Project 3: Balanced Glasses Display

Author: Diego Vela

e-mail: dvela2@csu.fullerton.edu

Instructor: Zhiyuan Yang

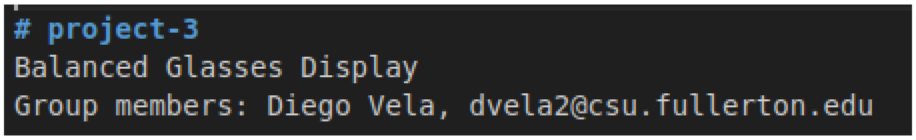
April 18, 2024

INTRODUCTION

At an optometrist store, a list of glasses are to be stored based on a digit of their ID number. Each ID number contains 7 digits from 0-9. Using a HashTable, this algorithm will determine which of the 7 digits will provide the most balanced way to store the glasses such that the difference between the most and least populated cubby is minimized.

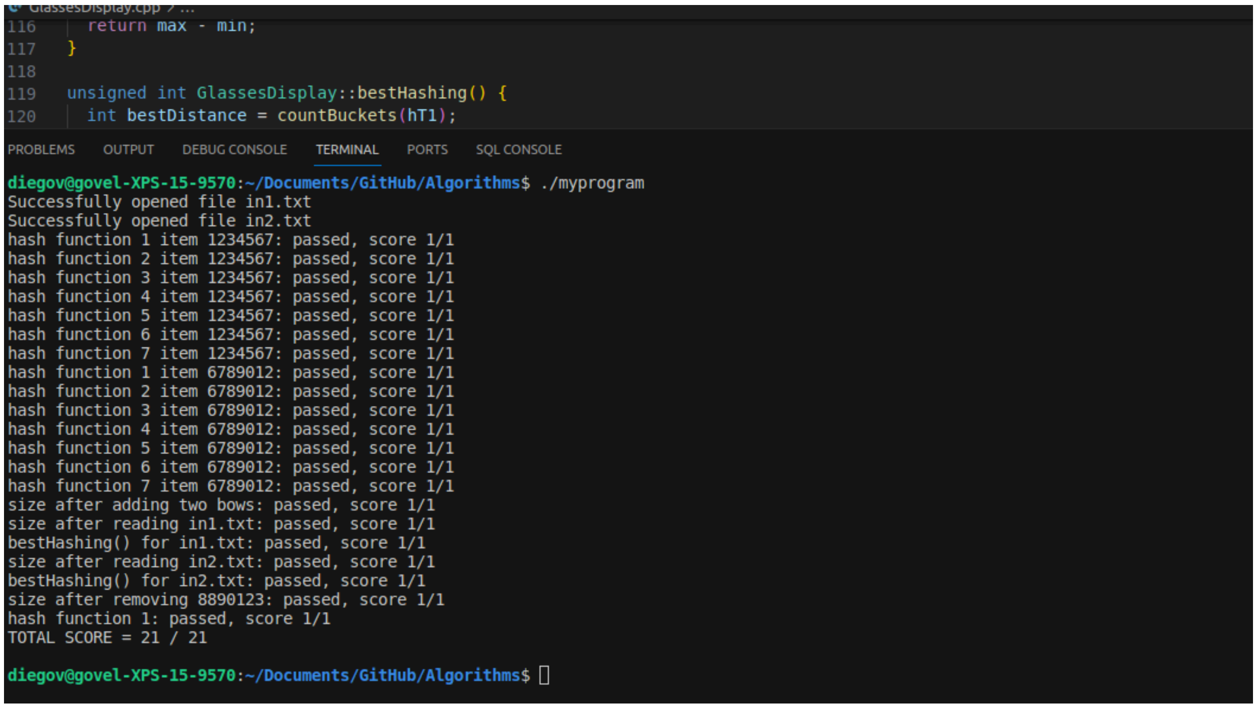
The objective of the report is to show that hashTables are an efficient way to store and access elements and can optimize algorithms. This will be done by using a mathematical time complexity analysis of the pseudocode.

