Gebze Technical University
Computer Engineering

CSE 222 - 2018 Spring

HOMEWORK 6 REPORT

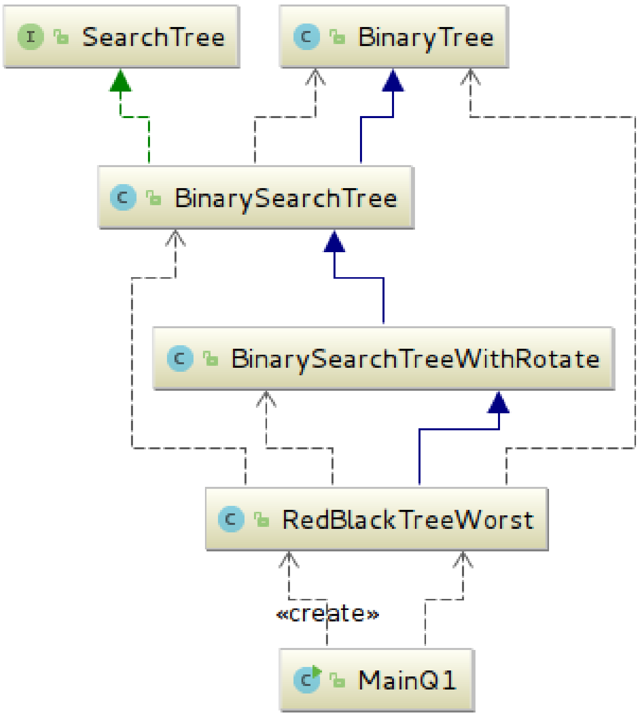
SİNAN ELVEREN
111044074

Course Assistant: Fatma Nur Esirci

1 Worst RedBlack Tree

We need maximum rotation and recoloring while insert elements for worst Red&Black Tree. For that, we need to insert minimum 14 element up to level 6 of tree. Tree can rotate maximum 2 times for one insert (3 rotate max for deletion). If we want to cause 2 rotations, we need to add element to parant.right.left.right (or the opposite). So, this is worst senario while add element. I created a sequence for provide this. Tree will recoloring in a second step, and tree will double rotating in a second step(first step recoloring, second step double rotating – generally up to level 6)
Worst case always O (log n) asymptoticly but in real, we can analysis rotate and re color count. This execuitons are being constant time(tetha 1). We can think rotate and recolor count for worst case.

1.1 Problem Solution Approach



No need to pseudocode,becouse of I didn't implement any method.

For worst case R&B Tree,

Need to sorted array, and
add first element
add last element
add 2 elements more from begin,
then add all from last-1 to 3.element

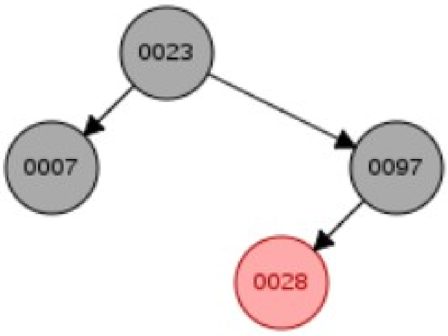
Ex. we have array that:
1 2 3 4 5 6 7 8 9 10 11 12 13 14
so, we will add to tree in order..
1 14 2 3 **13 12 .. 9 8 7 6 5 4**

Always double rotate in a second step.
Minimum element,
Maximum level – rotate - *recoloring

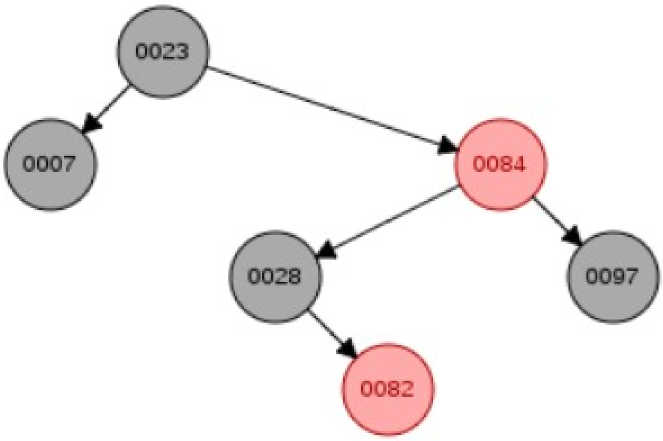
1.2 Test Cases

GetRandomSortedSeq() : 7 23 28 31 41 42 50 61 67 71 74 82 84 97
Total elements count is : 14

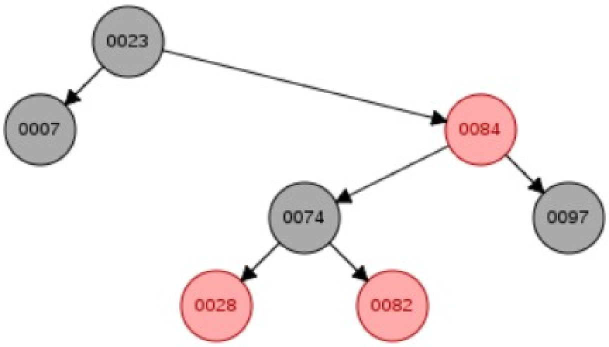
```
redblack.add(last);
redblack.add(first);

redblack.add(arr[2]); 2 rotate +recolor
redblack.add(arr[3]); 3 recolor
```



```
from .. redblack.add(last – 1) 2 rotate + recolor

        redblack.add(last – 2) recolors..

     .

     .
```

