POTD45.1. Problem of the Day #45

Download and Extract

An initial setup of files is provided to you via a shell script: Download potd-q45

Using a terminal, extract the initial files by running the shell script you just downloaded (you will need to navigate to the directory where you saved the file):

```
sh potd-q45.sh
```

Your files for this problem will be in the `potd-q45` directory.

The Problem

An important part of implementing hashing is knowing how to handle collisions. Let's say we have a hash function to hash integers into our hash table. What happens when two integers hash to the same value?

Double hashing is a way to handle these collisions. You use one hash function to find the index of where the element should go. If there is already a value there, you use a second hash fucntion to find a step size to add to original index. You keep increasing the current index by step size until you find an empty space. In psuedo code:

```
find index from first hash function
if list[index] is empty:
   put the element there
while list[index] is not empty
   calculate a step from second hash function and add it to the
   index. This is the new index. Make sure it is in the bounds of the table.
```

Todos:

For this problem, our hash table will be stored in a vector.

Write a function `firstHash` that takes two integers: the element to be added,`elem`, and the length of the vector,`len`, and returns the index of where `elem` should go. The index is calculated by multiplying `elem` by four then finding the remainder of dividing that value by the length. Examples: `firstHash(5, 27);` returns 20 and `firstHash(27, 5);` returns 3.

Write a function `secondHash` that takes one int: the element to be added,`elem`, and returns a step size. The step size is calculated by finding the remainder of dividing `elem` by three, and then subtracting that result from three.
Examples: `secondHash(5);` returns 1 and `secondHash(27);` returns 3.

Write a void function `doubleHashInput` that takes in a vector v and the element you want to add to it. Empty spots in v will be represented by a value of -1. For example if you start with an empty vector `v` of size 5: `v[0]= -1 v[1]= -1 v[2]= -1 v[3]= -1 v[4]= -1` It will contain 5 values which are -1.

Implement the pseudocode listed above so that doubleHashInput uses firstHash to calculate the index and secondHash to calculate the step size.

Here's a visual representation:
https://www.cs.usfca.edu/~galles/JavascriptVisual/ClosedHash.html

POTD 45

Total points: 0/1

Score: 0%

Question

Value: 1

History:

Awarded points: 0/1

Report an error in this question

Previous question

Next question

Note: We understand that these hash functions are not good, at all. DO NOT use them in real life.

Upload Solution

Drop files here or click to upload.

Only the files listed below will be accepted—others will be ignored.

Files

- Hash.cpp
  not uploaded
- Hash.h
  not uploaded

Save & Grade | Save only