Gebze Technical University
Computer Engineering

CSE 222 - 2018 Spring

HOMEWORK 6 REPORT

BURAK DEMİRCİ
141044091

Course Assistant: Fatma Nur Esirci

1 Worst RedBlack Tree

1.1 Problem Solution Approach

Bu bölümün kodları Ders kitabından alınmıştır. Sadece Test amaçlı kod eklenmiştir.
Test kodu:
6 yüksekliğinde RedBlack Tree için 14 elaman artan sıra ile yerleştirilerek worst case durumu elde edilmeye çalışıldı.

1.2 Test Cases

İlk durumda Integer alan bir RedBlack oluşturuldu sırasıyla 1 den 14 e kadar sayılar Tree ye eklenerek test edildi İkinci durumda ise String alan bir RedBlack oluşturuldu “0” dan “13”e kadar olan sayıların Stringleri yerleştirilerek test edildi.

1.3 Running Commands and Results.

RedBlackTree.java

```
package Q1;

public class MainQ1 {

    public static void main(String[]a){

        RedBlackTree<Integer> redBlackTreeInt = new RedBlackTree<>();
        RedBlackTree<String> redBlackTreeStr = new RedBlackTree<>();

        System.out.println("*******         TEST Q1       ******");
        System.out.println("****************************************");
        System.out.println("Worst RedBlack Tree with height of 6.");
        System.out.println("\n    Try Integer ");
        System.out.println("redBlackTreeInt.add(1)  ->"+redBlackTreeInt.add(1));
        System.out.println("redBlackTreeInt.add(2)  ->"+redBlackTreeInt.add(2));
        System.out.println("redBlackTreeInt.add(3)  ->"+redBlackTreeInt.add(3));
        System.out.println("redBlackTreeInt.add(4)  ->"+redBlackTreeInt.add(4));
        System.out.println("redBlackTreeInt.add(5)  ->"+redBlackTreeInt.add(5));
        System.out.println("redBlackTreeInt.add(6)  ->"+redBlackTreeInt.add(6));
        System.out.println("redBlackTreeInt.add(7)  ->"+redBlackTreeInt.add(7));
        System.out.println("redBlackTreeInt.add(8)  ->"+redBlackTreeInt.add(8));
        System.out.println("redBlackTreeInt.add(9)  ->"+redBlackTreeInt.add(9));
        System.out.println("redBlackTreeInt.add(10) ->"+redBlackTreeInt.add(10));
        System.out.println("redBlackTreeInt.add(11) ->"+redBlackTreeInt.add(11));
        System.out.println("redBlackTreeInt.add(12) ->"+redBlackTreeInt.add(12));
        System.out.println("redBlackTreeInt.add(13) ->"+redBlackTreeInt.add(13));
        System.out.println("redBlackTreeInt.add(14) ->"+redBlackTreeInt.add(14));

        System.out.println("\n    Try String ");
        System.out.println("redBlackTreeStr.add(\"0\")  ->"+redBlackTreeStr.add("0"));
        System.out.println("redBlackTreeStr.add(\"1\")  ->"+redBlackTreeStr.add("1"));
        System.out.println("redBlackTreeStr.add(\"2\")  ->"+redBlackTreeStr.add("2"));
        System.out.println("redBlackTreeStr.add(\"3\")  ->"+redBlackTreeStr.add("3"));
        System.out.println("redBlackTreeStr.add(\"4\")  ->"+redBlackTreeStr.add("4"));
        System.out.println("redBlackTreeStr.add(\"5\")  ->"+redBlackTreeStr.add("5"));
        System.out.println("redBlackTreeStr.add(\"6\")  ->"+redBlackTreeStr.add("6"));
        System.out.println("redBlackTreeStr.add(\"7\")  ->"+redBlackTreeStr.add("7"));
        System.out.println("redBlackTreeStr.add(\"8\")  ->"+redBlackTreeStr.add("8"));
        System.out.println("redBlackTreeStr.add(\"9\")  ->"+redBlackTreeStr.add("9"));
        System.out.println("redBlackTreeStr.add(\"10\") ->"+redBlackTreeStr.add("10"));
        System.out.println("redBlackTreeStr.add(\"11\") ->"+redBlackTreeStr.add("11"));
        System.out.println("redBlackTreeStr.add(\"12\") ->"+redBlackTreeStr.add("12"));
        System.out.println("redBlackTreeStr.add(\"13\") ->"+redBlackTreeStr.add("13"));



    }


}
```

