Kevin Rodriguez

LAB 4 Report

***Introduction:***

In this lab, I had to improve the running time of the previous lab (lab3) from using binary search trees to using the hash tables. The hash table created had to solve collisions by chaining. I had to write functions that determine the average number of comparisons required to perform a successful retrieve or search operation. In order to do this, I have to create different hash functions in order to compare the average number of comparisons done.

***Solution / Implementation***

In this lab, I created a file reader that would read each word from the file and hash appropriately. My hash function grabs the ASCII value of each of letter of the word and concatenating the entire word into their ASCII value. After having the letter value, I would power it in the following way (pow(2,i)) i, being the value of each letter. In this case it covers all the permutations and anagrams. After having the entire word in its ASCII value, I mod it with the size of the table, which the result is the position in the table.

In order to see the average number of comparisons, I devided the number of comparisons by the number of used spots in the hash table and divided it by 2.

As well as, whenever you would search a word, it would return the node and the number of comparisons it took to find that word.

***Experimental Results***

In order to test my program and to calculate my average comparisons, I made 4 different hash functions with a prime number whenever I am powering the value of each letter. After that, I used a text file that wasn’t as big and would play around with the table size in order to see the average number of comparisons done in order to find a specific word.

***Conclusion***

The purpose of this entire lab was to see how we can easily fix the running time of a problem by using a hash function. But as well, it has its cons which is memory. Whenever you want a hash table that has little to none collisions, you want it to be big. Because the higher the size of the hash tables, the less the number of collisions it has. It is great to see how we are able to implement each data structure to a similar problem and improve the running time of the program.

***Signed Academic Honesty Certification***

I certify that this project is entirely my own work. I wrote, debugged, and tested the code being presented, performed the experiments, and wrote the report. I also certify that I did not share my code or report or provided inappropriate assistance to any student in the class