Govinda KC
CS2302 Lab4 Report
Instructor: Diego Aguirre
TA: Manoj Sah

Problem Description:
We had used either AVL tree or RedBlack Tree but running time was not good. So, the purpose of this lab is to improve the running time by using the hash table instead of Balanced search tree. Hash table uses the chaining to solve the collisions. For the comparison purpose, I have written multiple hash function and also, I have used the method to compute the average number of comparisons required to perform a successful retrieve operation. Also, a method to compute the load factor is written. The main purpose is to reduce the running time.

Purposed solution:
In this problem, I created the data structure of words in a hash table instead of binary tree such as AVL or Red Black Tree. At first the letters are converted into the number. All the keys are inserted into the table so that linked list is formed. I also have written the method to find the average number of comparisons. Load factor is defined as the ratio of number of elements in the table and size of the table. I have written the method to find the load factor. Search method is to search the key in the hash table. After running the code, I see that there is the significant amount of the reduction on the time.

Appendix:

```
# -*- coding: utf-8 -*-
#-------------------------------------------------------------------------#
# Course: CS 2302 Lab 4-A                                                 #
# Govinda KC                                                              #
# Instructor: Diego Aguirre                                               #
# TA: Manoj Sah                                                           #
#-------------------------------------------------------------------------#

# Node class that used for the Hash Table
class Node:
    def __init__(self, words, next):
        self.key = words
        self.next = next
# Defining the class to create the linked list in Hash table.
class H_table:
    def __init__(self, quantity):
        # initializing the table size
        self.Htab = [None] * quantity
    # Method to covert the letter into integer.
    def hash(self, k):
        s = 0
```

Govinda KC
CS2302 Lab4 Report
Instructor: Diego Aguirre
TA: Manoj Sah

```
        for i in range(len(k)):
            s = s + ord(k[i])
        return s % len(self.Htab)

    # Inserting the key into the table and so that linked list is formed.
    def insert(self, k):
        p = hash(k) # p for position
        self.Htab[p] = Node(k, self.Htab[p])

    # Search for the key in the table
    def search(self, k):
        p =  hash(k)
        temp = self.Htab[p]
        while temp != None and temp.key != k:
            temp = temp.next
        return temp

    # Method to find the average numbers of comparisons
    def average_compare(self, k):
        count = 0
        p =  hash(k)
        for i in self.Htab:
            temp = self.Htab[p]
            while temp != None:
                if temp.key == k:
                    count = count + 1
                temp = temp.next
            return count / len(self.Htab)

    # Method to calculate the load factor. Load factor is being calculated
    # by dividing the number of elements by the size of the table.
    def load_factor(self, k):
        counter = 0
        p = hash(k)
        for i in self.Htab:
            temp = self.Htab[p]
            while temp != None:
                counter = counter+1
                temp = temp.next
            return counter / len(self.Htab)
```

Govinda KC
CS2302 Lab4 Report
Instructor: Diego Aguirre
TA: Manoj Sah

```
# Main function
if __name__ == '__main__':
    # Read the given file
    file = open("glove.6B.50d.txt", encoding="utf-8")
    line = file.readline()
    file.close()
    for i in line:
        h = H_table(400000)
        number = h.hash(i)
        print("Insertion: ")
        print(h.insert(number))
        print()
        print("Search for a key:")
        print(h.search(8))
        print()
        print("The average numbers of comparison is: ")
        print(h.average_compare(number))
        print()
        print("The Load Factor is: ")
        print(h.load_factor(number))
        print()
```