

**Gebze Technical University  
Computer Engineering**

**CSE 222 - 2018 Spring**

**HOMEWORK 6 REPORT**

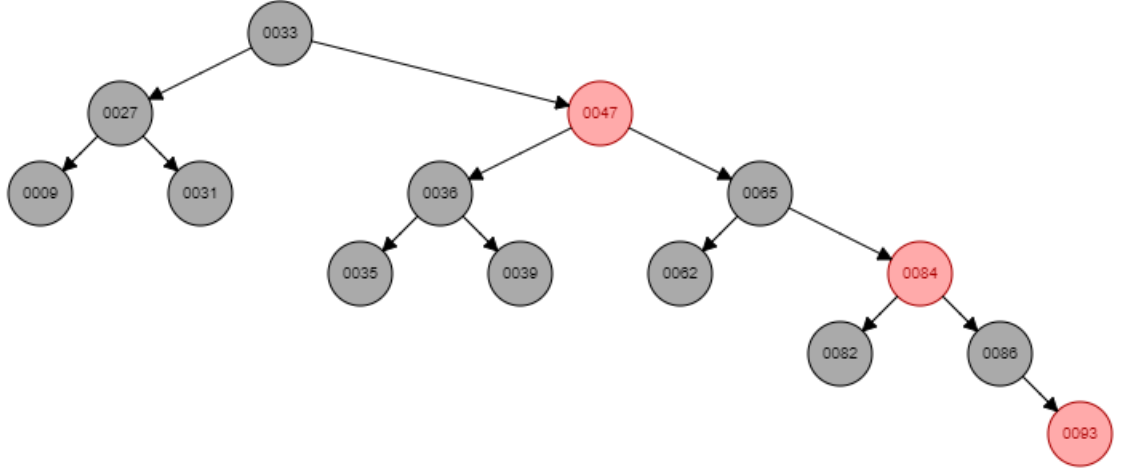
**AHMET MUZAFFER DÜLGER  
131044082**

Course Assistant: Fatma Nur Esirci

# 1 Worst RedBlack Tree

## 1.1 Problem Solution Approach

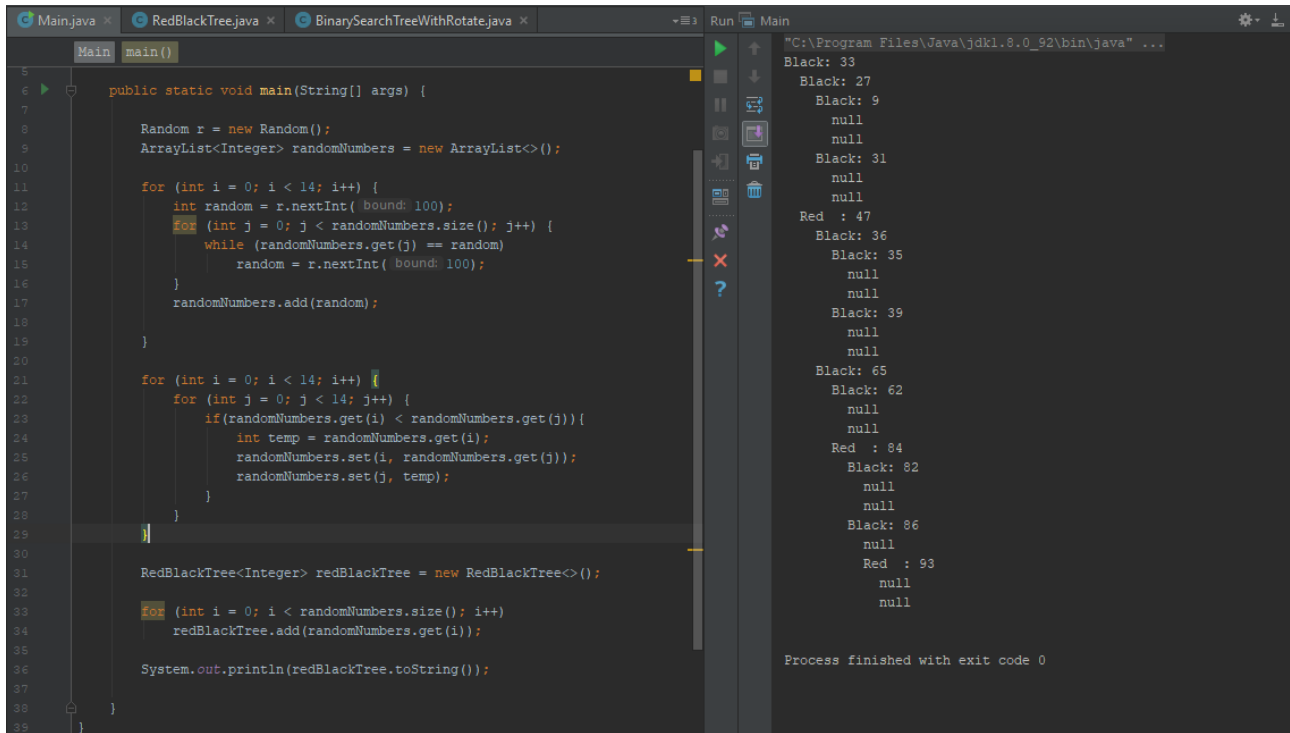
Bu partta yüksekliđi 6 olan en kötü Red Black Tree oluřturmamız istenildi. Bunun için ilk önce main de 14 adet random sayı ürettim ve ardından bu random sayıları küçükten büyüđe doğru sıraladım. Bunu yapmamdaki sebep ise 6 yüksekliđindeki en kötü RedBlack Tree'nin küçükten büyüđe veya büyükten küçüđe doğru sıralama yapıldıđında oluřmasıdır. Main kodu her çalıştıđında farklı bir tree imlement edilecektir.