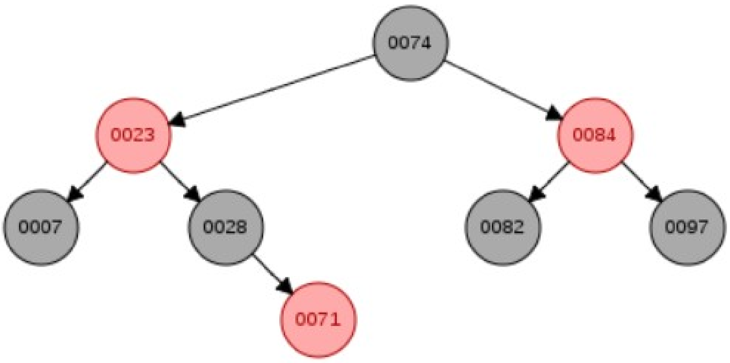


add: 74

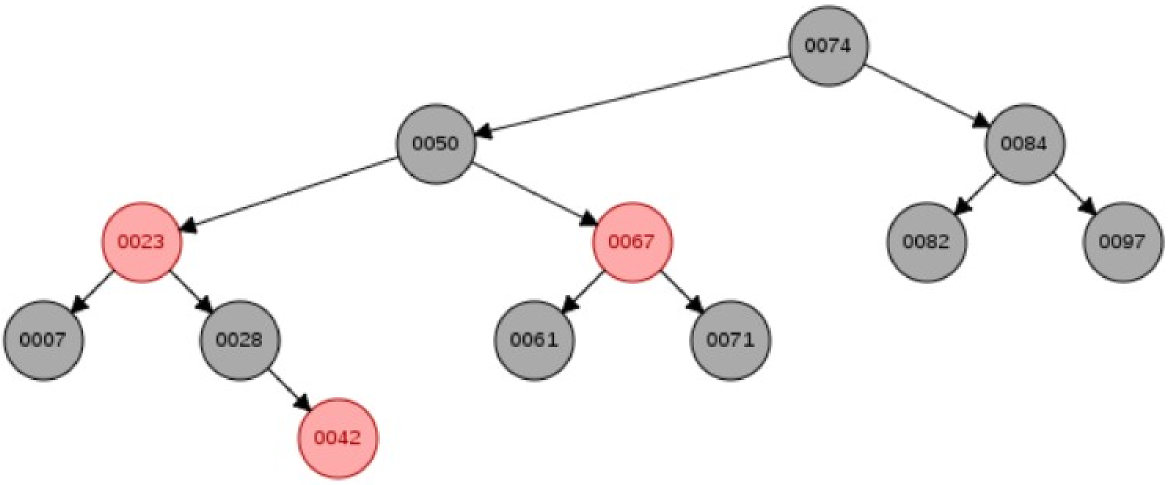


->

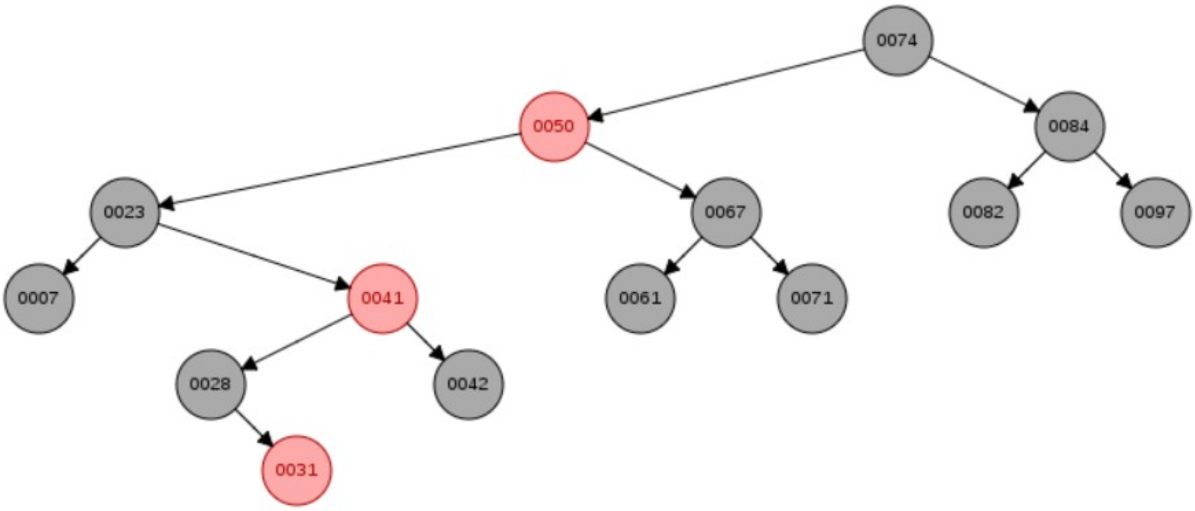
add: 71



add: 67, 61, 50, 42

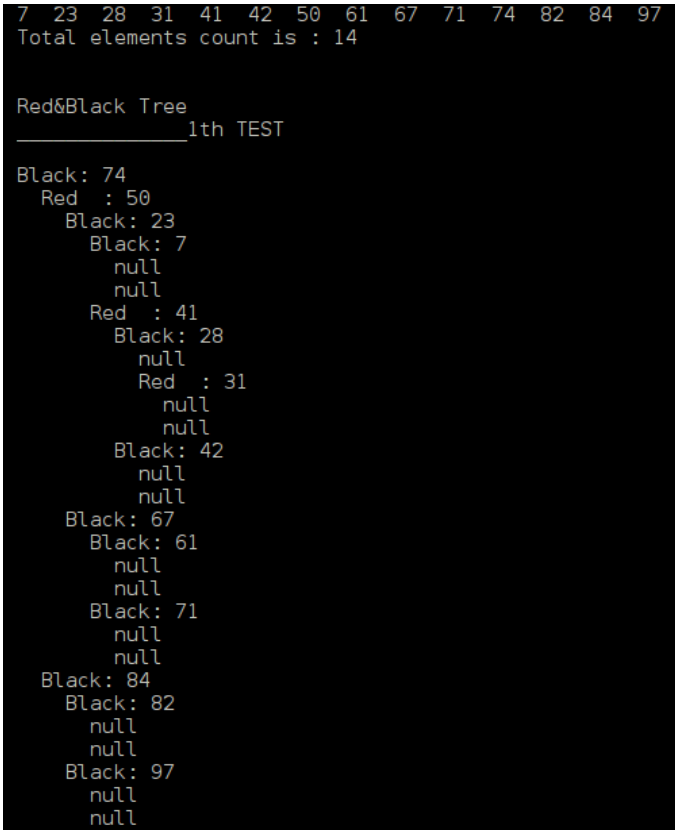


```
                                add: 41, 31

to ..   redblack.add(arr[4])  --  now 6 level.
```



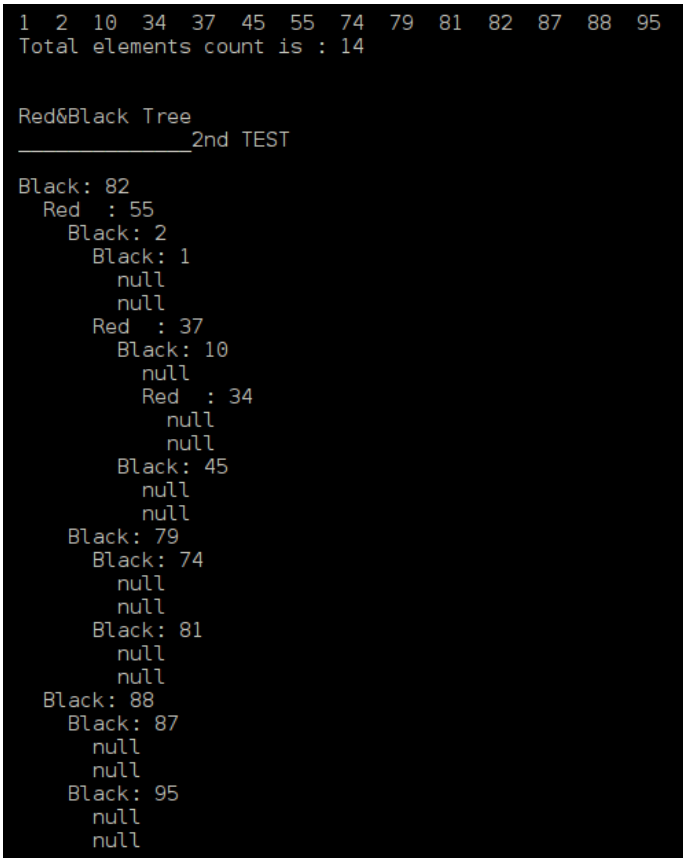
1.3 Running Commands and Results

Same array:



```
7  23  28  31  41  42  50  61  67  71  74  82  84  97
Total elements count is : 14


Red&Black Tree
______________1th TEST

Black: 74
  Red  : 50
    Black: 23
      Black: 7
        null
        null
      Red  : 41
        Black: 28
          null
          Red  : 31
            null
            null
        Black: 42
          null
          null
    Black: 67
      Black: 61
        null
        null
      Black: 71
        null
        null
  Black: 84
    Black: 82
      null
      null
    Black: 97
      null
      null
```

Second test:



```
1  2  10  34  37  45  55  74  79  81  82  87  88  95
Total elements count is : 14


Red&Black Tree
______________2nd TEST

Black: 82
  Red  : 55
    Black: 2
      Black: 1
        null
        null
      Red  : 37
        Black: 10
          null
          Red  : 34
            null
            null
        Black: 45
          null
          null
    Black: 79
      Black: 74
        null
        null
      Black: 81
        null
        null
  Black: 88
    Black: 87
      null
      null
    Black: 95
      null
      null
```

