Jose Escobar

80536060

CS2302 Lab 4 Report

Instructor: Diego Aguirre

**Introduction**

In this lab it was required to implement a hash table to improve the running time of the previous lab and show the table load factor as well as the comparisons made.

**Proposed Solution**

Since this lab was a lot shorter than the previous one I decided to create a single file containing the node of Hash Tables, methods to create the table, get the number of comparisons and load factor, and finally the main method that would read the given file and call methods to get the expected results.

**Experimental Results**

In this lab, I used zybooks for help to implement the hash table methods and get the following results



Notice that depending on the hash table size, the load factor and comparisons made will change.

**Conclusion**

This lab helped to better understand the implementation of Hash Table and how it can improve noticeably the running times from other data structures like the ones implemented in the las lab. The implementation is much simpler and easier to understand, in my opinion, than AVL and Red-Black trees.

**Appendix**

**HashTable file**

'''

Jose Escobar

UTEP ID 80536060

CS3 Lab 4: Hash Table

'''

class HashTableNode:

def \_\_init\_\_(self, word, next):

self.word = word

self.next = next

class HashTable:

def \_\_init\_\_(self, size):#Creates the hash table

self.table = []

for i in range(size):

self.table.append([])

def hash(self, k):#Converts characters into integers

n = 0

for i in range(len(k)):

n = n + ord(k[i])

return n % len(self.table)

def insert(self, k): #Insertion Method

index = self.hash(k)

list = self.table[index]

node = HashTableNode(k, list)

list.append(node)

def search(self,k): #Search Method

pos = self.hash(k)

list = self.table[pos]

if k == list:

return k

else:

return None

def createTable(file, hashtable):#Initializes hash table with values provided by file

with open(file) as file:

for line in file:

items= line.split(" ")

if items[0][0].isalpha():

word = line.split(" ")[1]

hashtable.insert(word)

return hashtable

def numberOfComparisons(self):#Counts the number of comparisons made by dividing all elements by the occupied ones

total\_elem = 0

counter = 0

for i in range(len(self.table)):

temp = self.table[i]

if temp is not None:

counter +=1

for temp in self.table[i]:

total\_elem +=1

temp = temp.next

return total\_elem/counter

def loadFactor(self):#Devides number of elements by the size of table

num\_elements = 0

for i in range(len(self.table)):

temp = self.table[i]

for temp in self.table[i]:

num\_elements += 1

return num\_elements / len(self.table)

def main():

file = "glove.6B.50d.txt"

table = HashTable(4)

createTable(file, table)

print("Hash Table size: 4")

print("Load Factor: ", loadFactor(table))

print("Comparisons made: ", numberOfComparisons(table))

main()

**Academic Honest 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.

Jose Escobar