CIS 313, Intermediate Data Structures
Winter 2019

CIS 313 Lab 4

Due: March 15th, 2019 at 4:59pm

For this lab, your task is to implement a Red-Black Tree.

Overview

You will Construct a Red-Black Tree of numbers. You will extend your code in programming assignment 2 to now include a balancing operations. For your convenience, some of the BST operations have already beed written for you in the skeleton code. You are also given a working traverse method and getNode method. You simply need to extend the functionality to support balanced insert and delete operations.

A BST is a Red-Black tree if it satisfies the following Red-Black Properties:

1. Every node is either red or black
2. Every leaf node counts as black
3. If a node is red, then both its children are black
4. Every simple path from a node to a descendant leaf contains the same number of black nodes
5. The root node is always black

Fill out all of the methods in the provided skeleton code.
You may add additional methods, but should NOT add public fields. You also should not change the name of any of the classes or files.
The program should continually read instructions from a file.

In particular, you will implement the following functionality for your Red-Black Tree:

insert
Preform BST insert
Check to see if the tree breaks any of the Red-Black Properties
If so, preform the appropriate rotations and or re-colorings

delete
Preform BST delete
Check to see if the tree breaks any of the Red-Black Properties
If so, preform the appropriate rotations and or re-colorings

search
Return the key for a node corresponding to the given number

Print "Found" if the corresponding key is in the tree.
Otherwise, print "Not Found"

leftRotate
If x is the root of the tree to rotate with left child subtree T1 and right child y, where T2 and T3 are the left and right children of y:

- x becomes left child of y and T3 as its right child of y
- T1 becomes left child of x and T2 becomes right child of x

rightRotate
If y is the root of the tree to rotate with right child subtree T3 and left child x, where T1 and T2 are the left and right children of x:

- y becomes right child of x and T1 as its left child of x
- T2 becomes left child of y and T3 becomes right child of y

Additionally, implement the following instructions in the main() function in lab4.py

Input Description

The input will be a text file, for example *inSample.txt* below will be provided. Each of the lines will contain a different set of words specifying a task along with a number (if applicable). You should create an empty Red-Black Tree (with root null), and then perform a sequence of actions on that tree.

Note: You should implement your Red-Black tree with generics, but create a Red-Black tree that takes in integers.

```
insert 10
insert 5
insert 20
insert 3
traverse
insert 2
traverse
search 17
insert 1
delete 20
traverse
```

Output Description

Using the sample input above, your program should output:

```
10 5 3 20
10 3 2 5 20
Not Found
3 2 1 10 5
```

Testing Protocol

We will test your program in the same fashion as previous labs. We strongly suggest you test your program in the following ways:

- While creating it
- With inSample.txt
- With any corner-cases you can think of

Grading

This assignment will be graded as follows:

Correctness 40%
Your program runs without syntax errors: 10%
Your program runs without runtime errors: 10%
Your program produces the correct output: 20%

Implementation 40%
Insert, delete, search, traverse, and exit all preform in the correct time complexity for the RBT: 10%
leftRotate, and rightRotate are also preform in the correct time complexity: 30%

Documentation 20%
To earn points for implementation, your code must be clear enough for us to understand.
** This should be the easiest 20 points **