2 binarySearch method

For binarySearch(E[] items, E target, int first, int last) method firstly, searching array in node.
İt is checking 2 elements, not one for find the (target)index. First check is :
array[middle - 1] < target < array[middle - 1]
Second check is :
array[middle] > target > array[middle + 1]
It will find index recursively and return it which target between two indexes

2.1 Problem Solution Approach

We need to recursively method(both node.data and child.data in found node)for find (target)index which between two index (in array in node).

```
binarySearch(node.arr, target, first, last)
    if first greater last then
        return -1        // Base case for unsuccessful search.
    Else
        middle ← (first+last) / 2
        if  (target smaller arr[middle]) and (target greater arr[middle-1])
            return middle
        else
            binarySearch(node.arr, target, first , middle-1)
        if  (target greater arr[middle]) and (target smaller arr[middle+1])
                retrun middle + 1;
        else
                binarySearch(node.arr, target, first , middle +1)

    return middle            // element is already exist
```

2.2 Test Cases

Degree 4

________________________add 12

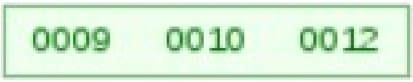
```
12
  |null
  |null
```

________________________add 10

```
10, 12
  |null
  |null
  |null
```

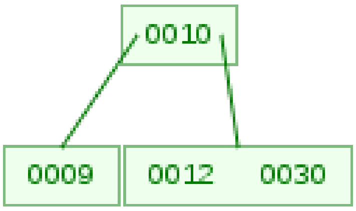
________________________add 9

```
9, 10, 12
  |null
  |null
  |null
  |null
```



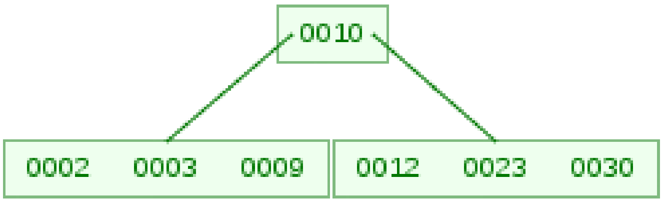
need to split_____________add 30

```
10
  |9
  |  |null
  |  |null

  |12, 30
  |  |null
  |  |null
  |  |null
```

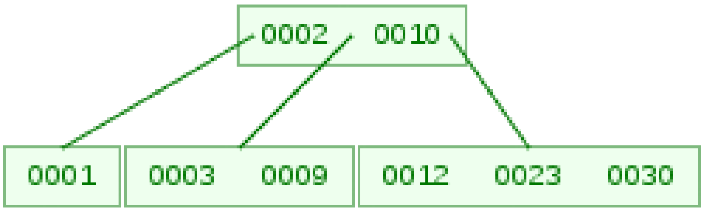


________________________add   23 - 3 - 2

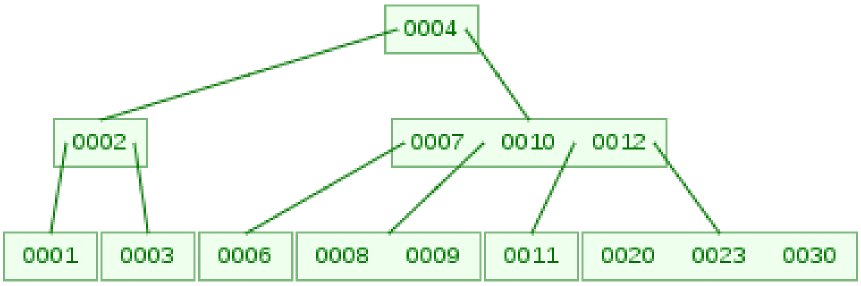
```
10
  |2, 3, 9
  |  |null
  |  |null
  |  |null
  |  |null


  |12, 23, 30
  |  |null
  |  |null
  |  |null
  |  |null
```



```
________________________add 1
2, 10
 |1
 |  |null
 |  |null

 |3, 9
 |  |null
 |  |null
 |  |null

 |12, 23, 30
 |  |null
 |  |null
 |  |null
 |  |null
```

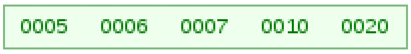


```
________________________add 11  4   20  6   7  8
4
 |2
 |  |1
 |  |  |null
 |  |  |null


 |  |3
 |  |  |null
 |  |  |null


 |7, 10, 12
 |  |6
 |  |  |null
 |  |  |null


 |  |8, 9
 |  |  |null
 |  |  |null
 |  |  |null


 |  |11
 |  |  |null
 |  |  |null


 |  |20, 23, 30
 |  |  |null
 |  |  |null
 |  |  |null
 |  |  |null
```



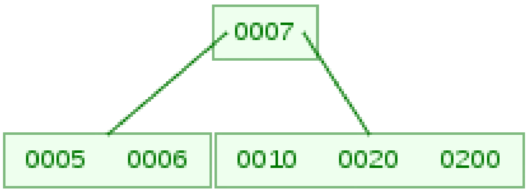
Degree 6

```
__________________________add 5  6  7  10  20
5, 6, 7, 10, 20
  |null
  |null
  |null
  |null
  |null
  |null
```

