Natalie Valett, nvalett

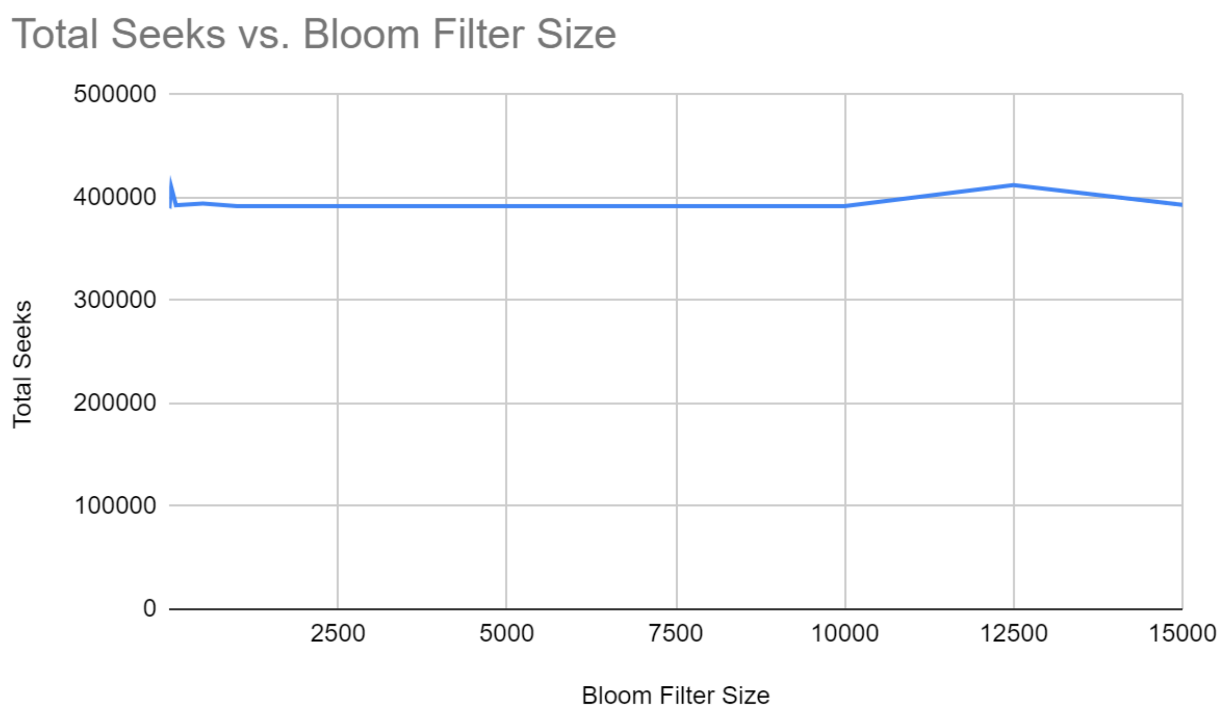
CSE13S

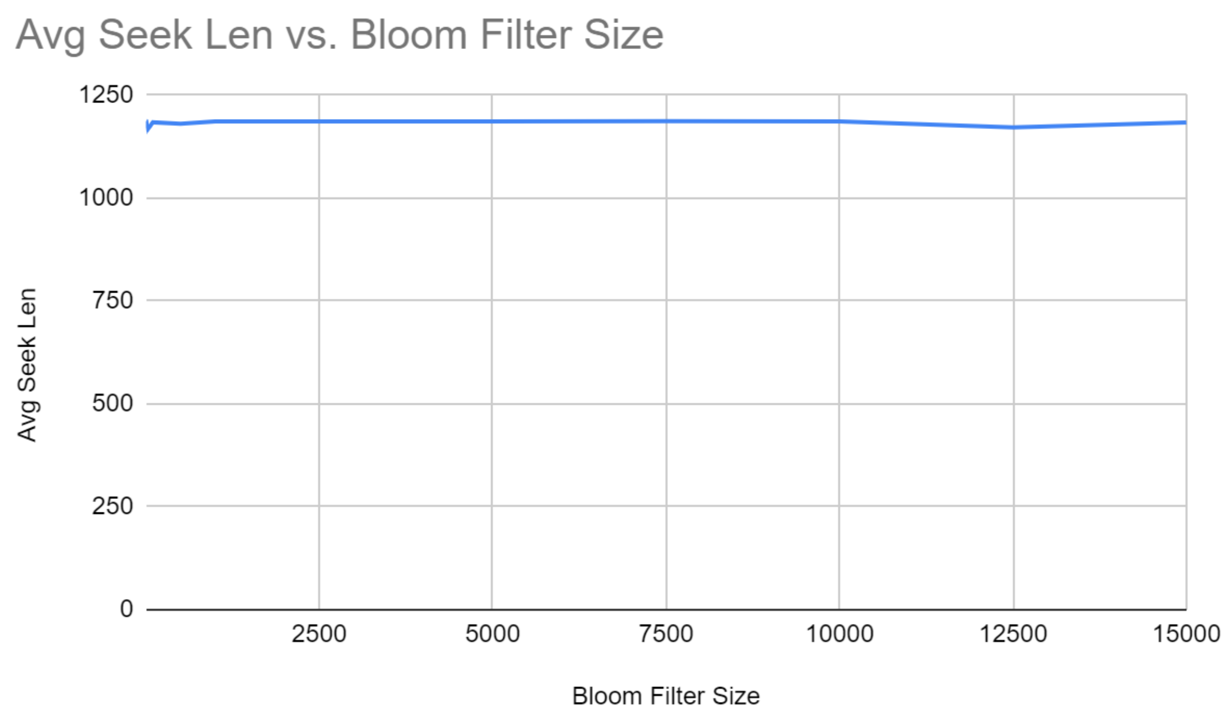