## 1.2 Test Cases

14 adet random sayı (9,27,31,33,35,36,39,47,62,65,82,84,86,93) üretiliyor. Bu random sayılar küçükten büyüđe doğru sıralanıp AVLTree objesine tek tek ekleniyor. Oluřturulan ağacın son hali bölüm 1.3'deki gibidir.

## 1.3 Running Commands and Results



The screenshot shows an IDE with three tabs: Main.java, RedBlackTree.java, and BinarySearchTreeWithRotate.java. The Main.java tab is active, showing the main method. The code generates 14 random numbers, sorts them, and inserts them into a RedBlackTree. The output window on the right shows the tree structure with nodes colored Black or Red and their values. The process finished with exit code 0.

```
5 public static void main(String[] args) {
6
7     Random r = new Random();
8     ArrayList<Integer> randomNumbers = new ArrayList<>();
9
10
11     for (int i = 0; i < 14; i++) {
12         int random = r.nextInt( bound: 100);
13         for (int j = 0; j < randomNumbers.size(); j++) {
14             while (randomNumbers.get(j) == random)
15                 random = r.nextInt( bound: 100);
16         }
17         randomNumbers.add(random);
18     }
19
20
21     for (int i = 0; i < 14; i++) {
22         for (int j = 0; j < 14; j++) {
23             if(randomNumbers.get(i) < randomNumbers.get(j)){
24                 int temp = randomNumbers.get(i);
25                 randomNumbers.set(i, randomNumbers.get(j));
26                 randomNumbers.set(j, temp);
27             }
28         }
29     }
30
31     RedBlackTree<Integer> redBlackTree = new RedBlackTree<>();
32
33     for (int i = 0; i < randomNumbers.size(); i++)
34         redBlackTree.add(randomNumbers.get(i));
35
36     System.out.println(redBlackTree.toString());
37
38 }
39 }
```

Output:

```
Black: 33
Black: 27
Black: 9
null
null
Black: 31
null
null
Red : 47
Black: 36
Black: 35
null
null
Black: 39
null
null
Black: 65
Black: 62
null
null
Red : 84
Black: 82
null
null
Black: 86
null
Red : 93
null
null

Process finished with exit code 0
```

## 2 binarySearch method

### 2.1 Problem Solution Approach

Bu partta insert metodu içerisinde kullanılan binarySearch() metodunu implement etmemiz isteniyordu. Bu metot Binary Search Tree’de kullandığımız search metoduna çok benziyor ve yine aynı şekilde recursive mantık ile çalışıyor. Mantık olarak aynılar. Tek farkları burada first ve last olarak verdiğimiz aralıklarda arama yapması.