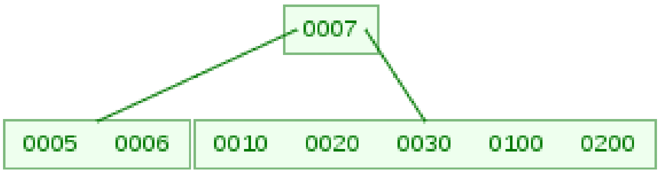


split

```
__________________________add 200
7
  |5, 6
  |  |null
  |  |null
  |  |null


  |10, 20, 200
  |  |null
  |  |null
  |  |null
  |  |null
```

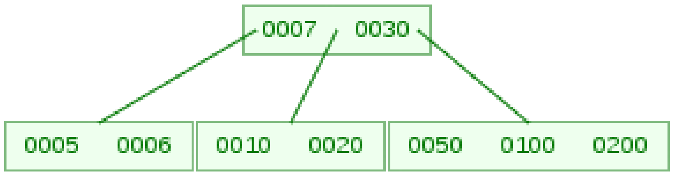


```
__________________________add 100  30
7
  |5, 6
  |  |null
  |  |null
  |  |null


  |10, 20, 30, 100, 200
  |  |null
  |  |null
  |  |null
  |  |null
  |  |null
  |  |null
```

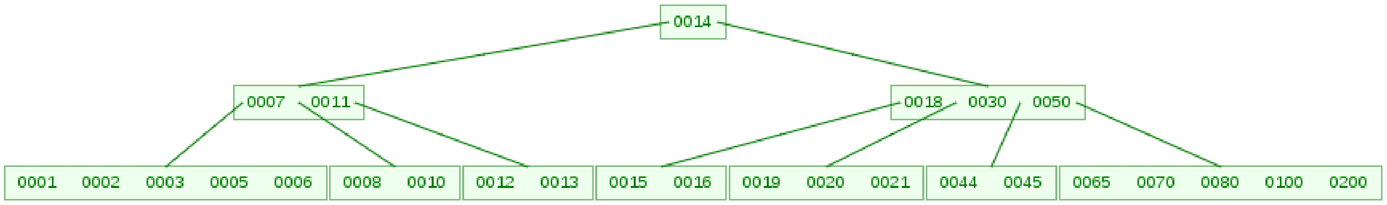


add 50

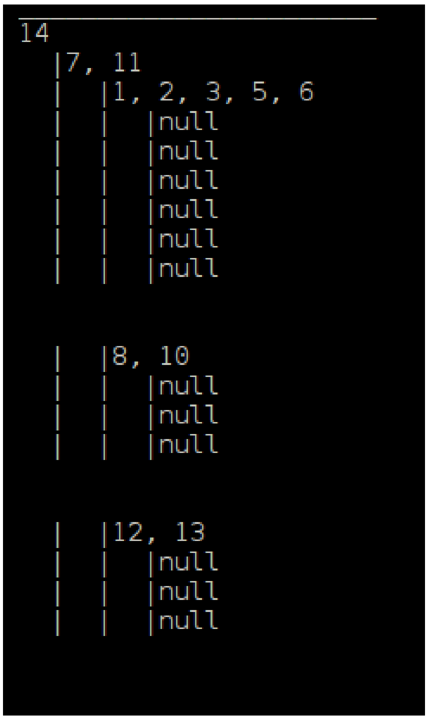
```
7, 30
 |5, 6
 |  |null
 |  |null
 |  |null


 |10, 20
 |  |null
 |  |null
 |  |null


 |50, 100, 200
 |  |null
 |  |null
 |  |null
 |  |null
```



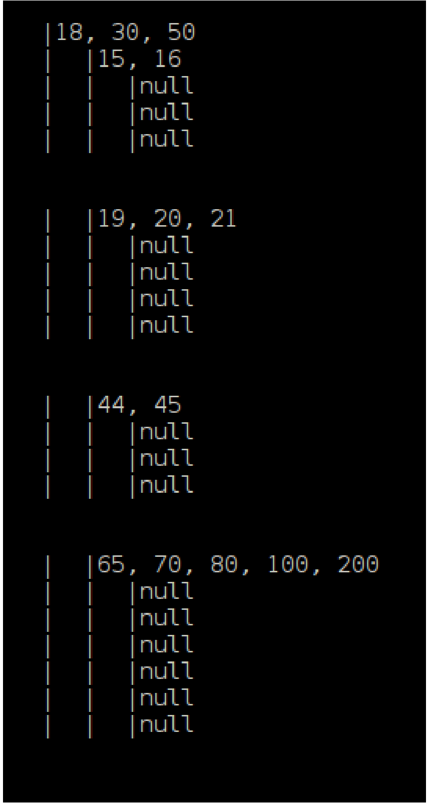
```
   bTree2.add(45);
   bTree2.add(44);
   bTree2.add(65);
   bTree2.add(70);
   bTree2.add(80);
   bTree2.add(15);
   bTree2.add(1);
   bTree2.add(2);
   bTree2.add(3);
   bTree2.add(8);
   bTree2.add(11);
   bTree2.add(12);
   bTree2.add(13);
   bTree2.add(14);
   bTree2.add(16);
   bTree2.add(18);
   bTree2.add(19);
   bTree2.add(21);
```



