CPSC 2150 – Algorithms and Data Structures II

Lab8: Hash Functions

Total - 40 Marks

Learning Outcomes

- Design and develop different hash functions
- Analyzing the Hash functions
- Program with C++

Resources

- Chapter 9 of the text book
- Chapter 10 of the reference book
- en.wikipedia.org/wiki/Hash_function#Hash_function_algorithms.

Description

In this lab you are going to test various hash functions to see how good they are, in terms of number of collisions. To investigate this, we are not saving the values into the hash table, but we only use hash table to count the number of collisions on each index of hash table. Then we can decide whether a hash function is evenly distributed or not.

Your input will be strings, in fact, *all* the strings that are stored in a file named **keys.txt** and is uploaded into D2L. You can download a copy to your local computer for testing. This file contains just under 100,000 English words, with one word per line. We are going to use it to test the uniformity of various hash functions.

Your hash functions will hash strings into *16-bit* (not 32-bit) `int`s. In C++, `unsigned short` is a 16-bit unsigned integer.

This is important, because we're going to keep a table of the number of collisions for *each* hash value. With 16-bit `int`s there are only 65,536 possible hash values, so this table will easily fit in the memory. If we used 32-bit `int`s then there would be 4,294,967,296 possible hashes, a more troublesome amount.

Implementation

Create a class named Hash as following:

```
class Hash{
   public:
         Hash(the parameters like input file name or hash function's name){
               //develop the body
          };
          // add more methods if needed…
   private:
        const int SIZE = (int)pow(2, 16);
        vector <int> hashTable = vector <int>(SIZE);
          // add more methods or data field if needed….
}
```

As part of the class, implement the following hash functions:

a. **[10 marks] String length** (modulo 2^{16})
b. **[10 marks] First character**
c. **[10 marks] Additive** (add all characters together), modulo 2^{16}
d. **[10 marks] Mystery** (Apply your own idea to have less collisions).
e. **[10 bonus marks] Bonus**: Any other hash schemes that has an acceptable performance. I am looking for a hash function that is perfect or close to perfect hash function for these inputs. Your function must outperform the other functions (ie. equal or less than 10 as for the difference between maximum and minimum collisions). To get some ideas have a look at:
   en.wikipedia.org/wiki/Hash_function#Hash_function_algorithms.

For each of the possible hash functions your program should:

- Create a `vector<int> hashes` of size 65,536
- Process the list of words, and for each word, compute its hash *h*
- Increment the entry in the table for that hash: `hashes.at(h)++`
- When finished, find the *largest* and *smallest* entries in the vector, and print out the difference between them. This is our approximation for a measure of how evenly distributed the hashes are. (A better method would be to use Pearson's chi-squared test [en.wikipedia.org/wiki/Pearson's_chi-squared_test], but that requires numeric methods that are unfortunately not part of the C++ standard library.)

Sample output

Here is a sample output:

```
The difference between maximum and minimum collision on the entries of the hash
table using the following hash function are:

String length: 15669
First character: 9933
Additive: 280
Mystery: 10   //this should be some value more than 10 and less than 280
Bonus: 8
```

Your output does not have to look exactly like the sample, as long as you include the relevant information (the difference between maximum and minimum, printed after the name of the hash method).

Submit to D2L

Make a **zip file** named **StudentNumber-lab8.zip** including all related files by the end of the lab time**.** For example, if your student number is 10023449, the submitted file must be named as **10023449-lab8.zip**.