README.MD



```
# project-3
Balanced Glasses Display
Group members: Diego Vela, dvela2@csu.fullerton.edu
```

CODE EXECUTION



```
  return max - min;
}

unsigned int GlassesDisplay::bestHashing() {
  int bestDistance = countBuckets(hT1);
```

```
diegov@govel-XPS-15-9570:~/Documents/GitHub/Algorithms$ ./myprogram
Successfully opened file in1.txt
Successfully opened file in2.txt
hash function 1 item 1234567: passed, score 1/1
hash function 2 item 1234567: passed, score 1/1
hash function 3 item 1234567: passed, score 1/1
hash function 4 item 1234567: passed, score 1/1
hash function 5 item 1234567: passed, score 1/1
hash function 6 item 1234567: passed, score 1/1
hash function 7 item 1234567: passed, score 1/1
hash function 1 item 6789012: passed, score 1/1
hash function 2 item 6789012: passed, score 1/1
hash function 3 item 6789012: passed, score 1/1
hash function 4 item 6789012: passed, score 1/1
hash function 5 item 6789012: passed, score 1/1
hash function 6 item 6789012: passed, score 1/1
hash function 7 item 6789012: passed, score 1/1
size after adding two bows: passed, score 1/1
size after reading in1.txt: passed, score 1/1
bestHashing() for in1.txt: passed, score 1/1
size after reading in2.txt: passed, score 1/1
bestHashing() for in2.txt: passed, score 1/1
size after removing 8890123: passed, score 1/1
hash function 1: passed, score 1/1
TOTAL SCORE = 21 / 21

diegov@govel-XPS-15-9570:~/Documents/GitHub/Algorithms$
```

MATHEMATICAL ANALYSIS

Assumptions

- Getting the first element in a bucket is 1 step.
- Getting the last element in a bucket is 1 step.
- Getting the distance between the first and last element is 1 step.

CountBucket() Helper Function: PseudoCode

First we analyze the helper function "countBuckets" to find the steps:

```
max = 0                                                      //1 step
min = INT_MAX                                                //1 step

for i in range(10):                                          //10 iterations
    first = firstElement in table                            //1 step
    last = lastElement in table                              //1 step
    distance = distance(first,last)                          //1 step

    if (distance > max or max == 0):                         //3 steps
        max = distance                                       //1 step
    if (count < min or min == INT_MAX):                      //3 steps
        min = distance                                       //1 step

return max - min                                             //1 step
```

$$Total\ Steps\ =\ 1\ +\ 1\ +\ 10(1\ +\ 1\ +\ 1\ +\ (3\ +\ 1)\ +\ (3\ +\ 1))$$

$$Total\ Steps\ =\ 113$$

bestHashing() function: PseudoCode:

Now we know how many steps the helper function is to plug into the main algorithm.

```
bestDistance = countBuckets(hashTable1);                     //114 steps
bestTable = 1;                                               //1 step

//Note: In the code, this loop is individually done.
for each currHT 2 through 7:                                 //6 iterations
    currentDistance = countBuckets(currHT)                   //114 steps
    if currentDistance < bestDistance:                       //1 step
        bestDistance = currentDistance                       //1 step
        bestTable = currHT                                   //1 step

return bestTable
```

Mathematical Analysis

$Total\,Steps\ =\ 114 + 1 + 6(114 + 1 + 1 + 1)$

$Total\,Steps\ =$ 817

817 $constant\ steps\ belong\ to\ BigO(1)$

CONCLUSION

In this project, the efficiency of hashTables were proven by finding an efficient way to store a set of glasses based on a digit in an identifying barcode. Using a bestHashing() method, we compared 7 different hashTables that stored the glasses using that digit. Then we returned the hashTable with the best distribution of storage for the glasses. We analyzed the number of steps in the function and discovered that the size of items would not change the time complexity. Although there are a large number of individual steps in each calculation, the time complexity is constant and is consistent with the hypothesis that hashTables are efficient data structures when it comes to storing and accessing data.