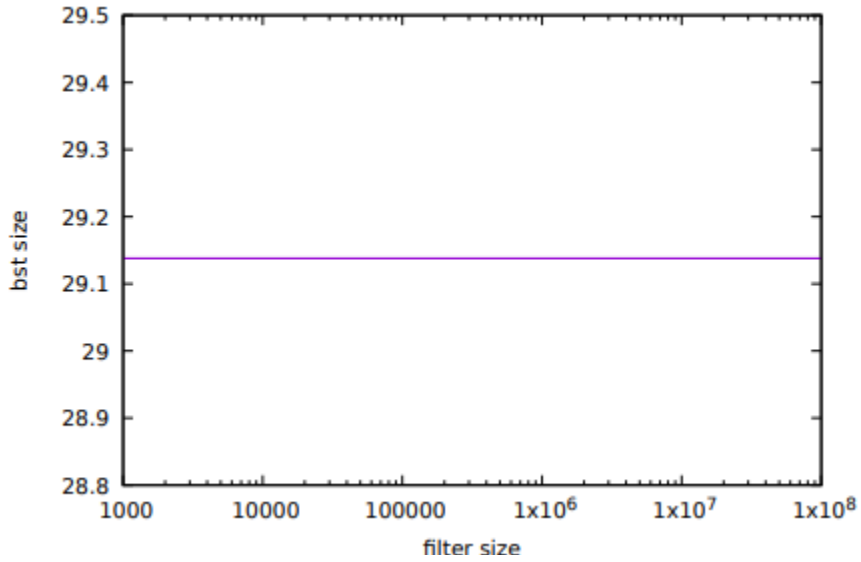
The number of branches is the count of links traversed during our calls in our binary search tree. As our filter size increases, the number of branches needing to traverse also tremendously decreases.

Bf Size vs BST Height



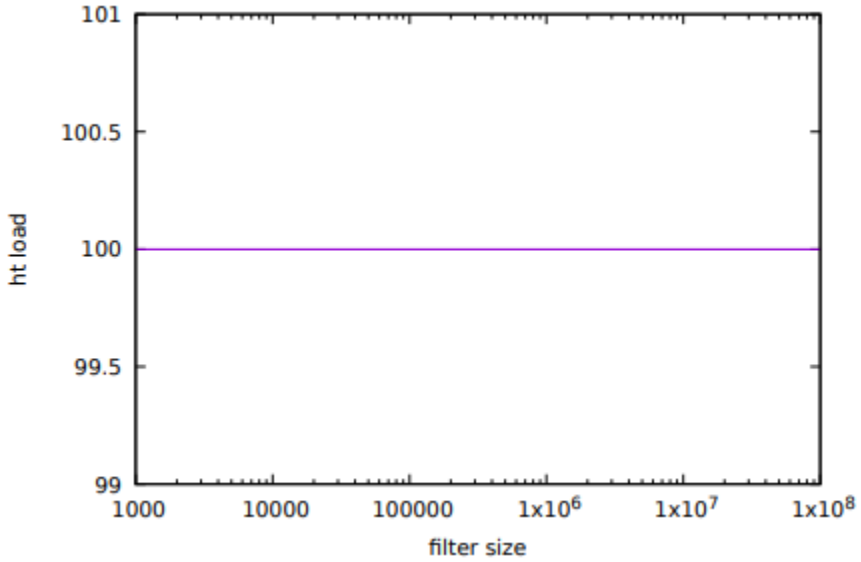
Since a bloom filter is just an infinitely big data structure, it would make sense that there should be no reason why our binary search tree height would change as the filter increases or decreases.