Bu metot 4 adet parametre alıyor. İlk parametre E tipinde item; aranacak öge. İkinci parametre E[] tipinde data; item bu sıralanmış biçimde duran datalar içerisinde aranır. Üçüncü parametre int tipinde first; aramanın nereden başlayacağını gösterir. Son parametre ise int tipinde last; aramanın nerede biteceğini gösterir.

İlk olarak first ve last’ın birbirine eşitlik durumu kontrol edilir. Eğer eşit iseler first return edilir. Bu, eleman olmadığını gösterir. Eğer ki last, first’ın 1 büyük ise item ve array’in first indexinde bulunan eleman karşılaştırılır ve gelen değer 0’dan küçükse first değilse last return edilir. Küçükse aradığımız item ağacın sol kısmında, aksi durumda da sağ kısmında kalıyor demektir.

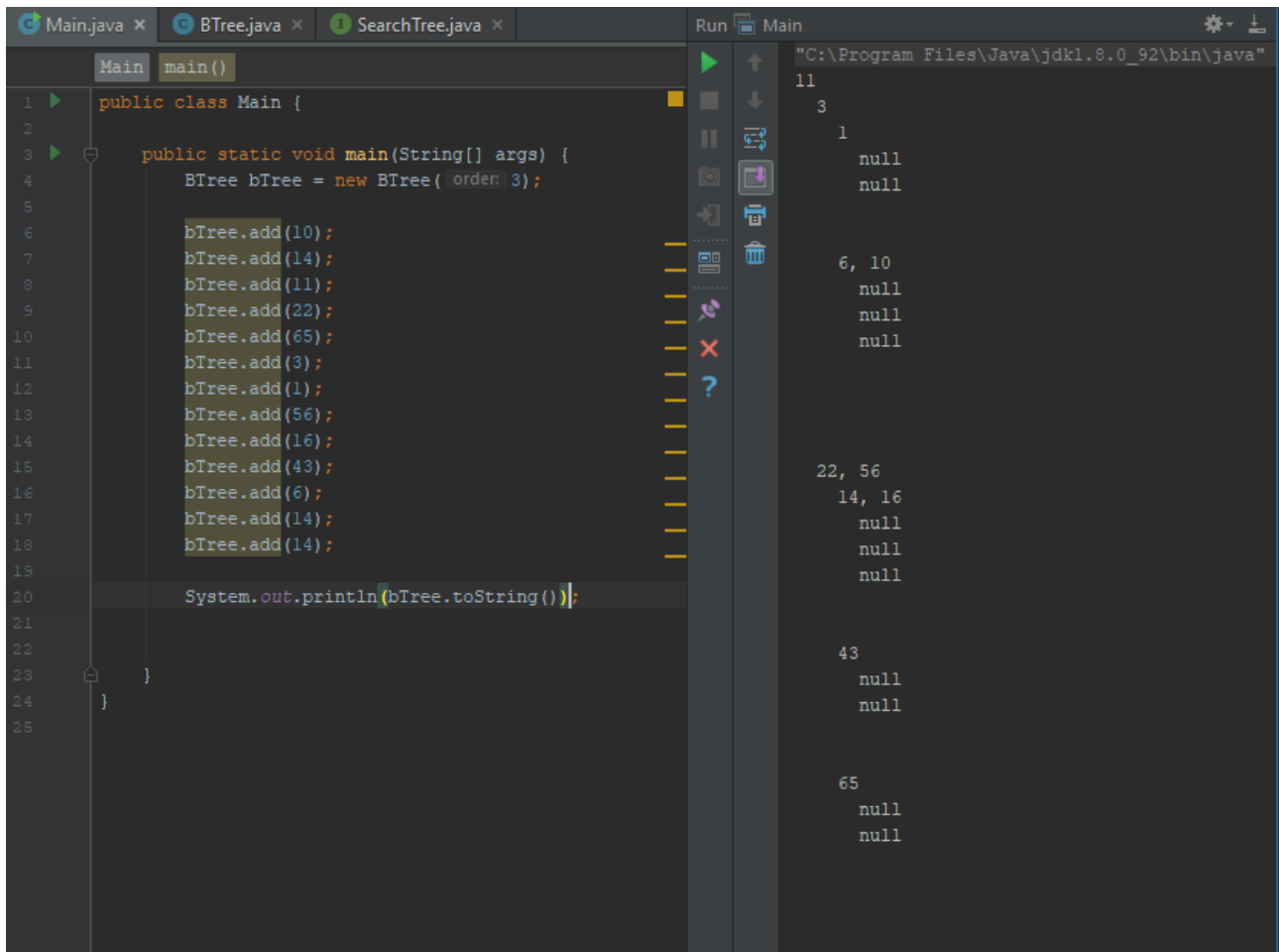
Daha sonra middle isminde bir integer değer tanımladım ve bu değer ile ortada bulunan kısmı temsil ettim. Ardından data’nın middle indexinde bulunan değer ile item’i karşılaştırdım ve bu karşılaştırmanın sonucunda 0 değeri geliyorsa ise bu ikisi eşit demektir ve middle’ı return ettim. Eğer karşılaştırma sonucunda 0’dan küçük bir değer geliyor ise ağacın solunda arama yapmam gerekiyor demektir. Bu yüzden tekrar binarySearch metodunu çağırdım. Bu çağırma parametre olarak sırasıyla item, data, first ve middle’ı gönderdim. Eğer karşılaştırma sonucunda 0’dan büyük bir değer geliyor ise ağacın sağında arama yapmam gerekiyor demektir. Bu yüzden tekrar binarySearch

