Brady Chan
brmchan@ucsc.edu
11/29/2020

CSE 13s Fall2020

Down the Rabbit Hole and Through the Looking Glass:

Bloom Filters, Hashing and the Red Queen's Decrees

Write Up

In this assignment I coded a bloom filter, linked list, and hash table to efficiently store two lists of words and check if the input from the user matched these lists. First the program checks if the word is in the bloom filter, then it checks if it is in the hash table and linked list. Due to the bloom filter's false positive it needs to be checked a second time. When you vary the size of the hash table, it will change how the linked list length and how fast the program can search for words. Because the larger the hash table is, the shorter the linked list will be and the shorter the hash table is, the longer the linked list will be. Since the words are hashed and indexed in the hash table, the time complexity is O(1). So this means that after finding the correct index, if the linked list is longer, it will take the program more time to traverse the list and find the matching word. Thus a longer linked list causes the program to be slower, while a shorter linked list will be much faster. When the bloom filter size is altered, the chances of false positives are a lot higher. Since there are less bits to be changed, the chances of having overlapping bits becomes significantly higher than that of a longer bloom filter. This is because when you hash bits they change 3 bits in the bloom filter to 1 to save that specific word inside. But with a smaller bloom filter there are less bits to change and the more likely to have overlapping bits when you hash the word and get false positives. Lastly, the move to front rule is not really needed. The list of words isn't long enough for the move to front to really impact it. This is because the computers are powerful enough to go through the list regardless. However, if the list of words were longer or the linked list was extremely long, the move to front would be very beneficial because it would shorten the time the program would have to search for certain words by moving the statistically more used words to the front and the less common words would eventually shift towards the end. All in all, this lab made me understand how these different data structures work and the reason for their uses. Bloom filter is incredibly quick but

has a chance of false positives that needs to be checked. Linked lists are helpful with storing data in an easy to access way. And hash tables are helpful for storing a key and value like data similarly to a dictionary in python. This lab also gave me better practice with object oriented programming and memory. As each struct and object created in the program had to be freed, it was imperative that each helper file contain a delete to successfully free data used. This was extremely important because this simplified the helper files that relied on others. So as long as each file correctly referenced the other delete functions, they would successfully free memory without error and avoid a large amount of memory leaks.