Prof. Darrell Long

June 04, 2021

Asgn7 Writeup:

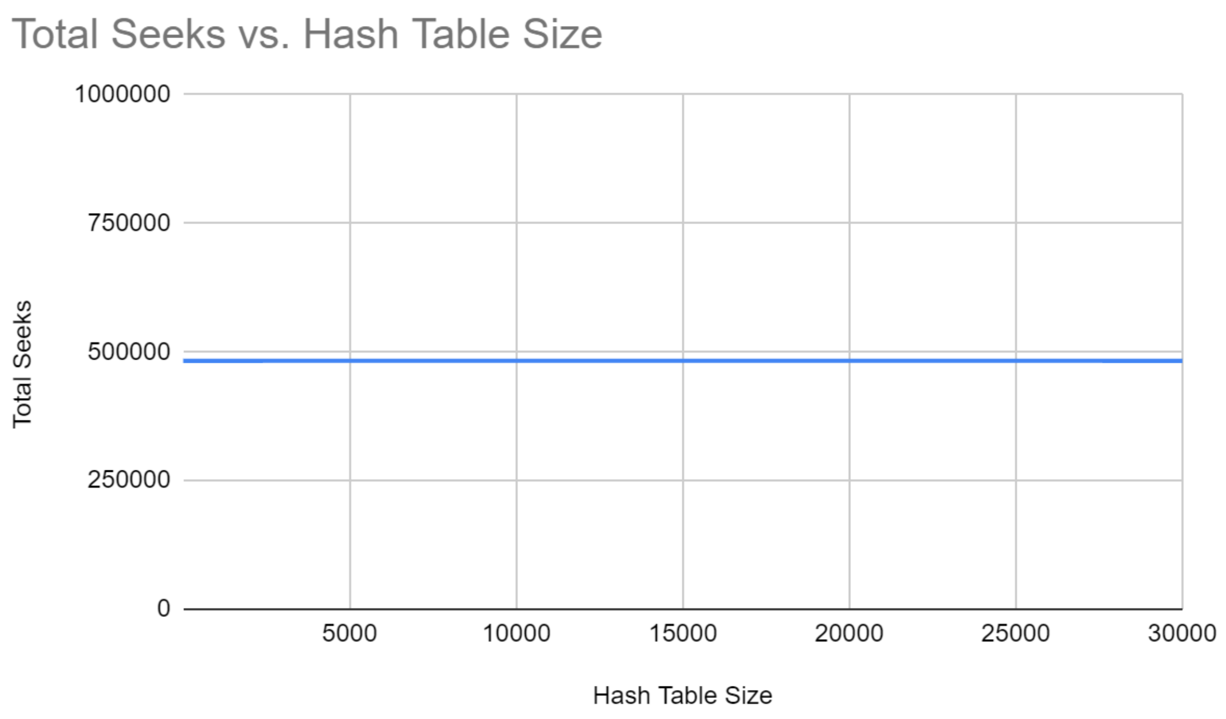
Bloom filter size and effects on seeks:

