ALI SALEHI ASGN 6 WRITEUP.pdf

What happens when you vary the size of a hash table?

Changing the size of the hash table in either direction, too large or too small, will be detrimental to the space and or time efficiency of the program. The ideal size would allow for a hash table close to 100% filled with an average linked list size of 1; however this is difficult to accomplish exactly. Choosing too large of a hash table size will achieve a small average linked list size, and a faster lookup time, but will not be efficient in terms of memory because of the large overhead— too many NULL linked list heads. The opposite case, results in a fully allocated hash table with large linked lists in each index, meaning lookup times will be large and the program will me slower.

What happens when you vary the Bloom filter size?

Changing the size of the Bloom filter also changes the efficiency of the program, with less of an impact on memory usage and size. Because Bloom filters consist of Bit vectors, they are relatively memory efficient, and having an overhead in size is more beneficial than risking false positives and slower run times. A Bloom filter that is too small is almost useless. It will not serve as a timely check before checking the hash table, and ht_lookup will be called when it would not need to be otherwise— causing slower run times. By contrast, a large Bloom filter is much less vulnerable to false positives and will serve as a better and more efficient check to weed out words that are not in the hash table without doing ht_lookup().

Do you really need the move to front rule?

The move to front rule does have some uses. In a scenario where only unique words are implemented by the user, the move to front rule will not cause any improvement in the run time (or specifically the average seek length) of the program. In contrast, when a certain word is inputted by the user multiple times, move to front does save time by lowering the average seek length, keeping the commonly looked for word near the front of the linked list.