metodunu çağırdım. Bu çağırmaya parametre olarak sırasıyla item, data, middle + 1 ve last'ı gönderdim.

## 2.2 Test Cases

Elle eklediğim integer değerlerden 3 order'a sahip bir BTree oluşturuluyor.

## 2.3 Running Commands and Results



The screenshot shows an IDE with three tabs: Main.java, BTree.java, and SearchTree.java. The Main.java file is open, showing the main method. The code creates a BTree with an order of 3 and adds several integers. The output of the program is displayed in the Run console, showing the tree structure as a series of values and nulls.

```
public class Main {  
    public static void main(String[] args) {  
        BTree bTree = new BTree( order: 3);  
  
        bTree.add(10);  
        bTree.add(14);  
        bTree.add(11);  
        bTree.add(22);  
        bTree.add(65);  
        bTree.add(3);  
        bTree.add(1);  
        bTree.add(56);  
        bTree.add(16);  
        bTree.add(43);  
        bTree.add(6);  
        bTree.add(14);  
        bTree.add(14);  
  
        System.out.println(bTree.toString());  
    }  
}
```

Run Console Output:

```
11  
3  
1  
null  
null  
6, 10  
null  
null  
null  
22, 56  
14, 16  
null  
null  
null  
43  
null  
null  
65  
null  
null
```

## 3 Project 9.5 in book

### 3.1 Problem Solution Approach

Bu partta decrementBalance, incrementBalance, rebalanceleft ve rebalanceRight metodlarının tekrar yazılmasının yanı sıra parametre olarak BinaryTree alan constructor yazmamız isteniyordu. Bu constructor parametre olarak gelen tree'nin AVL Tree olup olmadığının kontrolünü yapıyor.

Constructor'ı yaparken maxDepth ve minDepth olmak üzere iki adet yardımcı metod kullandım. Her iki metotda recursion mantıkta çalışmaktadır. maxDepth, yüksekliği en büyük olan node'un

yüksekliğini, minDepth ise yüksekliği en düşük olan node'un yüksekliğini return eder.