Run: MainQ1

```
"C:\Program Files\Java\jdk1.8.0_144\bin\java.exe" ...
*******         TEST Q1       ******
****************************************
Worst RedBlack Tree with height of 6.

    Try Integer
redBlackTreeInt.add(1)  ->true
redBlackTreeInt.add(2)  ->true
redBlackTreeInt.add(3)  ->true
redBlackTreeInt.add(4)  ->true
redBlackTreeInt.add(5)  ->true
redBlackTreeInt.add(6)  ->true
redBlackTreeInt.add(7)  ->true
redBlackTreeInt.add(8)  ->true
redBlackTreeInt.add(9)  ->true
redBlackTreeInt.add(10) ->true
redBlackTreeInt.add(11) ->true
redBlackTreeInt.add(12) ->true
redBlackTreeInt.add(13) ->true
redBlackTreeInt.add(14) ->true

    Try String
redBlackTreeStr.add("0")  ->true
redBlackTreeStr.add("1")  ->true
redBlackTreeStr.add("2")  ->true
redBlackTreeStr.add("3")  ->true
redBlackTreeStr.add("4")  ->true
redBlackTreeStr.add("5")  ->true
redBlackTreeStr.add("6")  ->true
redBlackTreeStr.add("7")  ->true
redBlackTreeStr.add("8")  ->true
redBlackTreeStr.add("9")  ->true
redBlackTreeStr.add("10") ->true
redBlackTreeStr.add("11") ->true
redBlackTreeStr.add("12") ->true
redBlackTreeStr.add("13") ->true

Process finished with exit code 0
```

