Project 3: Balanced Glasses Display (35 points)

CPSC 335 - Algorithm Engineering
Summer 2023
Instructor: Doina Bein (dbein@fullerton.edu)

Abstract

In this project you will design and implement one algorithm related to hash tables that can aid a store clerk make maximum use of a display for glasses. Your project is about reading a large number of distinct barcodes for glasses, each being a 7-digit barcode, and deciding which digit among the seven digits gives the best balanced distribution of the glasses. You will implement the algorithm using C++, compile, test it, and submit the files. No algorithm analysis is needed for this project and no PDF report.

The Hypothesis

This experiment will test the following hypotheses:

1. Different hash functions produce hash chains of various lengths when applied to the same dataset.
2. Hash tables offer a relatively fast way of storing and retrieving data.

The Problem

At an optometrist store, glasses of various sizes and shapes are displayed in a display that has 10 cubbies or sections. Each pair of glasses is unique and has a 7-digit barcode. You have too many pairs of glasses to store them in the same cubby, plus only few would be visible. You are tasked with organizing the pairs of glasses into cubbies such that most of them are visible.

Image taken from https://www.etsy.com/listing/682739046/custom-personalized-logos-signs-123456?gpla=1&gao=1&&utm_source=google&utm_medium=cpc&utm_campaign=shopping_us_e-accessories-sunglasses_and_eyewear-glasses_cases&utm_custom1=_k_Cj0KCQjwreT8BRDTARIsAJLI0KI4nu7Zuk1nETGm8gAMxOcCL8SoDD-0SrZP3f1986nitMp_WZ93gMMaAu7yEALw_wcB_k_&utm_content=go_1844702577_72372880000_346397601745_pla-352597983086_c__682739046_134985293&utm_custom2=1844702577&gclid=Cj0KCQjwreT8BRDTARIsAJLI0KI4nu7Zuk1nETGm8gAMxOcCL8SoDD-0SrZP3f1986nitMp_WZ93gMMaAu7yEALw_wcB

You came up with an idea of storing the pairs of glasses based on a single digit of their product number. So you try to group them based on the first digit, second digit, .., seventh (and last digit). (The digits are read from left to right; e.g. for the number 8976565, the first digit is 8, the second is 9, the third is 7, the fourth is 6, the fifth is 5, the sixth is 6 and the seventh and last is 5.) You settle on the digit that gives the best "balancing" of storing the pairs of glasses, i.e., the difference between the number of pairs of glasses in the most populated cubby and the least populated cubby is minimized among all possible seven options.

Your project is about reading a large number of distinct product numbers (7-digit each) and deciding which digit among the seven gives the best balanced storage of the pairs of glasses.
For example, let's assume that we have the following product numbers:
1234567, 2345678, 3456789, 4567890, 5678901, 6789012, 7890123, 8901234, 9012345, 5432109, 6543210, 7654321, 8765432, 9876543, 1098765, 2109876, 3210987, 4321098

If we choose storing the glasses based on the first digit that we have:
Cubby 0: no pair of glasses
Cubby 1: 1234567, 1098765
Cubby 2: 2345678, 2109876
….
Cubby 9: 9012345, 9876543
The difference in the cubbies' load is 2.

If we choose storing based on the second digit:
Cubby 0: 9012345, 1098765
Cubby 1: 2109876
Cubby 2: 1234567
…
Cubby 9: 8901234,
The difference in the cubbies' load is 1 so clearly this is a more balanced organization of the pair of glasses.

Implementation

You are provided with the following files.

1. `README.md` contains a brief description of the project, and a place to write the names and CSUF email addresses of the group members. You need to modify this file to identify your group members. This information will be used so that we can enter your grades into Canvas. To be completed.
2. `LICENSE` contains a description of the MIT license.
3. `main.cpp` contains tests to check on the correctness of the function members. This is provided for you to use to test your software as you are writing it. You may change this file to add helpful functions for your own testing. I will test your project with a different but similar file.
4. `GlassesDisplay.cpp` and `GlassesDisplay.hpp` contain the skeleton of the classes and function member
5. `in1.txt` contains 18 types of glasses
6. `in2.txt` contains 36 types of glasses

The code to decide which digit leads to the most balanced hashtable is to be implemented in class . Since you will be comparing seven hashtables (which differ only in their hash function), class `GlassesDisplay` only has seven hash tables as its member variables. The other member functions are:

1. `addGlasses()`: Given information about one pair of glasses, create a Glasses object and insert into each of the seven hashtables. Note that each hash table has the product number as the key and a `Glasses` object as the value. To be completed.
2. `removeGlasses()`: Given product number, remove the corresponding pair of glasses from each of the seven hashtables. To be completed.
3. `bestHashing()`: The logic to calculate the balance for each of the seven hashtables, and then identifying the hashtable with the best balance should go into this method. Here, balance is defined as the difference between the sizes of the largest bucket and smallest bucket. Only check the first 10 buckets! (If the lowest balance factor is shared by more than one hash table, then return the first hash table with that lowest balance factor. For example, if both hT3 and hT7 have the lowest balance factor, then return the number 3. If hT2, hT4, and ht6 all share the lowest balance factor, then return the number 2.) Some hints on how to get the number of items in each bucket are included. To be completed.
4. `readTextfile()`: The list of pairs of glasses are in a text file. This method calls `addGlasses()` for each line. The code to read from the text file is already given.

The seven hash tables will differ in only the hash function that they will use. You are to provide code for these hash functions. Each hash function will take a 7-digit number and return either the first, second, etc. , seventh (last) digit.

- `hashfct1(number)`: return the first digit of `number`. To be completed.
- `hashfct2(number)`: return the second digit of `number`. To be completed.
- `hashfct3(number)`: return the third digit of `number`. To be completed.
- `hashfct4(number)`: return the fourth digit of `number`. To be completed.
- `hashfct5(number)`: return the fifth digit of `number.` To be completed.
- `hashfct6(number)`: return the fourth digit of `number`. To be completed.
- `hashfct7(number)`: return the fifth digit of `number`. To be completed.

Obtaining and Submitting Code

This document explains how to obtain and submit your work:

GitHub Education Instructions

Here are the invitation links for the projects:

https://classroom.github.com/a/1-wFqNOR

What to Do

First, add your group member names to `README.md`. Implement all the skeleton functions in the provided header file. Use the test program to check whether your code works.

When you go to commit files in git, make sure you are not checking any generated executable/object files into git. Use the *.gitignore* file to accomplish this.

Grading Rubric

Your grade will consist of two parts: *Form* and *Function*.

Function refers to whether your code works properly as defined by the test program. We will use the score reported by the test program as your Function grade.

Form refers to the design, organization, and presentation of your code. A grader will read your code and evaluate these aspects of your submission.

The grading rubric is below.

1. Function - 21 points: passes all the tests
2. Form - 14 points
    a. README.md complete: 3 points
    b. Style (whitespace, variable names, comments, etc.): 3 points
    c. Design (where appropriate, uses encapsulation, helper functions, data structures, etc.): 4 points
    d. Craftsmanship (no memory leaks, gross inefficiency, taboo coding practices, etc.): 4 points

Deadline

The project deadline is June 25, 11:59 pm.

You will be graded based on what you have pushed to GitHub as of the deadline. Commits made after the deadline will not be considered. Late submissions will not be accepted.