Hash Tables

COSC2030 Lab9
November 13, 2017
Due: Nov 20 @ 12:01pm

In class we have been discussing hashing and the storing of hashes. In this lab you will now implement a hash table that uses two different methods for dealing with collisions in that table. The two methods you will implement, which were outlined in the slides, are Open Addressing and Separate Chaining. Those slides have been uploaded to WyoCourses under Files/hash-102.pdf. Revisit them to refresh yourself about what the structures look like.

Project Setup

By now you have pulled the lab repository from github. There are a lot of files so here is a quick description:

1) **modified Node and LinkedList from Lab2 - You will use these for the Separate Chaining Method**
2) **GeneralHashFunctions - An Open Source Library containing different hashing functions, see the example file marked GeneralHashTest.cpp for usage**
3) **HashTables - This is where you will implement your two methods**
4) **main.cpp - Main driver file, read input files and perform all your tests here**
5) **randoWords.txt - file containing 50 lines of 5 random words each**
6) **names.txt - file containing 400 random names**
7) **stlHashEx.cpp - example program demonstrating the stl::hash function**
8) **unordered-mapEx.cpp - example program demonstrating the stl::unordered_map data structure which is the base structure for our HashTables**

If on a Windows machine you will need to scan through the .h files and fix any includes statement at the top that is marked with "change to .h"

Begin Coding

Now import these files into whichever editor you are using. After looking through the code you will find code stubs marked with a TODO which is where you will fill in the code to perform the following steps:

1) **Add functionality to the two addToTable functions in HashTables.cpp. The key for our unordered_map with be the hash value generated from a hashing function, and the value will be a custom bucket struct that is defined in HashTables.h.**
2) **Read in names.txt and randoWords.txt line by line (code from lab1 comes in handy here). Each line of the file will be hashed and used as the hash key.**
3) **Create a HashTables object for each input file, and use the stl::hash function to generate a hashkey which you will use to call HashTables::AddToTables(hashkey, test) function for each line of each input file. At the end of this section you should have 2 HashTables objects.**
4) **Repeat Step1 but now use the WeakHash() function I added to the GeneralHashFunctions class. At the end of this section you should have 4 HashTables objects.**
5) **Repeat Step1 again but now choose any two other hashing function you want from the GeneralHashFunctions class. At the end of this section you should now have 8 HashTables objects.**
6) **Edit the README of you repository to report the number of collisions found using each insert method, for each input file, for each hash function using the following tables:**

RandomWords:

Function	Open Addressing	Separate Chaining
stl::hash	numCollisions	numCollisions
WeakHash	numCollisions	numCollisions
HashPick1	numCollisions	numCollisions
HashPick2	numCollisions	numCollisions

Names:

Function	Open Addressing	Separate Chaining
stl::hash	numCollisions	numCollisions
WeakHash	numCollisions	numCollisions
HashPick1	numCollisions	numCollisions
HashPick2	numCollisions	numCollisions

Grading

This lab is worth a total of 20 pts with the breakdown as follows:

1) **5pts - stl::hash implemented and tested for both input files - 5pts**
2) **5pts - WeakHash implemented and tested for both input files5pts**
3) **5pts - Pick one other hash implemented and tested for both input files**
4) **5pts - Pick another hash implemented and tested for both input files**

2.5 points will be taken off per point-item if one storage method (Open Address/Separate Chaining) is incorrect.

5 points will be taken off per point-item if both storage methods are wrong.