2.3 Running Commands and Results



```
14
  |7, 11
  |  |1, 2, 3, 5, 6
  |  |  |null
  |  |  |null
  |  |  |null
  |  |  |null
  |  |  |null
  |  |  |null


  |  |8, 10
  |  |  |null
  |  |  |null
  |  |  |null


  |  |12, 13
  |  |  |null
  |  |  |null
  |  |  |null
```



```
  |18, 30, 50
  |  |15, 16
  |  |  |null
  |  |  |null
  |  |  |null


  |  |19, 20, 21
  |  |  |null
  |  |  |null
  |  |  |null
  |  |  |null


  |  |44, 45
  |  |  |null
  |  |  |null
  |  |  |null


  |  |65, 70, 80, 100, 200
  |  |  |null
  |  |  |null
  |  |  |null
  |  |  |null
  |  |  |null
  |  |  |null
```

3 Project 9.5 in book

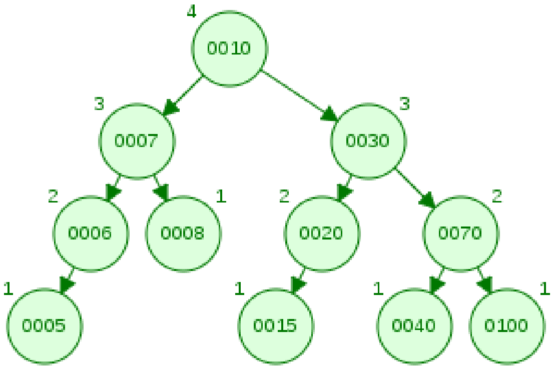
AVLTree(BinaryTree tree)
Constructor is taking a binary tree and check it,so print possitive message if it is avl tree else throw exception with negative message. Constructor checking this via isAVL() method. The method returns true if it s avl tree. İsAVL method also inserts all elements to this.root. The best part of remain, I coppied from source code of course book.

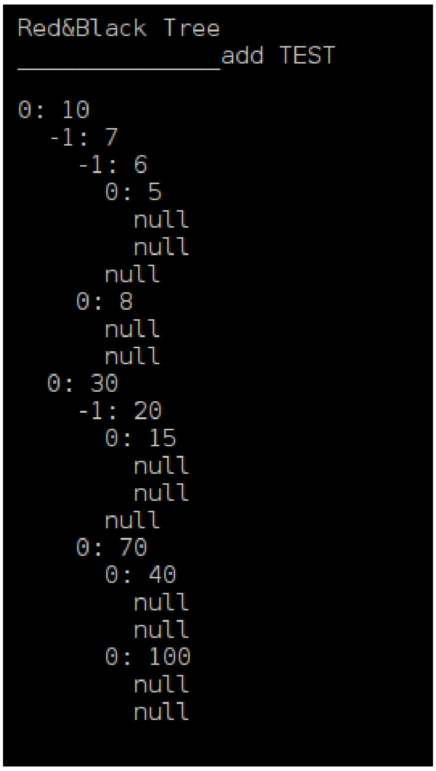
3.1 Problem Solution Approach

```
isAvl(node)
    if node EQU null then
        return true
    add(node.data)          //add new item in avl tree  - - reconstructor the tree
    addReturn ← false
    increase ← false
    this.root.balance ← 0

    leftHeight ← level(node.left)
    rightHeight ← level(node.right)

    if (leftHeight – rightHeight) <= 1 && isAvl(node.left)  &&  isAvl(node.right) then
        return true;
    return false


level(node)
    if node EQU null then
        return 0
    return 1 +getMax( level(node.left),  level(node,right))
```

3.2 Test Cases

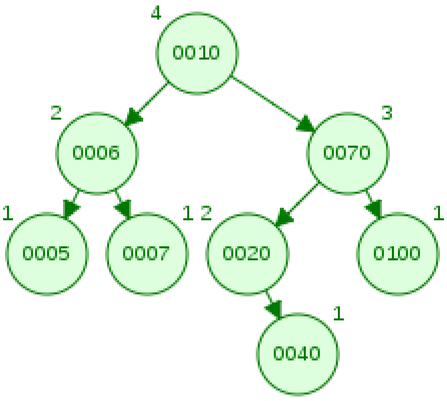
```
____add TEST
avlTree.add(5);
avlTree.add(10);
avlTree.add(20);
avlTree.add(30);
avlTree.add(40);
avlTree.add(100);
avlTree.add(70);
avlTree.add(6);
avlTree.add(7);
avlTree.add(8);
avlTree.add(15);
```

