Seoul National University

Data Structure

Fall 2015, Kang

Programming Assignment 5 : Searching (Chapter 9)

Due: Dec. 10, 11:00 am, submit at eTL

Reminders

- The points of this homework add up to 100.
- Like all homeworks, this has to be done individually.
- Lead TA: Minsoo Jung (qtyp456987@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.
   - You **MUST** obey the I/O specification of the programming assignment, and rules for the submission of the programming assignment.
   - Before submitting, check if your JAR file runs properly in your terminal with the following command line: java -classpath ./PA_05_2015-12345.jar ds.test.Main.

2) Compress **the JAR file, a readme file and the project folder.** 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_05_2015-12345.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_05_2015-12345.zip
     - *05: the fifth 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 can not 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 can not 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. Hence, before submitting your programming assignment, please be sure that your program makes correct answers in reasonable time for any input case.

3. Problem

In this assignment, you should implement a “HashTable” with 1001 slots using either linear probing or quadratic probing. The hash function is $\mathrm{h(k)} = \mathrm{k} \bmod 1001$.

First, you will be given a collision resolution policy which is either linear probing or quadratic probing. Then, given distinct integers, you should insert the integers in the hash table using the hash function $\mathrm{h(k)}$. The specific operations you need to implement are as follows:

- Start: You need to create a new hash table with a given collision resolution policy. The collision resolution policy is either linear probing or quadratic probing. ‘Start’ is called only once at the beginning of each input file, and the collision resolution policy is given as a parameter.
- Insert: You need to insert an integer into the hash table using the hash function $\mathrm{h(k)}$ and the collision resolution policy.
- Find: You need to find the index of a given integer in the hash table. If there is no such integer in the hash table, you should print the message “Cannot Find”.

Here are several assumptions for clarity.

- The hash table has 1001 slots and the hash function is $\mathrm{h(k)} = \mathrm{k} \bmod 1001$
- All given integers are *distinct*. The maximum number of integers is 1001.
- The range of an integer is between 0 and $2^{31} - 1$.
- The constants of linear and quadratic probing are positive integers.

Complete “HashTable” class supporting the above requirements. You need to fill in “HashTable.java” and “Main.java” of “PA_05” java project. You can modify the skeleton codes, or add classes if you want.

4. ADT of Data structure

1) Start

Function
void start(String policy, int constant1, int constant2, int constant3)
Description
- This function creates a new hash table with a collision resolution policy. - If (policy) is “linear”, then the collision resolution policy is linear probing. (constant1) is a constant $\mathbf{c}$ for the linear probe function. In this case, (constant2) and (constant3) are 0. - If (policy) is “quadratic”, then the collision resolution policy is quadratic probing. (constant1), (constant2), and (constant3) are $\boldsymbol{c_1}$, $\boldsymbol{c_2}$, and $\boldsymbol{c_3}$ respectively for the quadratic probe function. - This function is called only once at the beginning of each input file.

2) Insert

Function
void insert(int value)
Description
- This function inserts (value) into the hash table. - All given values are distinct. - The range of (value) is between 0 and $\mathbf{2^{31} - 1}$.

3) Find

Function
int find(int value)
Description
- This function finds the table index where (value) is contained in the hash table. - If the function finds the index successfully, then return the table index. - If the hash table does not contain the (value), then print a message “Cannot Find”.

5. Specification of I/O

1) start

Input form	Output form
`start (policy) (constant1) (constant2) (constant3)`	

Description

- (policy) is either ‘linear’ or ‘quadratic’.
- (constant1), (constant2), and (contant3) are constants for the collision resolution policy. If (policy) is ‘linear’, then (constant2) and (constant3) are zeros.
- ‘start’ is given only once at the beginning of each input file.
- Note that there is NO output for this input.

Example Input	Example Output
`start linear 1 0 0`	
`start quadratic 1 2 3`	

2) insert

Input form	Output form
`insert (value)`	`INSERT: (value)`

Description

- (value) is an integer to be inserted into the hash table.

Example Input	Example Output
`insert 1010`	`INSERT: 1010`

3) find

Input form	Output form
`find (value)`	`FIND: (index)`

Description

- (value) is an integer to be found from the table.

- (index) is the index of (value) in the hash table.
- If the table does not contain (value), then print “Cannot Find”. (Note that you should NOT print a dot in the message.)

Example Input	Example Output
`find 1010`	`FIND: 9`
	`Cannot Find`

6. Sample Input

```
start linear 1 0 0

insert 1003

insert 9050

insert 8049

insert 2877

insert 3000

find 9050

find 8049

find 2000
```

7. Sample Output

```
INSERT: 1003

INSERT: 9050

INSERT: 8049

INSERT: 2877

INSERT: 3000

FIND: 41

FIND: 42

Cannot Find
```