Seoul National University

Data Structure

Fall 2017, Kang

Programming Assignment 5: Searching (Chapter 9)

Due: Dec. 12, 14:00, submit at eTL

Reminders

- The points of this homework add up to 100.
- Like all homeworks, this has to be done individually.
- Lead TA: Hyunsik Jeon (jeon185@gmail.com)
- Write a program in Java.
- Do not use Java Collection Framework

1. How to submit the programming assignment

1) Create a ***JAR*** file including 'src' folder that contains your sources files, but without 'release' folder. (Refer to '1 – Introduction.pptx' in the first lab session)
   - We will run your ***Main*** class in the JAR file to grade your programming assignments. Before submitting the JAR file, please check if your Main class in the JAR file works correctly in a terminal.
   - You **MUST** obey the I/O specification of the programming assignment, and rules for the submission of the programming assignment.

2) Compress the JAR file with a readme file and the project folder. If you don't use Eclipse, include source files into the compressed file. The readme file needs to include your student id, name, how to execute your program, and any other information TA needs to know.
   - The format of the JAR file is "*PA_##_(StudentID).jar*" where '##' is the programming assignment ID, and *(StudentID)* is your student ID.
     - ex) PA_01_2017-12345.jar
   - All documents should be written in English.

3) The name of the compressed file (zip file) should be '*PA_##_(StudentID).zip*' where '##' is the programming assignment ID, and (StudentID) is your student ID.
   - ex) PA_01_2017-12345.zip
     - ➢ *01: the first programming assignment*
     - ➢ *2015-12345: your student ID*

4) Submit the compress file to the eTL (http://etl.snu.ac.kr/) .

2. How to grade your programming assignment

1) We made a grading machine to automatically grade your programming assignment. The machine will run your program, and compare answers and outputs that your program generates for given inputs. If your program cannot generate correct answers for an input file, it will not give you the point corresponding to the input. Our machine will consider the following scenarios:

   - (***Accept***) When your program generates exact outputs for an input file, the machine will give you the point of the input.
   - (***Wrong Answer***) When your program runs normally, but generates incorrect outputs for an input file, including typos, the machine will not give you the point of the input.
   - (***Run Error***) When your program does not run, or is terminated suddenly for some reasons, the machine will not give you the point of an input file because it cannot generate any outputs.
   - (***Time Limit***) When your program runs over a predefined execution time for an input file, our machine will stop your program, and it will not give you the point of the input. The time limit of the execution is ***5 seconds***.

2) We will generate 10 input files, and assign 10 points for each input file. For example, if your program gets 9 accepts, and 1 wrong answer by the machine, the total point will be 90 points. <u>Hence, before submitting your programming assignment, please be sure that your program makes correct answers in reasonable time for any input case.</u>

3. Problem

In this assignment, you have to implement a Hash Table with quadratic probing for collision resolution. The task is specified as follows.

3.1. Hashing.

The hash table has 1009 slots and the hash function is defined as $h(k) = k \; mod \; 1009$.

3.2. Collision resolution policy.

The probe function is defined as $P(k, i) = c_1 i^2 + c_2 i + c_3$ for some choice of constants c_1, c_2, and c_3. Then the i th value in the probe sequence would be $(h(k) + P(k, i)) \; mod \; 1009$. For example, let's assume that a probe function is $P(k, i) = i^2$. If $h(k_1) = 10$, the probe sequence for k_1 is 10, 11, 14, 19, and so forth. c_1, c_2, and c_3 are given when you create a hash table.

3.3. Deletion.

Lets assume that $h(k_1) = h(k_2) = h(k_3) = index$, and k_1 was inserted first, then k_2, then k_3. After that, you deleted k_1 from the hash table setting it back to $null$. When later you will search for k_2 or k_3, you will find that $h(k_2) = h(k_3) = index$ and $table[index] = null$. Therefore you will get a wrong answer: k_2 or k_3 is not in the table. To solve this problem, you need to set $table[index]$ with a special marker when a key is deleted from the table. In this assignment, you have to set the special marker as '-1'.

The specific operations you need to implement are as follows:

- Create: You have to create a new hash table that has 1009 slots. Also the quadratic probing collision resolution policy has to be defined with given parameters.
- Insert: You have to insert the key into the hash table using the hash function and the collision resolution policy. Also you have to print the index (the position of the key inserted in the hash table). The duplicated keys are not allowed for insertion.