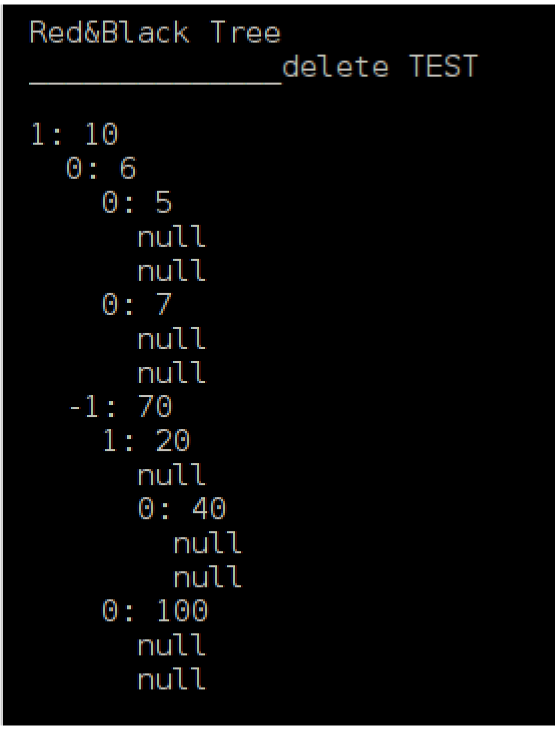




```
Red&Black Tree
______________add TEST

0: 10
  -1: 7
    -1: 6
      0: 5
        null
        null
      null
    0: 8
      null
      null
  0: 30
    -1: 20
      0: 15
        null
        null
      null
    0: 70
      0: 40
        null
        null
      0: 100
        null
        null
```

```
____delete TEST
avlTree.delete(30);
avlTree.delete(15);
avlTree.delete(8);
```

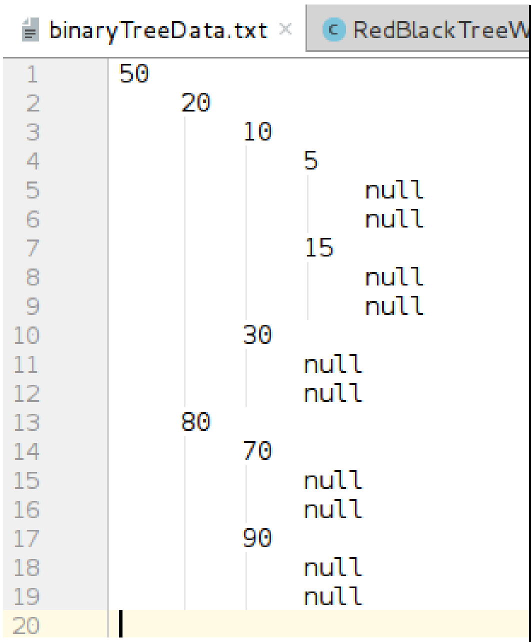




```
Red&Black Tree
______________delete TEST

1: 10
  0: 6
    0: 5
      null
      null
    0: 7
      null
      null
  -1: 70
    1: 20
      null
      0: 40
        null
        null
    0: 100
      null
      null
```

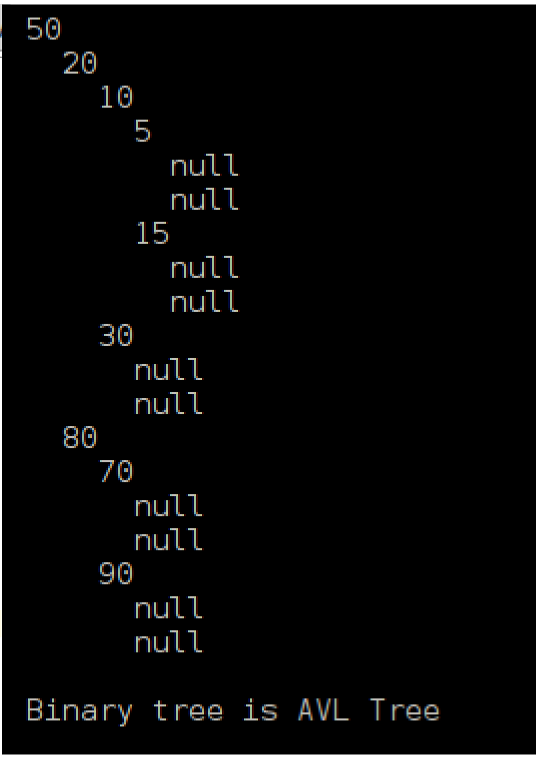
read binaryTree from file

file1
txt mode



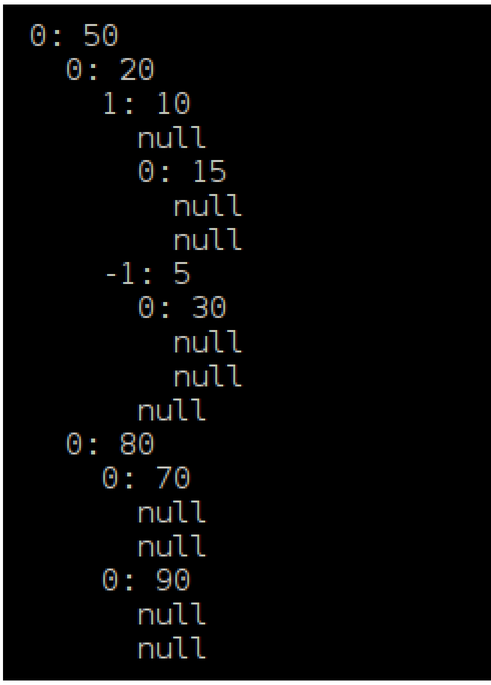
```
50
    20
        10
            5
                null
                null
            15
                null
                null
        30
            null
            null
    80
        70
            null
            null
        90
            null
            null

