

CSCE 221: Programming Assignment 5 Regex and csv files

Due on Tuesday, April 13, 2017

Dr. Leyk 4:20pm

Robert Quan

Abstract

In this assignment we create a HashTable to store the entry UINS from students taken from a excel .csv file. Then we use the HashTable to search for specific UINs and enter their corresponding grade values. The output of this program is a final csv file that contains the quiz values for those students that completed the quiz and submitted their score. I also included some statistics for the table that include the minimum, maximum and average length of the linkedlists within the HashTable.

Problem 1

a) *Describe the Data Structure and algorithms used by your program?*

In this program we use the HashTable data structure which relied on a hash function that is:

$$h(x) = k * \text{mod}(m) \quad (1)$$

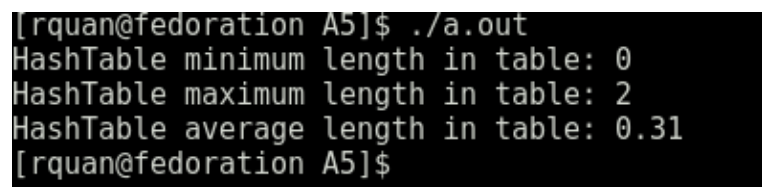
Where k is the UIN value entered and m is the hash table size which in our case is $m = 100$. The HashTable is stored in a vector of Doubly Linked Lists which then will store the key values and quiz grades for the students. In our case, our vector is of size 100 and contains between 0 and 1 linked lists with nodes. Each node contains a key and a quiz value variable.

We also implement the Doubly Linked List Abstract Data Structure and a HashNode Data Structure. These two data structures were modified for this assignment from the templated version that was used in Programming Assignment 3.

b) *Describe the input and output data and any restrictions*

A very important restriction on the input data is that the third column and fourth of the spreadsheet only contains digits because the regex code is respectively. Here we are assuming that the UIN is a 9 digit number and the quiz grade is a number that is smaller than 100. Another restriction we have imposed is that the size of the HashTable must be 100 entries. I have tested my program for correctness by creating a new input csv which contains repetitions of UINS and quiz grades. This was to test the functionality and the correct values for the maximum numbers of nodes in the link list. Both were a success and I have included the input2.csv file in the .tar to prove it.

c) *Have you tested your program for corrections?*

A terminal window with a black background and green text. The prompt is [rquan@federation A5]\$ and the command is ./a.out. The output consists of three lines: HashTable minimum length in table: 0, HashTable maximum length in table: 2, and HashTable average length in table: 0.31. The prompt returns to [rquan@federation A5]\$.

```
[rquan@federation A5]$ ./a.out
HashTable minimum length in table: 0
HashTable maximum length in table: 2
HashTable average length in table: 0.31
[rquan@federation A5]$
```

Figure 1: Testing the code for correctness

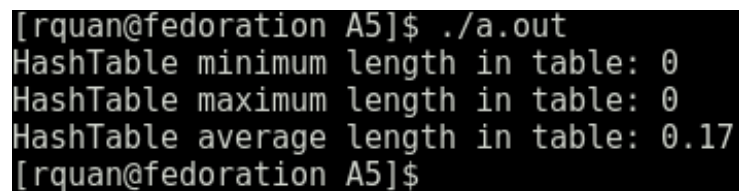
Yes, to test for correctness I created another input file which contained the same UIN for two quiz grades. Theoretically this should not be allowed, but I was curious as to see how the Linked List would compute.

As a result, the list computed correctly and the maximum number of nodes was updated accordingly. The figure 1 demonstrated the statistics when I use input2.csv

d) *Which C++ features or STL classes did you implement*

I used the class `cstdlib`, `iostream`, `stdexcept`, `vector`, `string`, `fstream` and `regex`. All of these classes aid in the implementation and the output of the csv files. As for the c++ features, I implemented the string to integer function `stoi` on the matches from my regex pattern.

e) *Provide statistics about the hash table*

A terminal window with a black background and green text. The prompt is [rquan@fedoration A5]\$ and the command is ./a.out. The output shows three statistics: HashTable minimum length in table: 0, HashTable maximum length in table: 0, and HashTable average length in table: 0.17. The prompt returns to [rquan@fedoration A5]\$.

```
[rquan@fedoration A5]$ ./a.out
HashTable minimum length in table: 0
HashTable maximum length in table: 0
HashTable average length in table: 0.17
[rquan@fedoration A5]$
```

Figure 2: HashTable statistics

When using the original input given to us, I obtain the statistics displayed in figure 2. This is correct because the maximum number of nodes that contain a quiz grade will be 1 for every individual UIN that has a corresponding grade. The minimum is 0 for those lists that do not contain values for quizzes. The average was the total number of quiz UIN grades per Total UINS. This gave me an average of 0.17 quizzes per UIN.

f) *Conclusion*

It is nice to see that the code for HashTables works, but I do not see the benefit from indexing a vector. From my understanding, If we do include extra nodes then those nodes would theoretically aid in the run time for searching a HashTable. As the code stands, I do not see how indexing a 100 entry vector with link lists and then using only one node per link list to store values aids us in searching or minimizes space.