CSCI 2110 Data Structures and Algorithms
Laboratory No. 9
Week of November 27 – December 1, 2023

Due: Sunday, December 3, 11.59 PM

Hashing and Hash Tables

The objective of this lab is an experimental study of hash tables with open hashing. Download the example code/files provided along with this lab document. You will need the following files to complete your work:

- HashTableExperiment.java (A starter program for the lab exercise)
- Node.java (Generic Node Class)
- LinkedList.java (Generic LinkedList class)

Marking Scheme

There is one exercise in this lab. Your final score will be scaled down to a value out of 10.
Error checking: Unless otherwise specified, you may assume that the user enters the correct data types and the correct number of input entries, that is, you need not check for errors on input

Submission Requirements:

- No submission other than a single ZIP file will be accepted.
- You MUST SUBMIT .java files that are readable by the TA. If you submit files that are unreadable such as .class file, the lab will be marked 0.
- Please additionally comment out package specifiers.

Late Submission Penalty: The lab is due on Sunday at 11.59 PM. Late submissions up to 5 hours (4.59 AM on Monday) will be accepted without penalty. After that, there will be a 10% late penalty per day on the mark obtained. For example, if you submit the lab on Monday at 12 noon and your score is 8/10, it will be reduced to 7.2/10. Submissions past two days (48 hours) after the grace submission time, that is, submission past 4.59 AM Wednesday will not be accepted.

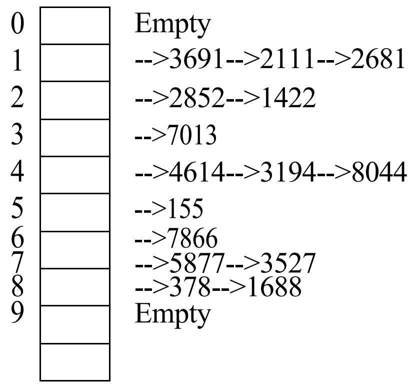
Review of hashing (from the lectures): A hash table is a simple array structure. Whenever a key needs to be mapped onto the hash table, it uses a hash function. A common hash function used is

$$key \ \% \ (table\_size)$$

For example, if the table_size is 10 and keys are 1008, 2540, 3467 and 789, they get mapped to locations 8, 0, 7 and 9, respectively.

However, multiple keys may map to the same location. For instance, in the above example, 3467 and 2487, will both map to location no. 7. A hash collision is said to occur. One way of resolving a hash clash is called separate chaining. Rather than storing the keys in the array, each array location contains a linked list. The keys are stored in the linked list.

The following is an example in which 15 keys 3527, 7013, 2681, 7866, 8044, 1688, 5877, 1422, 3194, 4614, 2852, 155, 2111, 3691, and 378 are mapped onto a table of size 10:



0	Empty
1	-->3691-->2111-->2681
2	-->2852-->1422
3	-->7013
4	-->4614-->3194-->8044
5	-->155
6	-->7866
7	-->5877-->3527
8	-->378-->1688
9	Empty

In the above example, the number of times a collision has occurred is 7. The longest linked list has a size of 3. We say that the hash table has a load factor of 15/10 = 1.5 (Load factor = number of keys mapped/table size. Note that this is not integer division).

Exercise

This is an experimental exercise on hashing. Your task is to modify the program called HashTableExperiment.java. It creates a hash table of various sizes and various numbers of keys. Then it collects statistics on the load factor, number of collisions and the longest list length. Steps are listed below.

1. Prompt the user to enter the table_size.
2. Declare an arraylist of linked lists of that size. Assume that the linked list stores integer objects.
3. Prompt the user to enter the number of keys to be hashed. Generate that many ***random*** keys in the range 1 to 10000. You need to generate non-duplicate keys. You can store the keys in another array or arraylist.
4. For each key generated, determine where it should be mapped (pos = key%table_size).
5. Go to the arraylist index pos and add the key into the linked list in that position.
6. After all the keys have been added, determine the number of collisions and the longest list length.
7. Run the program a number of times, experimenting with different number of keys (10,100, 500, 1000) and different table sizes (10, 20, 30). Create a table showing your experimental values.

Notes:

- A starter program HashTableExperiment.java has been given to you. Download this as well as LinkedList.java and Node.java (the generic LinkedList and Node classes). This program first creates an arraylist of linkedlist of integers, displays the empty linked list, then hashes a few keys, and displays them again.
- You are NOT allowed to make any changes to LinkedList and Node classes.
- Keys are added to the front of the linked list (not to the end), and that is perfectly OK.
- Run HashTableExperiment.java. If you enter the size of the table to be 10, you should see the following output:

```
Enter the hash table size: 10
Empty lists
0: null
1: null
2: null
3: null
4: null
5: null
6: null
7: null
8: null
9: null

After the keys are hashed
0: 120--> --> null
1: --> null
2: --> null
3: --> null
4: --> null
5: 205--> 105--> --> null
6: --> null
7: 187--> --> null
8: --> null
9: 189--> --> null
```

Note that the starter program has the five keys hard coded. In your exercise, you will not be hard coding any keys. You need to make the appropriate changes to the program to complete the exercise. Record your results in a table like the one given below:

Table size	Number of keys	Load factor	Number of collisions	Longest list length
10	10			
10	100			
10	500			
10	1000			
20	10			
20	100			
20	500			
20	1000			
30	10			
30	100			
30	500			
30	1000			

- The Load Factor is the Number of keys divided by the Table size.
- The number of collisions is equal to the number of additional keys mapped to the same linked list. For example, in the sample run shown below, there are five keys (6490, 9493, 635, 2595 and 4655) which experience collisions.

```
Enter the size of the hash table: 10
Enter the number of keys to be hashed: 10

Hash Table created:
0: 8180-->6490-->null
1: null
2: null
3: 9563-->9493 -->null
4: 324 -->null
5:  2375-->635-->2595-->4655 --> null
6: null
7: null
8: 2258 -->null
9: null

Statistics:

Table size: 10
Number of keys: 10
Load factor: 1

Number of collisions: 5
Longest list: 4
```

Copy and paste the console interaction for ONE sample run of the program into a text or word document.
Now run the program for different numbers of keys and table sizes, collect the statistics, and enter the values in the table. Add the table to the text or word document.

What to submit:
Submit one ZIP file containing all source code (files with .java suffixes) and the text/word/similar document containing sample output.
Your final submission should include the following files: HashTableExperiment.java, Node.java, LinkedList.java and the document containing the sample program run and the table of experimental values.

Note 1: Although you may not have made any changes to Node.java and LinkedList.java, please include it in your submission package so that the TAs can execute your code easily.

Note 2: Please ensure that your full name and Banner ID appears on the document containing sample program run and the table.