Dr. Phil Maguire

Maynooth University
National University of Ireland Maynooth

CS211FZ ALGORITHMS & DATA STRUCTURES II

LAB 6: **HASHING**

Objectives

- Understand how hash functions work.
- Use different approaches to address the hash collision problem.
- Reflect on the knowledge learned in the class.

NOTE:

- Do NOT use "package" in your source code.
- You must submit the source code files, i.e., the ".java" files.
- You are allowed to use course reference books or class notes during the lab.
- Sharing/copying your work is NOT permitted.

Pen and Paper Exercise

Insert the values (253, 444, 669, 880, and 707) into the following hash table using the following algorithms. The double hash function returns a value between 1 and 15.

i Linear Probing

ii Quadratic Probing

iii Double Hashing

0	457	12	543	24	777	36	744	48	
1	526	13	714	25		37	446	49	344
2		14		26		38		50	
3		15		27		39		51	280
4	181	16		28		40	453	52	
5		17	194	29	796	41		53	584
6	360	18	485	30		42	455	54	703
7		19		31	857	43	980	55	
8		20	492	32	681	44	870	56	
9		21	375	33	210	45		57	
10	482	22	961	34		46		58	294
11	478	23		35	94	47	519		

Task: Programming Exercise

To practice this question ahead of the lab, you can use the code below and load it in the file dictionary.txt

Problem statement

The goal is to insert all the words into the hash table and then find them all again with the minimum number of collisions. You can use any hashing strategy you like. The main method that reads in the data is provided. There is also a HashTable class - you must use the check() method to check for a collision when finding a word. A counter ticks up every time you make an unsuccessful check() call (i.e., a collision). You have to keep this as low as possible. You should just complete the fill() and find() methods. When you run the code, it will output the number of collisions you have. If you see the word "error!" it means that your find() method is not finding the words properly in the hash table.

Input Format

The input is introduced (dictionary.txt). Please check if its address (path) is set correctly. All you need to do is complete the fill() method, which fills the words into the hash table array, and the find() method, which should return the slot in the hash table where the word is to be found in the solution class.

Output Format

There are two test cases.

- The first just checks that everything is working - there are few items inserted, so you should get zero collisions, and if you get zero collisions, then the first test case is passed. You can try 10 inputs (int items=10 in line7), i.e., read the first 10 words from the dictionary.txt file. Set the table size 1009 (size = 1009 in HashTable class), the first prime number larger than 1000.
- The second test case will push the load factor to 90%. 90,000 inputs and table size 100,003, which is the first prime number larger than 100,000. You should minimize the collisions (anything under 400,000 is decent).

This lab will be graded by how low you can get the number of collisions.