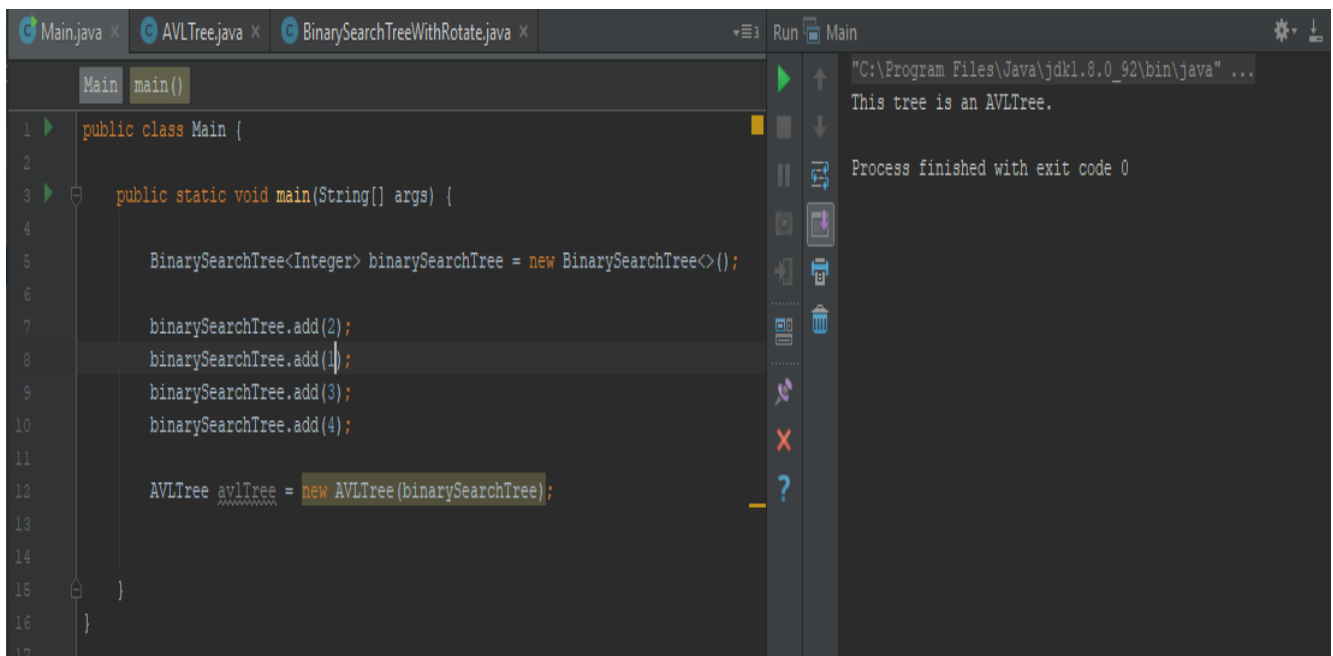
Constructor içerisinde ilk önce bu iki metodu çağırıp farklarını aldım. Bunu yapmamdaki amaç ise AVLTree olması için en düşük node ile en yüksek node'un yüksekliklerinin farkını 1 veya 0 olmasıdır. Eğer farkları 1 veya 0 ise AVLTree olduğuna dair ekrana bilgi mesajı bastırıyorum. Eğer bu değerlerden başka bir şey geliyor ise AVLTree olmadığına dair ekrana bilgi mesajı bastırıyorum.

### 3.2 Test Cases

Birinci test de AVLTree olan bir BinaryTree test edilmiştir.

İkinci test de ise AVLTree olmayan bir BinaryTree test edilmiştir.

