Lab 9

Objectives:

- Sign up for the lab
- To understand hashing concepts using simple examples
- To implement a hash table (hash function, collision resolution)
- Use Connex to submit one files: MyHashTable.*java* at the end of the lab

Why use a hash table?

For quick access. For a balanced binary search tree, the "search" method takes O(log n), the same data, if stored in a hash table with a good hash function, it takes constant time, in big O notation, it is O(1).

What is a hash table?

According to Wikipedia, it "*is a data structure used to implement an associative array, a structure that can map keys to values. A hash table uses a hash function to compute an index into an array of buckets or slots, from which the correct value can be found.*" For example, we created a Student data type in lab 2. The student number is a key which is unique. Each year, there are more than ten thousand students registered in UVic. If the student objects are stored in an array, if we can compute the index of the array from the student number, we can retrieved a student in constant time O(1).

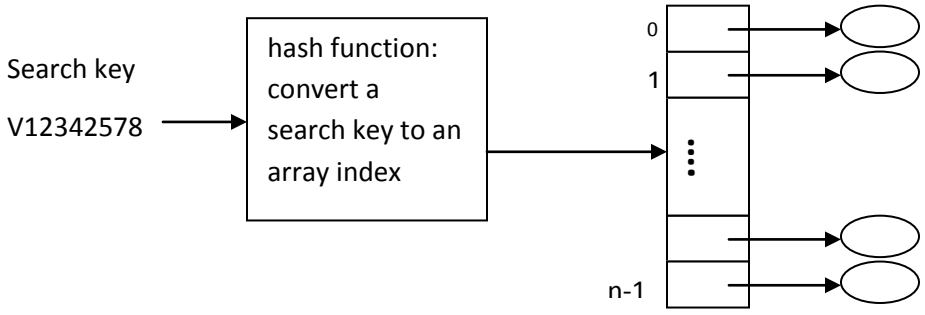

Fig. 9.1 A Hash Table Stores Student Objects

What is a hash function?

A hash function is a way to map a search key to the array index. A search key could be a number or an object, in our Student example, it is a String object. We need a good hash function to distribute the student objects evenly into the array.

Collision Resolution

Collision means that two different student objects mapped into the same array cell (same index). Even a good hash function is not a guarantee that there are no collisions. Therefore, collision resolution is required to implement a hash table. We are going to use **Separate Chaining** in this lab.

1. Take out a piece of paper, do some modulo arithmetic. What are the reminders of the following arithmetic operations?

1 ÷ 7 the remainder is _____ in java format: 1 % 7
8 ÷ 7 the remainder is _____ in java format: 1 % 7
3 ÷ 7 the remainder is _____ in java format: 1 % 7
16 ÷ 7 the remainder is _____ in java format: 1 % 7

2. Now, put the numbers 1, 8, 3, 16 into a hash table with size 7, using separate chaining, draw a diagram on a piece of paper.

3. Design a hash function for our student number string object:

Hint: convert each character to an integer (ASCII code maybe), add the integers, or use the Horner's rule (page 743 of Data Abstraction & Problem Solving with JAVA 3rd Edition by J. Richard and F. Carrano), mod the table size (choose a prime number as the table size).

4. Download the files and implement MyHashTable.java, write a main method to test it.

Task: implement MyHashTable.java. Test it thoroughly.

Use Connex to submit one file: MyHashTable.java at the end of your lab.