- Delete: You have to delete the key from the hash table. If there is no such key in the table, the message "Failed to find key" has to be printed and nothing happens to the table. You have to set the special marker as '-1' when you delete the key.

- Search: You have to find the index of the key in the hash table. If there is no such key in the table, the message "Failed to find key" has to be printed.

Here are several assumptions for clarify.

- All keys are positive integers.

- The hash table has 1009 slots, and the hash function is $h(k) = k \; mod \; 1009$.

- The probe function is defined as $P(k, i) = c_1 i^2 + c_2 i + c_3$ and c_1, c_2, and c_3 are given as parameters.

- All given integers are distinct.

- The special marker for deletion is -1.

You have to fill in "HashTable.java" and "Main.java" of "PA05" java project.

4. ADT of Data structure

1) Create

Function
void Create(int c1, int c2, int c3)

Description
- This function creates a new hash table with quadratic probing for collision resolution. - (c1), (c2), and (c3) are at least zero. - (c1)==0, (c2)==0, and (c3)==0 should not be allowed. - This function is called only once at the beginning.

2) Insert

Function
void Insert(Integer key)

Description
- This function inserts (key) into the hash table according to the defined collision resolution policy. - The duplicated keys are not allowed for insertion.

3) Delete

Function
void Delete(Integer key)

Description
- This function deletes (key) from the hash table. - If there is no such (key) in the table, the message “Failed to find (key)” has to be printed and nothing happens to the table. - The special marker for deletion is set to -1.

4) Search

Function

void Search(Integer key)
Description
- This function finds the index of the (key). - If there is no such (key) in the table, the message “Failed to find (key)” has to be printed.

5. Specification of I/O

1) create

Input form	Output form
`create (c1) (c2) (c3)`	`(No output)`

Description

- (c1), (c2), and (c3) are constants for the collision resolution policy.
- There is no output for this input.
- “create” is given only once at the beginning.

Example Input	Example Output
`create 3 7 0`	

2) insert

Input form	Output form
`insert (key)`	`INSERT: (key) / INDEX: (index)`

Description

- (key) is a positive integer.
- (index) is an index of the table where the (key) is inserted.

Example Input	Example Output
`insert 1009`	`INSERT: 1009 / INDEX: 0`

3) delete

Input form	Output form
`delete (key)`	`DELETE: (key) / INDEX: (index)`

Description

- (key) is a positive integer.
- (index) is an index of the table where the (key) is deleted.
- If there is no such (key) in the table, the message “Failed to find (key)” has to be printed and nothing happens to the table.

Example Input	Example Output
`delete 1009`	`DELETE: 1009 / INDEX: 0`
`delete 1009`	`Failed to find 1009.`

4) search

Input form	Output form
`search (key)`	`SEARCH: (key) / INDEX: (index)`

Description

- (key) is a positive integer.
- (index) is an index of the table where the (key) was found.
- If there is no such (key) in the table, the message “Failed to find (key)” has to be printed.

Example Input	Example Output
`search 3`	`SEARCH: 3 / INDEX: 3`
`search 4`	`Failed to find 4.`

6. Sample Input and Output

The grading machine expects the sample output for given the sample input. Hence, for the sample input, your program should print the same lines in the sample output. If your program prints different lines from the sample output for the sample input, the grading machine will not give you the point of the input. See more details of the policy on the grading machine in Section 2.

1) Sample Input

```
create 1 0 0

insert 1

insert 1010

insert 2019

insert 3028

search 1

search 1010

search 2019

search 3028

delete 1010

search 1

search 1010

search 2019

search 3028
```

2) Sample output

```
INSERT: 1 / INDEX: 1
INSERT: 1010 / INDEX: 2
INSERT: 2019 / INDEX: 5
INSERT: 3028 / INDEX: 10
SEARCH: 1 / INDEX: 1
SEARCH: 1010 / INDEX: 2
SEARCH: 2019 / INDEX: 5
SEARCH: 3028 / INDEX: 10
DELETE: 1010 / INDEX: 2
SEARCH: 1 / INDEX: 1
Failed to find 1010
SEARCH: 2019 / INDEX: 5
SEARCH: 3028 / INDEX: 10
```