### 3.3 Running Commands and Results

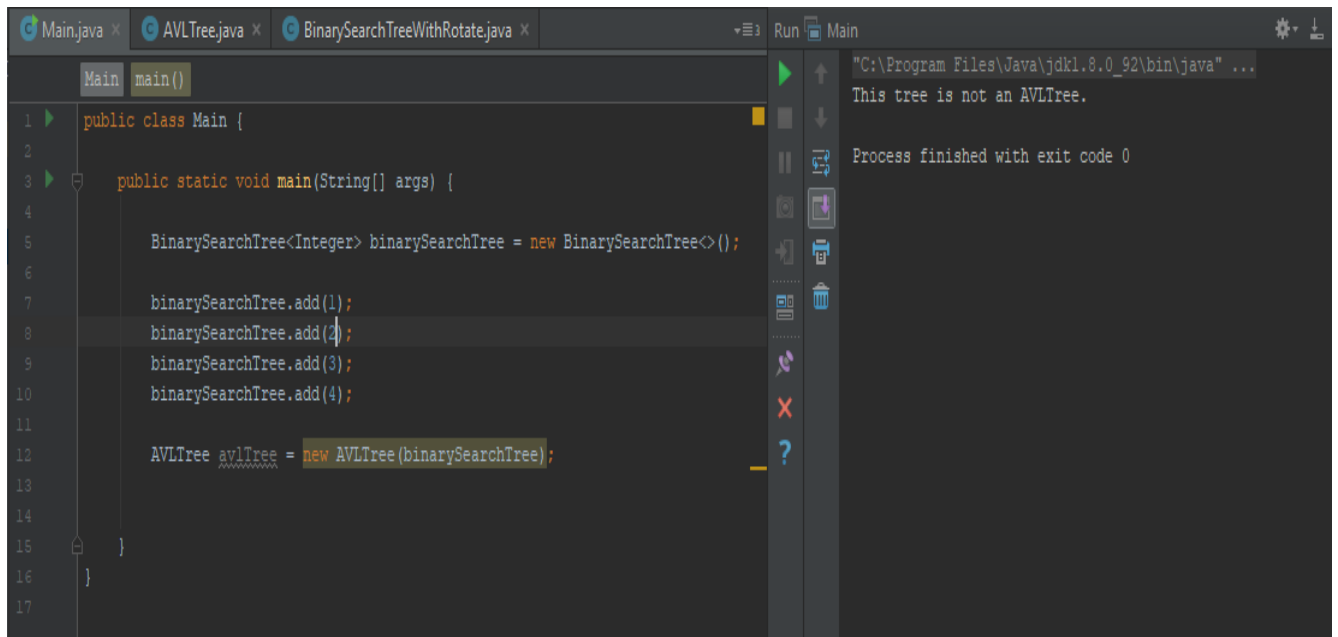


The screenshot shows an IDE with three tabs: Main.java, AVLTree.java, and BinarySearchTreeWithRotate.java. The Main.java tab is active, displaying the following code:

```
1 public class Main {
2
3     public static void main(String[] args) {
4
5         BinarySearchTree<Integer> binarySearchTree = new BinarySearchTree<>();
6
7         binarySearchTree.add(2);
8         binarySearchTree.add(1);
9         binarySearchTree.add(3);
10        binarySearchTree.add(4);
11
12        AVLTree avlTree = new AVLTree(binarySearchTree);
13
14    }
15 }
16
17
```

The Run button is clicked, and the output console shows the following messages:

```
"C:\Program Files\Java\jdk1.8.0_92\bin\java" ...
This tree is an AVLTree.
Process finished with exit code 0
```



The screenshot shows an IDE with three tabs: Main.java, AVLTree.java, and BinarySearchTreeWithRotate.java. The Main.java tab is active, showing the following code:

```
1 public class Main {
2
3     public static void main(String[] args) {
4
5         BinarySearchTree<Integer> binarySearchTree = new BinarySearchTree<>();
6
7         binarySearchTree.add(1);
8         binarySearchTree.add(2);
9         binarySearchTree.add(3);
10        binarySearchTree.add(4);
11
12        AVLTree avlTree = new AVLTree(binarySearchTree);
13
14    }
15 }
16
17
```

The Run button is clicked, and the output console shows the following message:

```
"C:\Program Files\Java\jdk1.8.0_92\bin\java" ...
This tree is not an AVLTree.
Process finished with exit code 0
```