BLG335E, Analysis of Algorithms I, Fall 2015 Project 4

Handed out: 28.11.2015

Due: 11.12.2015, until 23.30

PURPOSE:
The purpose of this project is to develop a better understanding of hash tables. This is accomplished by writing a hash table class and experimenting with different hash functions.

Part A. Questions on Hash Tables (10 points)

1) Why do we use Hash Tables as a data structure in our problems? Please explain briefly. **(5 points)**

2) What is the main applications of Hash Tables? **(5 points)**

Part B. Implementation and Report (90 points)

1) Implementation (90 points)

Problem: A student affairs office wants to store the records of students in a hash table. In order to decide on a suitable hashing strategy, they want to analyze the performance of different hashing strategies. The office wants to operate three operations, insert, delete, and retrieve.

The following rules should be satisfied with your implementation:

- Each student is hashed with their students IDs.
- In the beginning of the execution, all the hash slots are empty.
- **insert (key)**
  Inserts **key** into the appropriate slot in the hash table. The location (**index**) in the hash table is determined by the key and the hash function.

- **delete (key)**
  Searches the hash table for the given **key** and delete it from the table. In order to delete a key from a hash table slot, you should just mark the corresponding slot as **deleted** (Lazy deletion). Remember that during insertion, slots marked as **deleted** are treated as empty.
- **retrieve (key)**
  Finds the given **key** in the hash table and return its **index**. If the key is not in the table, it should return **-1**.
- During searching a key, if a slot marked with **deleted** is encountered, the search continues. Therefore, deleted slots are treated as occupied during search.
- It is not allowed to have more than 50 deleted marked slot in the table. In such a situation, you should organize the hash table in such a way that there are no deleted marked slots.
- You should use corresponding hash functions during all the operations.

You are given a text file namely "dataset.txt" which has **2500 studentIds**. Every line of the file indicates an operation and the corresponding key values. For example, when "**insert: 714589763**" is encountered, you should add 714589763 key value to the table. When **"delete: 714589763"** is encountered, you should delete the corresponding key from the table.

a. (15 points) Implement an m-sized hash table that uses **linear probing** strategy in order to store student information by using the following hash function. The table should support insert, delete, and retrieve operations.

$$\mathrm{h(k,i)} = (\mathrm{k} + \mathrm{i}) \bmod \mathrm{m} \text{ and } \mathrm{i} \in [0, \mathrm{m}-1]$$

b. (20 points) Implement an m-sized hash table that uses **double hashing** strategy in order to store student information by using the following hash function. The table should support insert, delete, and retrieve operations.

$$\mathrm{h(k,i)} = \mathrm{h_1(k)} + \mathrm{i} * \mathrm{h_2(k)} \bmod \mathrm{m} \text{ and } \mathrm{i} \in [0, \mathrm{m}-1]$$

$$\mathrm{h_1(k)} = \mathrm{k} \bmod \mathrm{m} \quad \text{and} \quad \mathrm{h_2(k)} = 1 + \mathrm{k} \bmod (\mathrm{m}-1)$$

c. (20 points) Implement an m-sized hash table that uses **quadratic hashing** strategy in order to store student information by using the following hash function. The table should support insert, delete, and retrieve operations. You may choose $\mathbf{c_1 = 1, c_2 = 2.}$

$$h(k,i) = (h'(k) + c_1 i + c_2 i^2) \bmod m \;\text{ and }\; i \in [0, m-1]$$

$$h'(k) = k \bmod m$$

d. (15 points) Read the input file "dataset.txt" and apply the defined operations on the hash table.

e. (20 points) Number of collisions when inserting an item k is defined as follows:

Initialize collisions = 0.
Compute $h(k,i)$, if $h(k,i)$ is occupied, then increment collisions by 1.

During the execution, compute the number of collisions for each strategy and fill in the table with the number of collisions occurred for each strategy. Comment on the results.

Linear	Quadratic	Double	Universal

Detailed Instructions

- All your code must be written in C++ using **object oriented approach** and able to compile and run on Linux using **g++.**
- Do not use external libraries such as STL.
- Since this is an individual project, you are required to work **individually**.
- Submissions will be done through the Ninova server. You must submit all your program (.h and .cpp) and header files. You must also submit a softcopy report.

- In your report, explain your classes and methods briefly.
- During the execution, delete, search and update methods should give explanatory messages to the screen. At the end of the execution, total number of collisions for each strategy and for each m should be printed out. An example output can be seen below: (Note that the numbers in the output are imaginary).

```
......
Deletion: Key 637789932 is deleted from the table.
Retrieve: Key 256478523 is found at index 548.
Retrieve: Key 124579863 is not found in the table.
Insertion: Key 124578956 is inserted in index 478.
.......
Linear Probing: 300 collusions have been occurred
```

If you have any questions, please feel free to contact Res. Asst. Hakan GÜNDÜZ via e-mail (hakangunduz@itu.edu.tr).