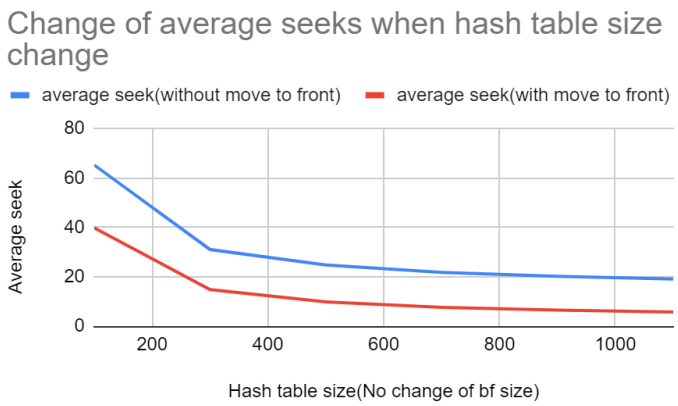
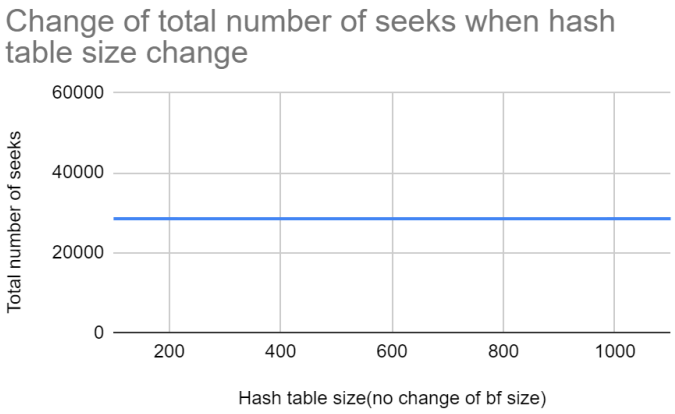
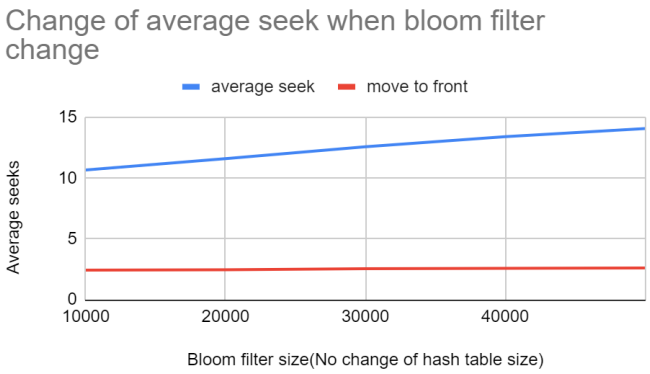
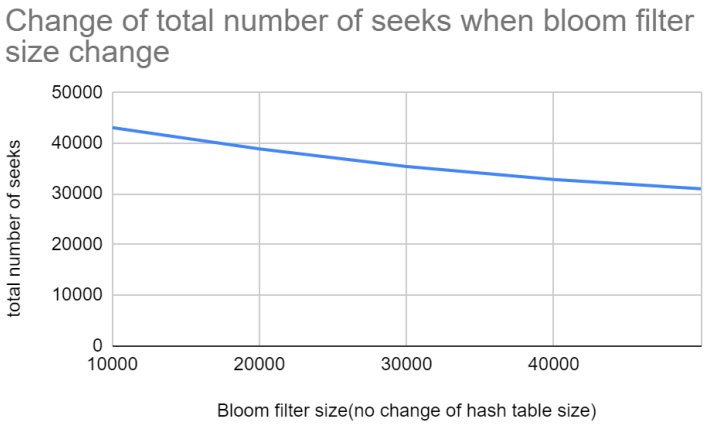
Change of hash table size





Change of bloom filter size





Analyze Graph:
Average seek length = links/seeks
Links = number of times we traverse the linked list
seek= each time we call ll_lookup

Do linked lists get longer?
As hash table size increases, average seeks will decrease. Less average seeks meaning quicker to find the bad word. Since more hash table size will decrease hash collision(shorter linked list length), it proves that greater hash table size and shorter linked list length will increase the search up time in the program.

How does changing the Bloom filter size affect the number of lookups performed in the hash table?
As bloom filter size increases, average seeks will also increase(less seeks). This is because a bigger filter size can filter more words before looking into hash table(less number of seeks), so it will increase search time in the program.

How does the number of links followed without using the move-to-front rule compare to the number of links followed using the rule?
Based on the graph, when we use the move to front rule, average seeks number is always lower than not using it(lower link number). Move to front meaning is we move forward each word we found to the front, if the same word appears next time, the search up time will be faster because the word is at the front of the linked list and we traverse less in the link list(less links).

Why does the total number of seeks not change when we use move to front rule?
Because we have to call ll_lookup no matter what, move to front rule will decrease the number of links(number of times we traverse) only.