Bf Size vs BST Height



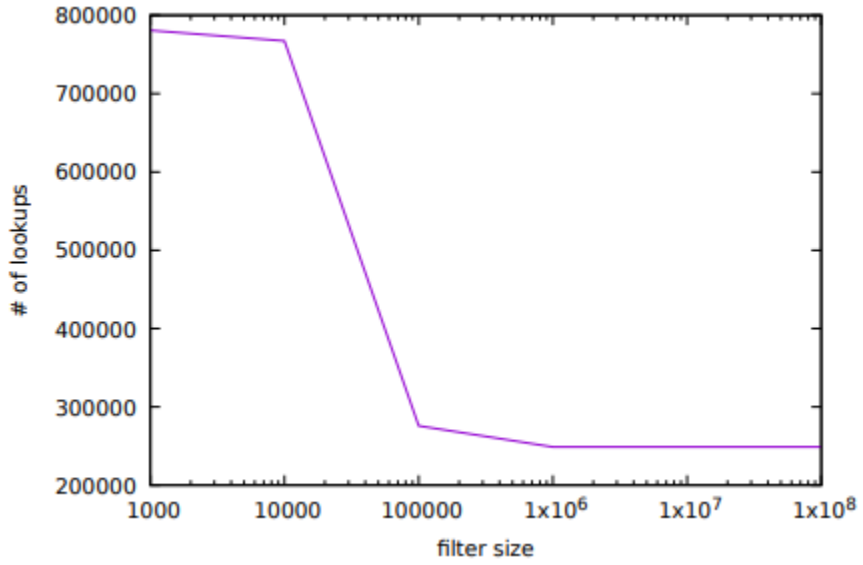
Consequently, it would make sense that our floom filter would not affect our binary search tree height as the filter increases or decreases.

Bf Size vs HT Load



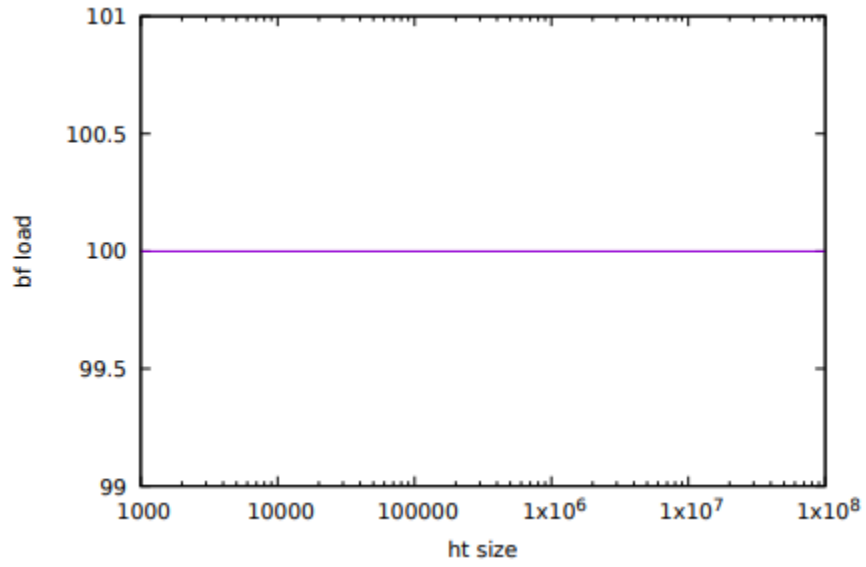
Bloom filters and hash tables are individual data structures so it would make sense that our bloom filter size would not affect our hash table load.

BF Size vs Number of Lookups



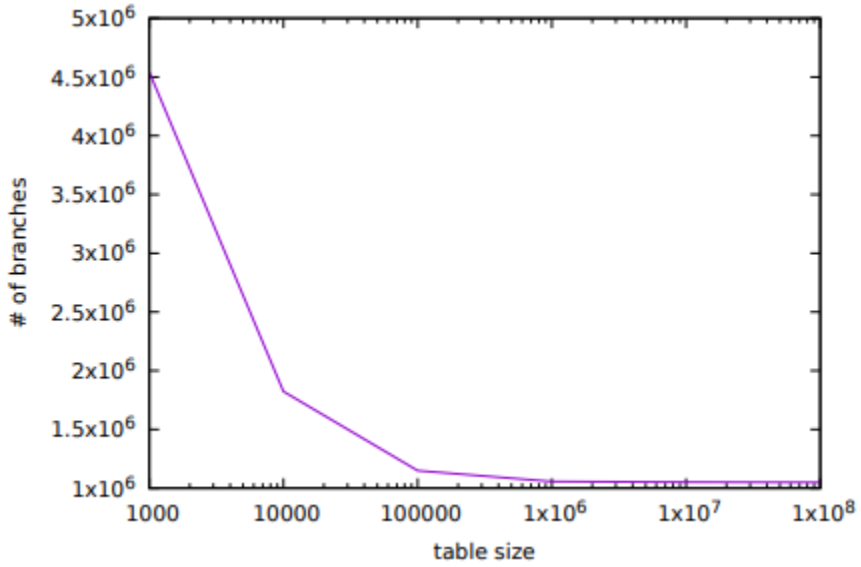
This graph is arguably the most important. We can see that as our bloom filter size increases, the number of lookups we would need to do decreases sharply until around 100,00 which then begins to stabilize. This makes sense because as we probe with our bloom filter, the large the filter is the more accurate it would be and hence the less lookups we would need to do.