2 binarySearch method

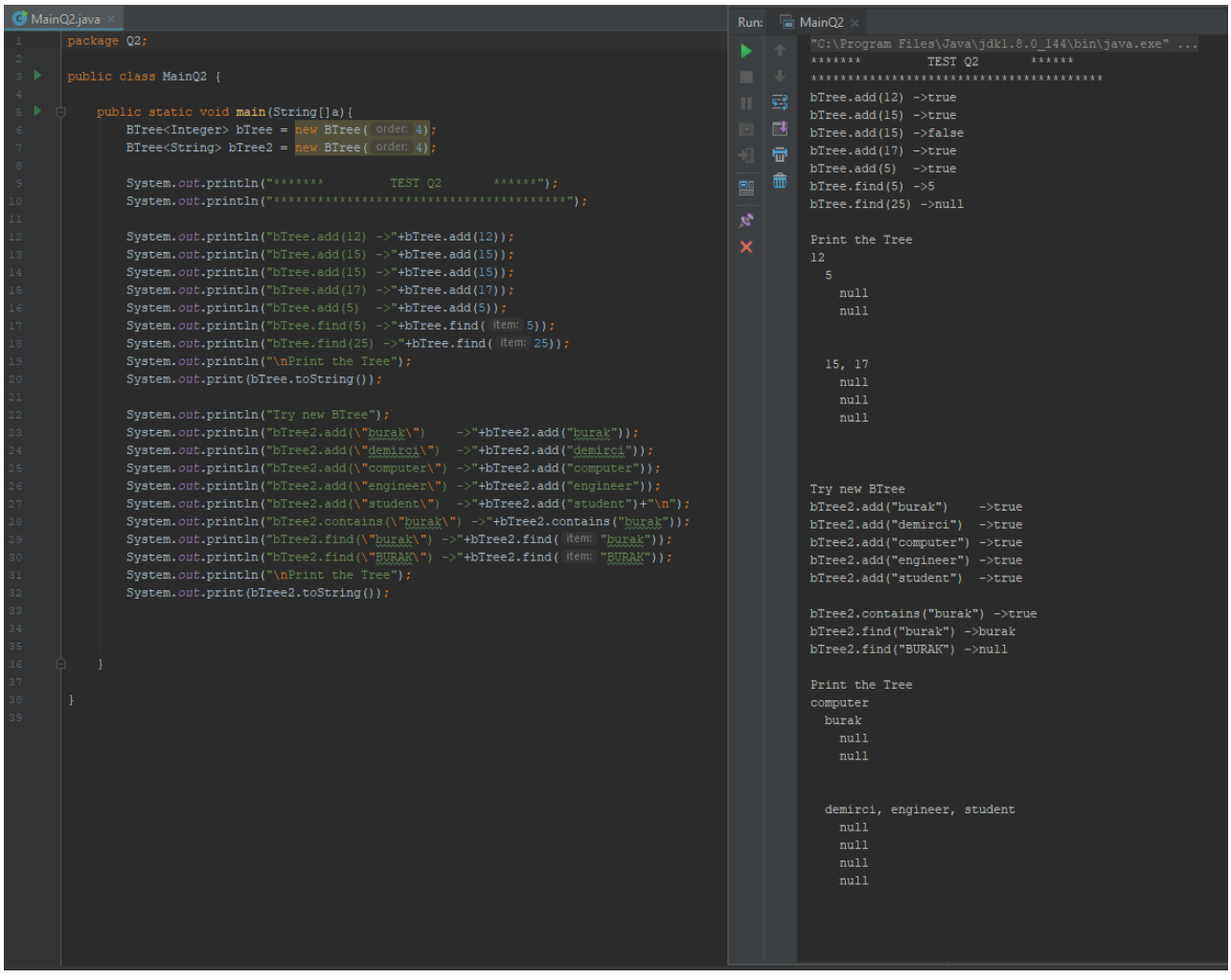
2.1 Problem Solution Approach

Bu bölümün kodlarının bir kısmı Ders Kitabından alınmıştır.
binarySearch item, root, left, right
if item compareTo root[left] is smaller than right
and item compareTo root[left] is smaller than 0
increase left
recursive call binarySearch item root , left, right
else
return left

2.2 Test Cases

Bu bölümde ilk test Integer alan bir BTree üzerinde yapılmıştır bazı sayılar eklenerek, eklenmiş sayıları tekrar eklemeye çalışarak ve BTree üzerinde find işlemi yapılarak son olarak bütün BTree'yi ekrana bastırarak işlem tamamlanmıştır. İkinci testte ise String şeklinde oluşturulan BTree üzerinde benzer uygulamalar yapılmıştır.

2.3 Running Commands and Results



3 Project 9.5 in book

Bu bölün kodlarının bir bölümü Ders Kitabından alınmıştır.

3.1 Problem Solution Approach

AVLB = AVLNODE BALANCED

IncrementBalance node
++node.balance
If node balance is bigger than AVLB
        increase assign true
else
        increase assign false

rebalanceRight (localRoot)
increase assign false
rightChild assign localroot get right
if rightChild is smaller than AVLB
       rightChildLeft assign rightChild get left
       if rightChildLeft smaller than AVLB
         rightChild balance assign RigtHeavy
         rightChidLeft balance assign AVLB
         localroot balance assign LeftHeavy
       else if rightChildLeft bigger than AVLB
         rightChild balance assign AVLB
         rightChidLeft balance assign AVLB
         localroot balance assign LeftHeavy
       else // equalty
         rightChild balance assign AVLB
         rightChidLeft balance assign AVLB
         localroot balance assign AVLB
       endof else
       localroot get right assign call rotateRight rightChild
       return call rotateLeft localRoot
else
        rightChild balance assign AVLB
        localroot balance assign AVLB
        return call rotateLeft localRoot

AVLTree (BinaryTree binTree)
If binTree is null
        set flag true
else
        set flag call CheckIsAvl binTree
if flag is true
        write “The Given Binary Tree is an AVLTree”
else
        write “The Given Binary Tree is NOT an AVLTree”

CheckIsAvl BinaryTree binTree
If call isBalance binTree is true
        return true
else
        return false

isBalance BinaryTree root
if root is null
        return true
leftHeigh assign call heigh root getLeftSubTree
rightHeigh assign heigh root getRightSubTree
if mutlak(leftHeigh - rightHeigh) is smaller with1 and
        call isBalance root get LeftSubTree and
        call isBalance root get RightSubTree
        return true
return false

heigh BinartTree root
if root is null
       return 0
else
        return 1 + call maxim root get LeftSubTree , root get RightSubTree

maxim value1 value2
return maximum of valu1 , value2

3.2 Test Cases

Bu bölümün testleri aşağıdaki gibidir.
Test1:
Integer olarak defauld constructorla oluşturulan AVLTree Integer elemanlar eklenmiştir öncelikle eklenen bazı elamanlar yeniden eklenmeye çalışılarak eklenip eklenmediği kontroledilmiştir. Eklenen öğelerin gerçekten eklenilip eklenilmediği test edilmiştir.

