Seoul National University

Data Structure

Fall 2019, Kang

Programming Assignment 4: Searching (Chapter 9)

Due: December 2nd, 23:59, submit at eTL

Reminders

- The points of this homework add up to 100.
- Like all homework, this has to be done individually.
- Lead T.A.: Chaeheum Park (chaeheum@snu.ac.kr)
- Write a program in Java (version 11.0.4).
- Do not use Java Collection Framework or any third-party implementation from the Internet.

Important

- Use Java 11.0.4, as this is the version used for evaluation. Run error will be given 0 points.

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 commands:

     "`java -classpath PA_04_(studentID).jar ds.test.Main`".
   - `(studentID)` must be in the following format:

     `2019-12345`

2) Submit the jar file to the eTL (http://etl.snu.ac.kr/).

2. How to grade your programming assignment

1) We made a grading script for the 'program execution' part. The script 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 script 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 5 points for each input file. For example, if your program gets 9 accepts, and 1 wrong answer by the machine, the points for 'program execution' part will be 45 points. Hence, before submitting your programming assignment, please be sure that your program generates correct answers in reasonable time for any input cases.

3. Problem

In this assignment, you have to implement a Hash Table with quadratic probing for collision resolution. The tasks are specified as the following.

a. Hashing.

The hash table has 577 slots and the hash function is defined as $h(k) = k \ mod \ 577$.

b. Collision resolution policy.

The probe function is defined as $P(k,i) = c_1 i^2 + c_2 i + c_3$ for constant c_1, c_2, and c_3. Then the i th value in the probe sequence would be $(h(k) + P(k,i)) \ mod \ 577$. 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. The collision resolution policy is applied regardless of collision($i = 0 \ when \ no \ collision$).

c. 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 $null$. Later when you 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 an exception: 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 operations you need to implement are the following:

- Create: Create a new hash table that has 577 slots. Also the quadratic probing collision resolution policy has to be defined with given parameters.
- Insert: 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). Here duplicated keys are not allowed for insertion.
- Delete: Delete the key from the hash table. If no such key exist 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: Find the index of the key in the hash table. If there no such key exist in the table, the message "Failed to find key" has to be printed.
- Print All: Print all key, index pairs in the hash table. This will be called always after at least one insertion.

Here are some assumptions for clarification.

- All keys are positive integers.
- The hash table has 577 slots, and the hash function is $h(k) = k \ mod \ 577$.
- 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.
- When calling "Print All" the output line should not contain whitespace at the end.

You have to fill in "HashTable.java" and "Main.java" of "PA04" java project.

4. Specification

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 is not be allowed. - This function is called only once at the beginning.

2) Insert

Function
`void insert(int key)`
Description
- This function inserts (key) into the hash table according to the defined collision resolution policy. - Duplicated keys are not allowed for insertion.

3) Delete

Function
`void delete(int key)`
Description
- This function deletes (key) from the hash table. - If there is no such (key) exists 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 -1.

4) Search

Function

```
void search(int key)
```

Description

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

5) Print All

Function

```
void printAll()
```

Description

- This function prints all key, index pairs in the hash table.
- This function will always be called after at least one insertion.

5. I/O Specification

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 577`	`INSERT: 577 / 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 577`	`DELETE: 577 / INDEX: 0`
`delete 577`	`Failed to find 577`

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 1`	`SEARCH: 1 / INDEX: 1`
`search 4`	`Failed to find 4`

5) Print All

Input Form	Output Form
`printAll ()`	`key(index), key(index), ...`

Description

- This function will always be called after at least one insertion.
- Deleted position must be printed with the format -1(index).
- Output should not contain any whitespace at the end (except for new line).

Example Input	Example Output
`print`	`6453(107), 132(133), 321(322), 435(436)`
`print`	`-1(107), 132(133), 321(322), 435(436)`

6. Sample Input and Output

The grading script expects the sample output for the given sample input. Hence, for the sample input, your program should print the same lines in the sample output shown below. If your program prints differently from the sample output the grading script will mark as wrong for the respective input line.

Sample Input	**Sample Output**
create 1 1 1 insert 132 insert 321 insert 435 insert 6453 search 132 search 321 search 435 search 6453 print delete 6453 search 132 search 321 search 435 search 6453 print	INSERT: 132 / INDEX: 133 INSERT: 321 / INDEX: 322 INSERT: 435 / INDEX: 436 INSERT: 6453 / INDEX: 107 SEARCH: 132 / INDEX: 133 SEARCH: 321 / INDEX: 322 SEARCH: 435 / INDEX: 436 SEARCH: 6453 / INDEX: 107 6453(107), 132(133), 321(322), 435(436) DELETE: 6453 / INDEX: 107 SEARCH: 132 / INDEX: 133 SEARCH: 321 / INDEX: 322 SEARCH: 435 / INDEX: 436 Failed to find 6453 -1(107), 132(133), 321(322), 435(436)