```

check avl

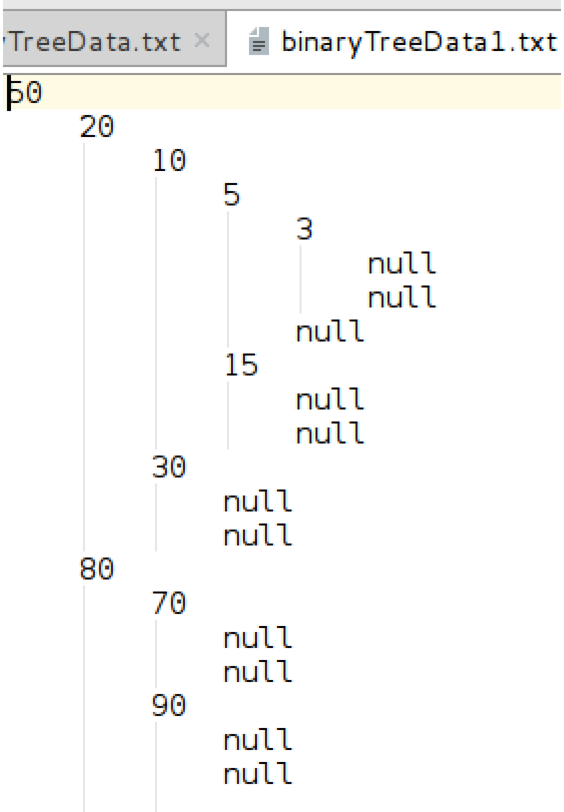


```
50
  20
    10
      5
        null
        null
      15
        null
        null
    30
      null
      null
  80
    70
      null
      null
    90
      null
      null

Binary tree is AVL Tree
```

copy to avl tree
(not correct exactly)

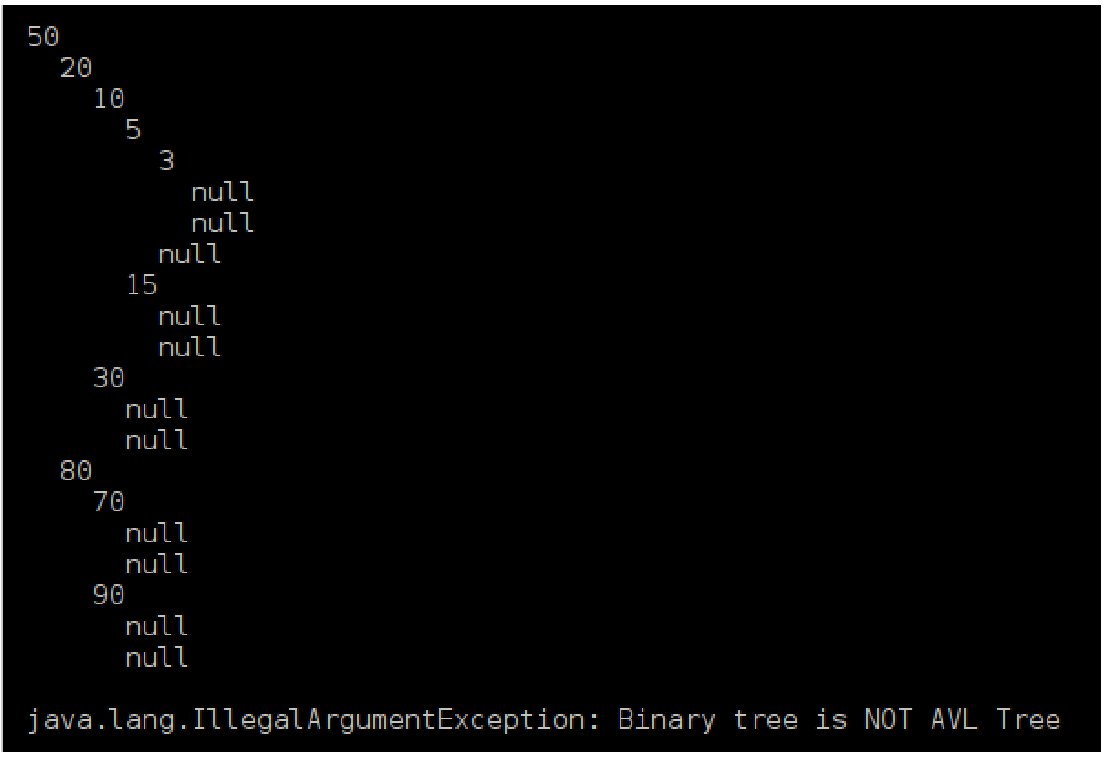


```
0: 50
  0: 20
    1: 10
      null
      0: 15
        null
        null
    -1: 5
      0: 30
        null
        null
      null
  0: 80
    0: 70
      null
      null
    0: 90
      null
      null
```



```
50
    20
        10
            5
                3
                    null
                    null
                null
            15
                null
                null
        30
            null
            null
    80
        70
            null
            null
        90
            null
            null
```

check **NOT** avl tree



```
50
  20
    10
      5
        3
          null
          null
        null
      15
        null
        null
    30
      null
      null
  80
    70
      null
      null
    90
      null
      null

java.lang.IllegalArgumentException: Binary tree is NOT AVL Tree
```

throw inception **intentionally**

3.3 Running Commands and Results

Show that test case results using screenshots.