syasar

CSE 13S

November 29, 2020

ASGN6 WRITE UP

When you vary the size of a hash table, its collision rate also changes. If you were to increase the size of your hash table, your indexing function uses a larger modulo, thus decreasing the chance for the same indexes when the hash-passed value is divided by the modulo. The reverse would happen if you decrease the size of the hash table; this would result in more inputs having the same indexes returned from hashing, thus more collisions, especially if the hash function is not that good to begin with. A larger hash table also takes up more space complexity and time complexity when iterating through it as you have more items to sift through.

The Bloom filter size reacts very similarly to the hash table. Increasing the bloom filter would aid in avoiding collisions but also increase the time and space complexity for inserting and probing functions. As such, decreasing the size of the bloom filter will result in more collisions, also affecting the probe functions. If there are more collisions, the probability of returning a false positive using a probe function increases, depending on the salt used in the bloom filter create function. This occurred in our hatterspeak program, especially when the bloom filter would be set with a relatively small size. However, I avoided false positives by checking if it was in the hash table.

The move to front rule occurs when looking up a node in a linked list. If the node is there, the lookup function will make sure to not only add that node, but also rewire the pointers so that node is the new head of the linked list. This makes it so that when you search up the same node again, you do not need to traverse as many nodes to find it as it will be the first one. The general principle behind this rule is that things already in the linked list will be more likely to be searched for again. As such, this decreases the time complexity of inserting the same probe into the linked list as you will not need to reiterate through all of the nodes again. However, this boost in efficiency will only be seen when repeatedly trying to enter nodes that are already in the linked list, and as such, you do not really need to use it if you are not being too picky on efficiency. It truly depends on the needs of the user and its specific case depending on the inputs of nodes/data.