Asgn 6 Writeup

By experimenting with different size of hash table and bloom filter, I realize the bloom filter starts to do a good job if the bloom filter is less than 60 percent loaded. Depends on how many strings we are putting in, if we put oldspeak.txt and hatterspeak.txt in. We need the bloom filter to has size of 50000 to maintain the bloom filter functional.

Bloom filter filters non-existing words a lot faster than searching in the hash table. Because it takes constant time regardless to filtering how many words are stored in the bloom filter. The bloom filter checks three hashed indices if the string, if one of the indices is not set, the word is filtered. However, bloom filter does give false positive because two different words might get exact same hash indices. Therefore, I still have to reply on the hash table to verify whether the word exist or not.

While the hash table is accurate, it’s slow. It is because we have to check all the string within the same linked list, and it has to loop till the end of the list if the word does not exist in the hash table. Therefore, having a resonable size of bloom filter reduce a lot of seeks because the bloom filter can filter the word before it searches the hash table.

The move_to_front option can also reduce the amount of seeks, because if the user keeps entering repeated word, the hash table can just return its head rather than searching the linked list again. Therefore, if the program always needs to deal with duplicated words, move to front would reduce the running time.

Also, I realize memory deallocation really depends on how many memories we allocated. For example, if we created a HatterSpeak Struct, but the string we passed was a reference or pointer, we don’t not need to deallocate the string.