Test 2:
Bu bölümde elimle bir Binary tree oluşturdum ilk önce dengeli olacak şekilde oluşturup AVLTree nin constructoruna vererek Test ettim test başarı ile sonuçlandı .
Devamında aynı binaryTree ye dengeyi bozacak şekilde yeni elemanlar ekledim ve yinme AVLTree nın constructoruna vererek AVLTree olup olmadığını kontrol ettim AVLTree yi desteklemeyen yapı başarı ile AVLTree olmadığı saqptanmıştır.

3.3 Running Commands and Results

MainQ3.java

```
package Q3;

public class MainQ3 {

    public static void main(String[]a) {

        AVLTree<Integer> avlTreeInt = new AVLTree<>();

        System.out.println("*******         TEST Q3       ******");
        System.out.println("***************************************");
        System.out.println("    Test Integer");
        System.out.println("avlTreeInt.add(1) ->"+avlTreeInt.add(1));
        System.out.println("avlTreeInt.add(2) ->"+avlTreeInt.add(2));
        System.out.println("avlTreeInt.add(3) ->"+avlTreeInt.add(3));
        System.out.println("avlTreeInt.add(4) ->"+avlTreeInt.add(4));
        System.out.println("avlTreeInt.add(5) ->"+avlTreeInt.add(5));
        System.out.println("avlTreeInt.add(6) ->"+avlTreeInt.add(6));
        System.out.println("avlTreeInt.add(6) ->"+avlTreeInt.add(6));

        System.out.println("\n   Test Binary Tree is an AVLTree ? ");

        System.out.println("BinaryTree<Integer> binaryTree = new BinaryTree()\n");

        BinaryTree<Integer> binaryTree = new BinaryTree();

        System.out.println("binaryTree.root = new BinaryTree.Node<Integer>(new Integer(12))");
        binaryTree.root = new BinaryTree.Node<Integer>(new Integer( value: 12));

        System.out.println("binaryTree.root.right = new BinaryTree.Node<Integer>(new Integer(15))");
        binaryTree.root.right = new BinaryTree.Node<Integer>(new Integer( value: 15));


        System.out.println("binaryTree.root.left = new BinaryTree.Node<Integer>(new Integer(10))");
        binaryTree.root.left = new BinaryTree.Node<Integer>(new Integer( value: 10));

        System.out.println("\nTest 1");

        AVLTree<Integer> binCheckAvl = new AVLTree<>(binaryTree);

        System.out.println();

        System.out.println("binaryTree.root.left.left = new BinaryTree.Node<Integer>(new Integer(9))");
        binaryTree.root.left.left = new BinaryTree.Node<Integer>(new Integer( value: 9));

        System.out.println("binaryTree.root.left.left.left = new BinaryTree.Node<Integer>(new Integer(7)");
        binaryTree.root.left.left.left = new BinaryTree.Node<Integer>(new Integer( value: 7));

        System.out.println();
        System.out.println("\nTest 2");
        binCheckAvl = new AVLTree<>(binaryTree);
    }
}
```

Run: MainQ3

```
"C:\Program Files\Java\jdk1.8.0_144\bin\java.exe" ...
*******         TEST Q3       ******
***************************************
    Test Integer
avlTreeInt.add(1) ->true
avlTreeInt.add(2) ->true
avlTreeInt.add(3) ->true
avlTreeInt.add(4) ->true
avlTreeInt.add(5) ->true
avlTreeInt.add(6) ->true
avlTreeInt.add(6) ->false

   Test Binary Tree is an AVLTree ?
BinaryTree<Integer> binaryTree = new BinaryTree()

binaryTree.root = new BinaryTree.Node<Integer>(new Integer(12))
binaryTree.root.right = new BinaryTree.Node<Integer>(new Integer(15))
binaryTree.root.left = new BinaryTree.Node<Integer>(new Integer(10))

Test 1
The Given Binary Tree is an AVLTree

binaryTree.root.left.left = new BinaryTree.Node<Integer>(new Integer(9))
binaryTree.root.left.left.left = new BinaryTree.Node<Integer>(new Integer(7)


Test 2
The Given Binary Tree is NOT an AVLTree

Process finished with exit code 0
```