**CS303 Lab 10 – Red-Black Tree**

**Problem Specification:**

The problem given to us is to create the code for a red-black tree and run it against the given input and data files. We had to specifically test out two different functions. The first was the inorder traversal and the second being the searching function.

**Program Design:**

I have five classes: BSTNode.java, BST.java, RBNode.java, RBTree.java, and Driver.java The driver class just reads in all the files and creates nodes for the RBTree file to use. The RBTree file has five functions: RBInsert(), RBInsertFix(), LeftRotate(), RightRotate(), inOrderTreeWalk(), and search(). The RBTree() function essentially creates the tree by appending a node to the previous node starting from the root each time. The first node to be added to the tree instantly becomes the root. The RBInsertFix function fixes any issues that come from insertion by rotating left and right. It uses conditions to check to see if a left or right rotate is necessary and then calls either LeftRotate() or RightRotate(). The inOrderTreeWalk function starts at the root of the function and goes down the left and right side of each node and prints out the key. Finally, the search function goes through the entire tree and finds every element that is listed within the input.dat or KEYs.dat file. After finding the element, it prints out the time it took to find it and the key and description of the node. The final class I have created for this lab is the RBNode class, which is essentially a class that holds the data for anything that is deemed to be a RBNode object. It holds the key and description of each node as well as its left, right, and parent node information. It also holds a Boolean variable called color that is false when red and true when black.

**Testing Plan:**

In order to implement this, I created a system that allows the user to choose which file they would like to create a binary search tree out of. Then I run the inOrderTreeWalk and print out all the keys of the nodes within the tree. For this lab, I commented out the line that prints out the key because the UPC-random.txt file has so many keys that it takes a while to print them all out. Afterwards, I ask the user if which input file they would like to use. The choices are the input.dat or the KEYs.csv. After calling the search on each of the keys within either file the program prints out how long it took to search for that key and prints out the key and its description.

**Test Cases:**

**Text

Description automatically generatedText

Description automatically generated**

**Text

Description automatically generatedText

Description automatically generated**

**Text

Description automatically generated**

**Analysis/Conclusion:**

We have two different results to look at, the first being the red-black tree search total time and the HashMap put, linear put, and quadratic put searching times. At first glance, the red-black tree takes a lot more time to search, almost 4 times as long. This is excluding whether the key itself was found and if the description was printed. When comparing the actual results of the search, the red-black tree is able to find the key and description for every key within the input.dat file, while the HashMap search finds very few to none of the keys that input.dat holds. So technically, the red-black tree is better because we are able to find all of the keys from the input.dat file even if the time taken is longer we get better searching results.

**References**:

The only references I must make are to the pseudocode and my previous lab in which I used a similar format for writing this report. Below I have added images of all my code for this lab. I am only adding the RBTree, RBNode, and Driver code because the rest is almost identical to the BST code from lab 8.

Text

Description automatically generatedText

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generatedText

Description automatically generated

A screenshot of a computer

Description automatically generated with medium confidence

Text

Description automatically generatedText

Description automatically generatedText

Description automatically generated