Ht Size vs Bf load



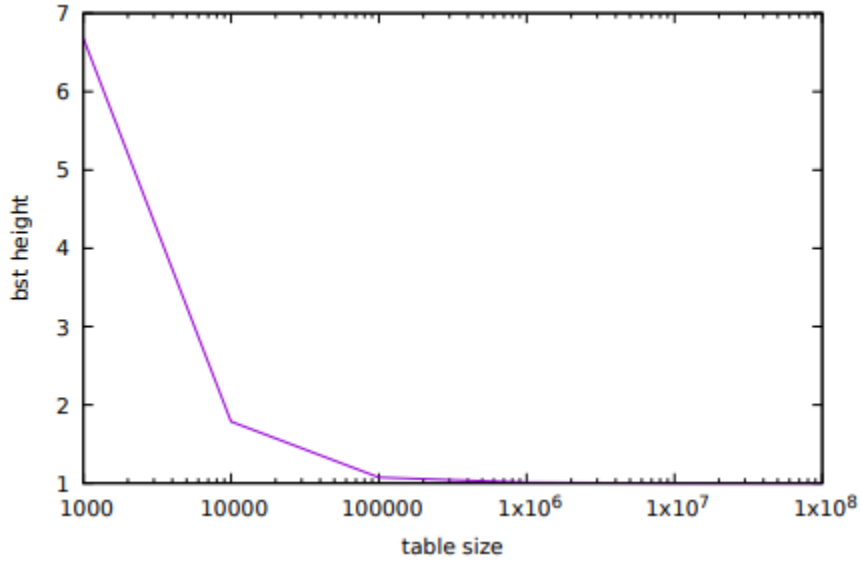
As stated above, hash tables and bloom filters are individual data structures. So like before, the hash table size will have no effect on the bloom filter load.

Ht Size vs Branches



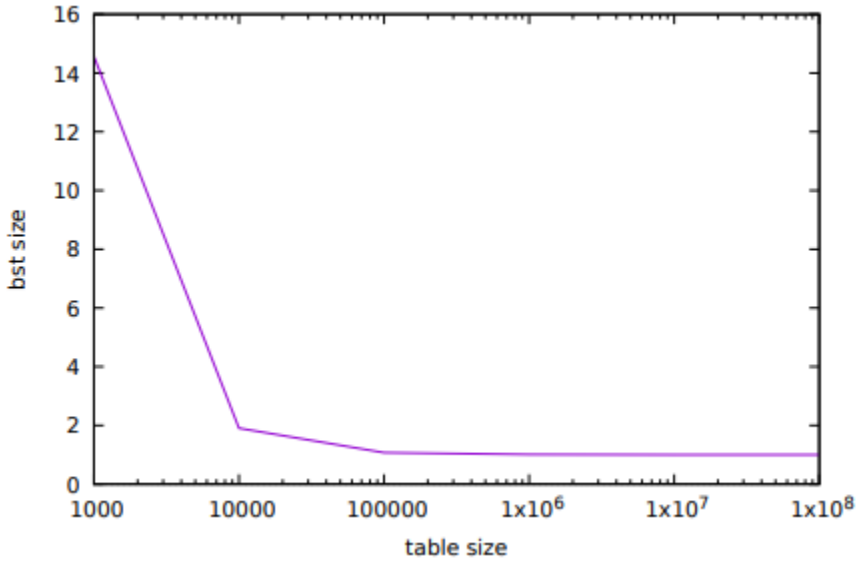
As our hash table increases, the amount of branches needing to traverse decreases tremendously. Since our hash table consists of a bunch of binary search trees which contain a bunch of nodes, as our hash table grows, we would need to traverse through less branches as our trees are inserted in an lexicographical way.

