Lab 8: Hashing Competition

1. Introduction

In the lecture today we talked about the hash table as well as chaining, which is one of the approaches to deal with collisions. In this lab, you will implement your own hash function for an English words dataset, and the goal is to minimize the length of the longest chain in the hash table using your own hash function. In order to make this lab engaging, the score of your hash function (See Section 2 for more details) will be shared to find the best hash function in class later on.

The code skeleton is under: **/comp/15/files/l8**

1. Log in to the CS homework server.
2. Move to your **comp15** directory that you created in Lab 1.
3. Create a directory named **lab8** under the comp15 directory.
4. Move to the lab directory.
5. Copy the code skeleton to the current working directory.
6. Leverage the features of Git to manage your progress toward writing the program.

The code skeleton provides you with three source files: **test.cpp**, **hasher.hpp**, **hasher.cpp**, as well as five object (relocatable) files: **main.o**, **HashTable.o**, **LinkedList.o**, **SLLNode.o**, **Item.o**.

- Note that you will implement your hash function in **hasher.cpp**.
- The HashTable class, which has already been implemented for you, will leverage your hash function to store given English words (strings). (The hash table holds instances of the Item class which just wrap the given string data.)
- The English words dataset contains **479,828** unique words and can be found at: **/usr/share/dict/words**
- The number of buckets in the hash table is fixed at **16,384**.
- **test.cpp** provides you with the skeleton of a test case in which you will need to fill out the expected output. You can also add more tests for your hash function.

2. Requirements

1. Implement the **generateHashCodeOf()** function in **hasher.cpp**.
2. Write tests for your hash function in **test.cpp**.
3. After testing your hash function, compile your hasher (and link it) with the given object files to create an executable. We will talk about this process during the lab.
4. Run the executable.
    a. After starting the program, you can manually enter one English word per line. The program ends when an empty line is given. (Just hit the enter key without typing any characters at the line.)
    b. Or, you can use redirect (<) and specify the input file at when you start the program.
    c. With a successful run, the program prints out the length of chain at each bucket (0-based index), as well as your score at the end.
    d. The score is defined as the length of the longest chain in the hash table. Smaller is better.
5. Lastly, in your **README**,
    a. Write your score for the given English words dataset.
    b. Briefly explain the strategy you took in your hash function.

3. README

Create the README file that includes the following categories with appropriate section headers.

1. **Name**: Your name.
2. **Date**: The last updated date.
3. **Summary**: A brief summary of the lab.
4. **Files**: A list of files that are necessary to build and test the program.
5. **Instructions**: A sequence of commands to compile and test the program. Note that you are expected to report procedures without using the make command.
6. **References**: A list of citations to information used to complete the lab.
7. **Score**: See 5.a. in Section 2.
8. **Strategy**: See 5.b. in Section 2.

4. Submission

Submit your files listed below using Gradescope.
Files: **test.cpp hasher.cpp hasher.hpp README**