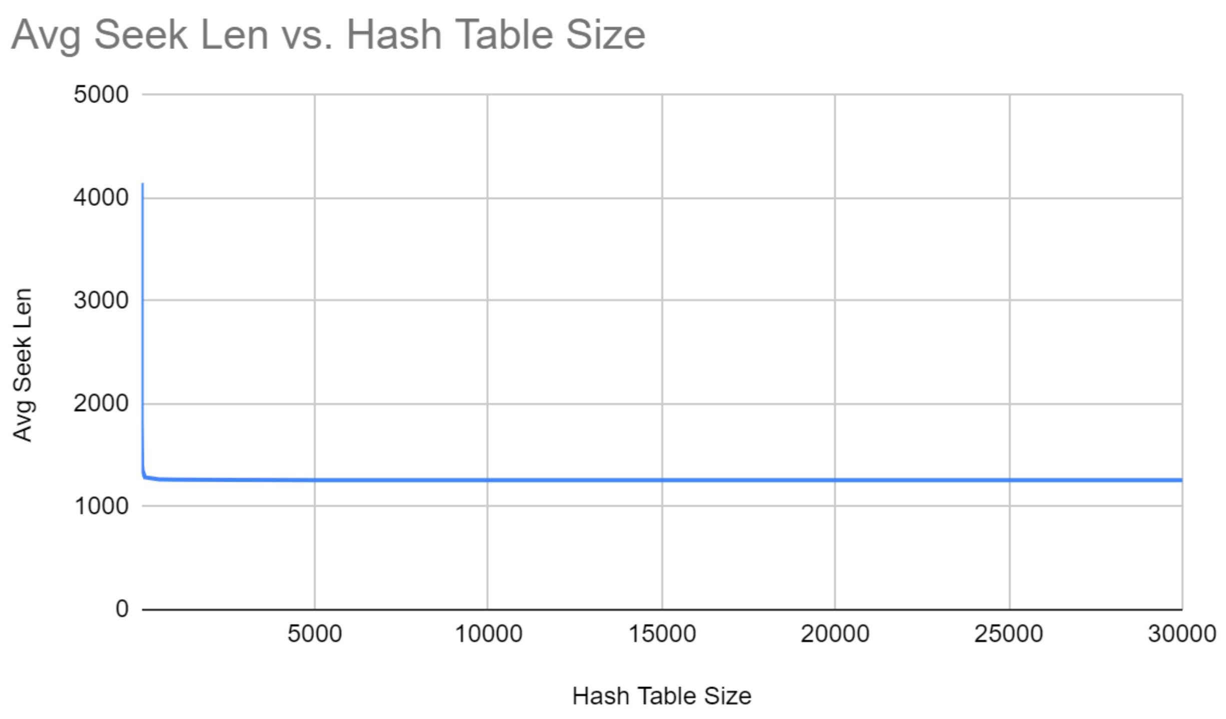




As the size of the bloom filter increases, the number of total seeks and the average seek length stay relatively constant. The size of the linked lists are unaffected by the size of the bloom filter, as they're completely unrelated data structures. The bloom filter size should only determine the frequency of false positives resulting from bf_probe() in the process of screening each word for being a bad or old word, not the construction of the hash table or any of the linked lists contained in it. The bloom filter size correlates inversely with false positives, because the less room there is in the bloom filter, the more likely it is for there to be overlap between hash values of certain words. This causes false positives because every index is more likely to hold the value of 1. Given this, it would make sense that a smaller bloom filter would result in more seeks. If more false positives are reported, then more hash table searches are required because the word will by default pass the preliminary test. Therefore, more seeks will be performed the smaller the bloom filter is.

Hash Table size and effects on seeks:





The total seeks stays constant regardless of Hash Table size, which makes sense as the size of the hash table has no bearing on the frequency at which the hash table is searched. However, the average length of each seek is affected by hash table length. The two appear to be inversely proportional, as when the hash table size is smaller, the average seek length is larger. This is because if the hash table has less indices in which to store nodes, the linked lists get longer, making the hash table grow horizontally instead of vertically (i.e. growing in length of the linked lists instead of in height of the vertical hash table). This proportionality is most noticeable at very small values of hash table size, and as the hash table grows the average length of each seek evens out to a relatively constant value. When the hash table only has one indice, the hash table basically becomes a singular linked list, which the program has to iterate through entirely to seek an element. However, when just one more indice is added to HT, the length of each linked list is cut in half, because the nodes are now distributed evenly between the two rows, as opposed to being stored in a singular long list. This allows for the average search length to also be cut in half. Each subsequent addition of a hash table index cuts down the average search length, but to a decreasing degree. This explains the exponential-like graph representation of the relationship between hash table length and average seek length.

In summary, the bloom filter size is inversely proportional to the total amount of seeks, and the hash table size is inversely proportional to the average seek length. Additionally, the bloom filter size has no correlation to the average seek length, and the hash table size has no correlation to the number of seeks performed.

1. Do linked lists get longer?
    a. Linked lists get longer the smaller the hash table is. The only determinant of linked list size is the size of the hash table, as shown in the graph “avg seek length vs hash table length”, and the number of badspeak or oldspeak words given to the program. Depending on how many bad words the hash table is required to store, and the available space it has to store

those words, the linked lists vary in size. In general, the less room the hash table has (the smaller its size), the longer the linked lists grow. Additionally, the more bad words the program is asked to screen for, the longer the linked lists are, as they have more elements to store.

2. How does the number of links followed without using the move-to-front rule compare to the number of links followed using the rule?
    a. For bible.txt, there were 465,340,140 links when using the -m move-to-front identifier, and 473,675,755 when not. This is a difference of 8,335,615 links. This is a significant decrease in the number of traversals required to find a certain node element when move-to-front is enabled. This is because it's often the case that certain words are used much more frequently than others, so moving each indexed word to the front of the list results in less computational power needed to find the most commonly searched-for words, as they consistently move to the front where they'll be able to be discovered first.
3. How does changing the Bloom filter size affect the number of lookups performed in the hash table?
    a. As discussed in the graph analysis, the bloom filter size is inversely proportional to the number of hash table lookups performed. This is because the less elements in the bloom filter, the more likely it becomes for the bloom filter to report false positives. Because the bloom filter check is used to reduce the amount of ht_lookups performed, the more it falsely

reports a positive, the more hash table lookups will be performed with words that were unable to be filtered out using the bloom filter.