CS 2420-001 Algorithms and Data Structures

Spring Semester, 2014

Assignment 5: Hash Tables

Due Date: Wednesday, Mar. 19, 2014 (at the beginning of CS 2420 class)

(**Note:** This assignment has two programming exercises and one written exercise. For the programming exercises, please submit **ONLY** your source files to Canvas.)

1. In this exercise, we will implement the hash tables with the separate chaining technique for resolving collisions, as we discussed in class. **(30 points)**

   We use a hash table T to support the following *dictionary operations*.

   (a) $insert(x)$: insert a new key x to T. To achieve the constant time insertion, x should be put at the head of the linked list located by the hash function, as we discussed in class.

   (b) $remove(x)$: remove the key x from T.

   (c) $search(x)$: determine whether the key x is in T. If yes, return "true"; otherwise return "false".

   On Canvas, go to the following directory: homework/hw5/question1. There are a starter cpp file "hw5_Q1.cpp" and an input file "hw5_Q1_input.txt". The hash function $hash()$ has already been provided in the cpp file, which is $hash(x) = x\%\ m$ and m is the size of the hash table T. The first line of the input file is the size of the hash table. The program first reads that value and then defines a hash table with size equal to that value.

   Each line of the rest of the input file is "insert x", "remove x", or "search x". The program reads the input file line by line and perform the operations accordingly. After all these operations finish, the program will print out all keys of the hash table T. All the output is given both on the console (the screen) and an output file "hw5_Q1_output.txt".

   Your task is to complete the three functions $insert()$, $remove()$, and $search()$.

   To help you check whether your program runs correctly, I put a file "Wang_hw5_Q1_output.txt" in the same directory, which contains the correct output. For your convenience, all above files are included in a zip file.

2. In this exercise, we will implement the hash tables with the open addressing technique for resolving collisions. The probing method we are going to use the linear probing, as we discussed in class. **(30 points)**

   We use a hash table T to support the same three operations as in Question 1. The difference is that when a collision happens, we "probe" the next cell of T based on the linear probing. Here, each element of T is associated with a "flag", which can be "EMPTY", "ACTIVE", or "DELETED", as we discussed in class. I defined an "enum" type for it.

On Canvas, go to the following directory: homework/hw5/question2. There are a starter cpp file "hw5_Q2.cpp" and an input file "hw5_Q2_input.txt". The hash function $hash()$ and the probe function $probe()$ have already been provided in the cpp file. The first line of the input file is the size of the hash table. The program first reads that value and then defines a hash table with size equal to that value.

Each line of the rest of the input file is "insert x", "remove x", or "search x". The program reads the input file line by line and perform the operations accordingly. After all these operations finish, the program will print out the entire hash table T. All the output is given both on the console (the screen) and an output file "hw5_Q2_output.txt".

Your task is to complete the three functions $insert()$, $remove()$, and $search()$.

To help you check whether your program runs correctly, I put a file "Wang_hw5_Q2_output.txt" in the same directory, which contains the correct output. For your convenience, all above files are included in a zip file.

Remark. The probe function calls the function $linearProbe(x, i)$ to implement the linear probing. If we wish to use the quadratic probing or double hashing, we only need to change the probe function accordingly, but the three functions $insert()$, $remove()$, and $search()$ should not need to change. Of course, we need to choose a secondary hash function for the double hashing, as we discussed in class. For this reason, I do not need to give you any exercise on either quadratic probing or double hashing. But you can try it yourself by using the same code and providing the corresponding probe functions.

3. Using a hash table of size $m = 11$ with hash function $hash(x) = x \ \% \ m$, show the hash table that results after the following keys are inserted in the given order: 26 42 5 44 92 59 40 36 12. **(30 points)**

   For each of the following approaches, show the resulting hash table.

   (a) Linear probing, i.e., $h_i(x) = (hash(x) + i) \ \% \ m$, for $i = 0, 1, 2, \ldots$.

   (b) Quadratic probing, i.e., $h_i(x) = (hash(x) + i^2) \ \% \ m$, for $i = 0, 1, 2, \ldots$.

   (c) Double hashing using the secondary hash function $hash_2(x) = x \ \% \ 9 + 1$, i.e., $h_i(x) = (hash(x) + i \cdot hash_2(x)) \ \% \ m$, for $i = 0, 1, 2, \ldots$. Note that this secondary hash function does not follow our discussion in class, but theoretically we can pick any function as the secondary hash function.

   Note: You may submit your answers to this question through Canvas (e.g., submit a txt file) or write your answers on a paper and turn it in in classroom, although the Canvas option is preferred.

Total Points: 90