Ht size vs BST height



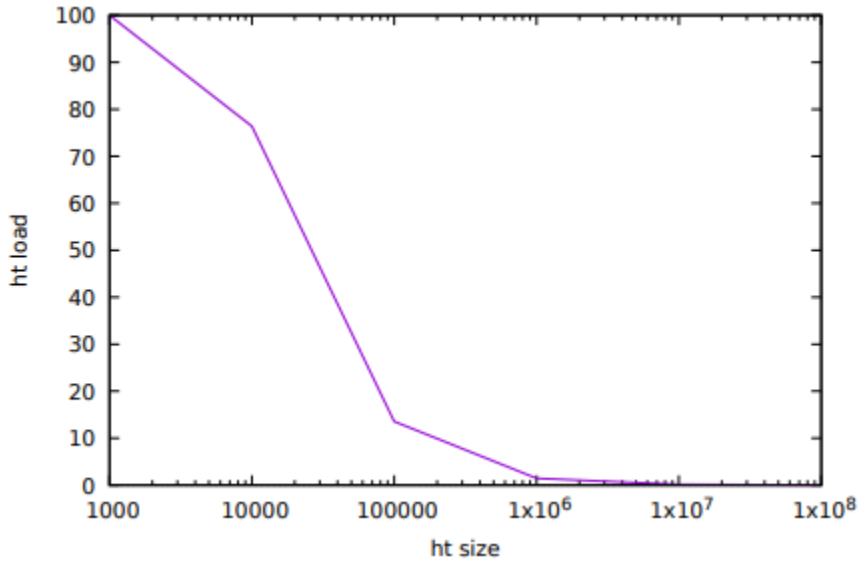
As our hash table size goes up, when we are hashing our values, the number would substantially decrease making our binary search tree height at that iteration smaller. This is shown by the graph above.

Ht size vs BST size



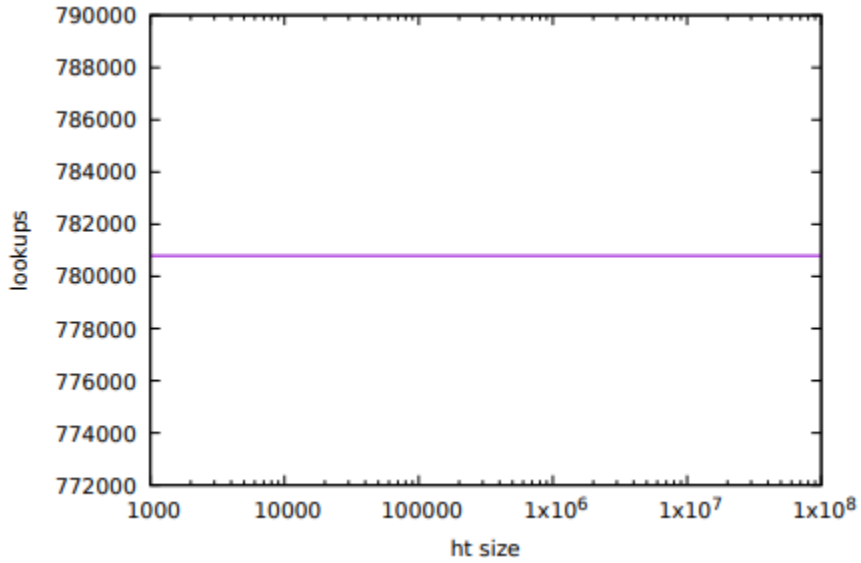
As our hash table size goes up, when we are hashing our values, the number would substantially decrease making our binary search tree size at that iteration smaller. This is shown by the graph above.

Ht size vs Ht load



Since our hash table load is computed by dividing the ht count by ht size, it would make perfect sense for our load to increase as the denominator, aka the ht size, is increasing.

Ht size vs Number of Lookups



We saw before that the number of lookups is intertwined with the bloom filter size, it would make sense that our hash table size would not affect the number of lookups we need to do.

Conclusion:
Here, I will just summarize my findings in a more organized way. Explanations/reasoning are provided in the graphs and the short analysis underneath.

Overall, we can tell that the hash table and bloom filter have no relevance to one another, in other words, changing the size of one will not affect the other. An increase in bloom filter size will decrease the bloom filter load, the number of branches needed to traverse in our binary search trees, and the amount of lookups to do. An increase in hash table size will decrease the number of branches needed to traverse, our average binary search tree height, our average binary search tree size, and our ht load.