# Implementing a HashMap
Report #10

By
Eduardo Castro

CS 303 Algorithms and Data Structures

November 5, 2014

## 1. Problem Specification

The goal of this assignment is to implement a HashMap table and implement linear and quadratic probings functions.

## 2. Program Design

The first part of the assignment was basically to create a HashMap, insert some elements on it and check if it is right.

The following steps were required to develop this program:

a) write the HashEntry class
b) write the HashMap class
c)use the driver class to display and measure the results

The following methods were defined within the RBNode:

a) HashMap()
    a. Constructor
b) getValuePosition(int position)
    a. returns the value into a given position
c) get(int key)
    a. returns the value into a given key
d) put(int key, String value)
    a. inserts a new element into a given key
e) HashEntry(int key, String value)
    a. creates the node for the hash entry that will be inserted at the hash table
f) getKey()
    a. returns the key
g) getValue()
    a. returns a value
h) setValue(String val)
    a. sets a value

Then we had to implement a linear probing and quadratic probing methods inside the HashMap class

a) linearProbe(int key, String value)
    a. implements the linear probe
b) QuadraticProbe(int key, String value)
    a. implements the quadratic probe

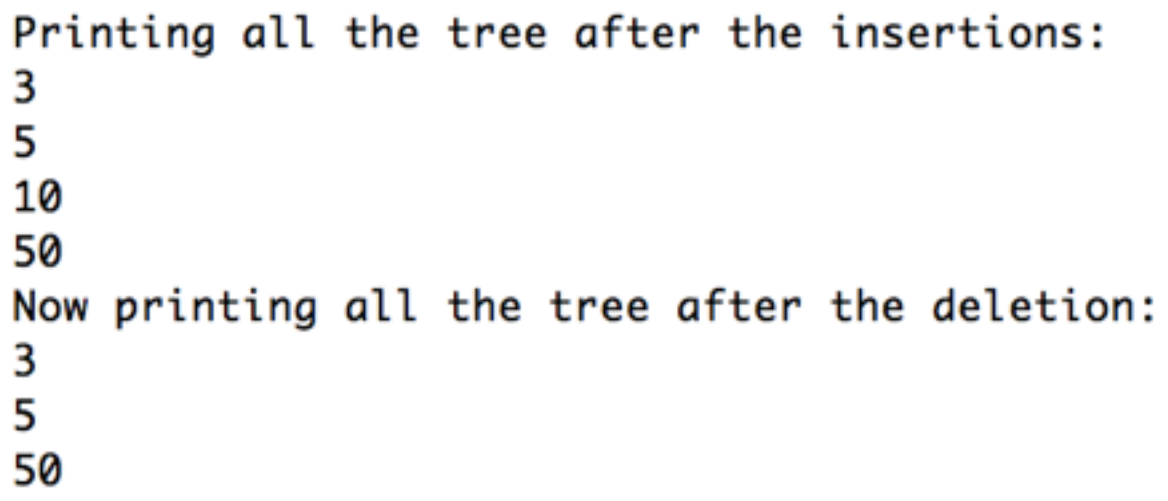
The Scanner class provided by Java was used to read in the necessary values within the provided driver program. The println method of the System.out object was used to display the inputs and results for the provided driver program.

## 3. Testing Plan

Insert some elements and prints the tree to check if the tree is right.

## 4. Test Cases

The tests case for the first part with some inserting and searchings:



```
Printing all the tree after the insertions:
3
5
10
50
Now printing all the tree after the deletion:
3
5
50
```

## 5. Analysis and Conclusions

The HashTable it is still a little bit confusing about the way that it works but it is an interesting data structure. It is notable that it is an easy way to avoid conflicts while inserting a lot of elements into a data structure but to understand how it works it is not an easy task. The cost for inserting and searching elements at the average case is pretty great, since it is O(1), but for the worst case is O(n), which is not that good.

I was not capable of testing the last part since I had some problems catching data from the .csv file so I was not capable of testing it but I measured the times for inserting and showed it at the test cases.

## 6. References

The parameters used was from the homework assignment provided in class, I also used as reference for learning how to work with compareTo the Java documentation and another references from an Algorithms course from Princeton.