**Gebze Technical University**
**Computer Engineering**

# CSE 222 - 2018 Spring

# HOMEWORK 6 REPORT

**SERKAN SORMAN**
**151044057**

Course Assistant: Fatma Nur Esirci

# 1 Worst RedBlack Tree

This part about Question1 in HW6

## 1.1 Problem Solution Approach

Comparable ve BinarySearchTreeWithRotate classlarından extend edilmiş olan RedBlackTree yapısı için add() metodundaki item > data durumu için eksik bölüm dolduruldu. Gerektiğinde parent ve childların rengini değiştirmeyi sağlayan moveBlackDown metodu yazıldı. En az node kullanılarak 6 tree yüksekliğine ulaşılması için her seferinde random sayılar üreten bir metod yazıldı. Treenin yüksekliği hesaplanırken farklı yaklaşımlar olduğu göz önünde bulundurularak root yüksekliğe dahil edilmeden (0 olarak alınıp) height 6 ya en az node ile ulaşılırken 22 node kullanıldı. Random sayılar üreten metod ürettiği sayıları büyükten küçüğe ya da küçükten büyüğe olarak eklendi bunun sebebi ağaçta sol ya da sağ ağırlıklı bir yapı oluşturup en az node ile height 6 ya ulaşmaktı. Problem diğer yükseklik hesaplama yaklaşımı ele alınsaydı (Root yüksekliğe dahil edilip) aynı şekilde büyükten küçüğe ya da ters sıralı bir şekilde 14 node eklenmesi height 6 ya ulaşmak için yeterli olacaktı.

Add metodu eksik kısım pseudocode:

```
Add()
        /*
         *
         */
        Else if item is greater than root data
                İf the right subtree is null
                        Insert a new node as right subtree and color it red
                Else
                        İf Both left and right child are red
                                Set color of children black and change local root red
                Recursively insert item into right subtree
                İf the right child is red
                        İf the right grandchild is red
                                Change the color of the right child to black and change
                                Local root to red
                                Rotate the local root left
                        Else if the left grandchild is red
                                Rotate the right child right
                                Change the color of the right child to black and change
                                Local root to red
                                Rotate the local root left


moveBlackDown()
        if  both local root left and right child are red
                Set black to left and right child
                Set red to local root
```

generateWorstTree()

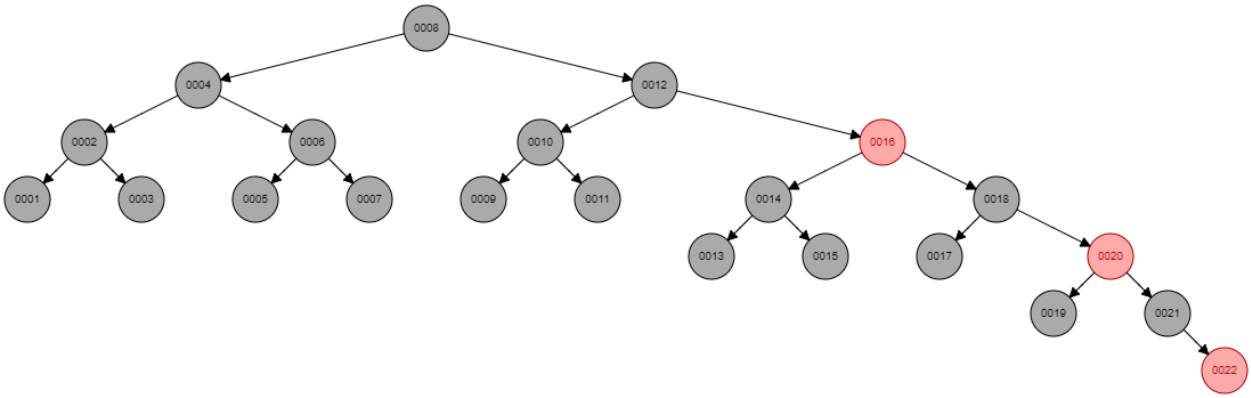
Generate random number

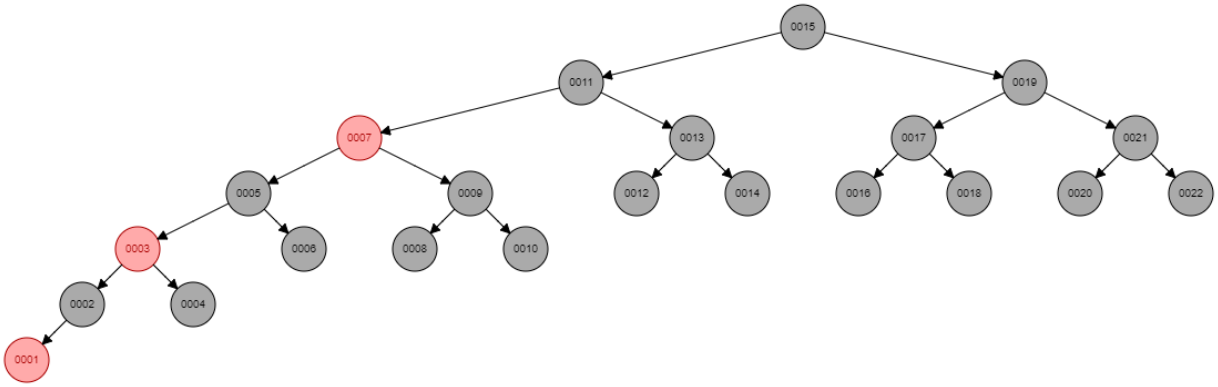
For i=0 to 22

Add numbers to tree according to flag ASCENDING or DESCENDING

## 1.2 Test Cases

generateWorstTree() metodu kullanılarak artan ya da azalan sırada sayılar tree ye eklenerek ve en az sayıda node kullanılmaya çalışılarak treenin yüksekliği metod her çağrıldığında test edildi. Sayılar artan sırada eklendiğinde sağ ağırlıklı bir ağaç ters sırada eklendiğinde sol ağırlıklı bir ağaç oluştuğu görüldü böylece height 6 ya en az 22 node kullanılarak ulaşılabileceği görüldü. Problem çözümünde bahsedildiği gibi height hesaplanırken root dahil edilseydi 6 yüksekliğine 14 node ile ulaşılabildiği ayrıca test edilmiştir. Ağaca her ekleme işlemi yapıldığında ağaçtaki değişimlerin gözlenmesi için ağaç print edilmiştir. Ters sıralı olarak eklenen tree main testtedir.



## 1.3 Running Commands and Results

```
        ##########   INSERT 12   ###########
Black: 12
  null
  null

        ##########   INSERT 14   ###########
Black: 12
  null
  Red: 14
    null
    null

        ##########   INSERT 18   ###########
Black: 14
  Red: 12
    null
    null
  Red: 18
    null
    null

        ##########   INSERT 20   ###########
Black: 14
  Black: 12
    null
    null
  Black: 18
    null
    Red: 20
      null
      null
```

```
        ##########   INSERT 29   ###########
Black: 14
  Black: 12
    null
    null
  Black: 20
    Red: 18
      null
      null
    Red: 29
      null
      null

        ##########   INSERT 30   ###########
Black: 14
  Black: 12
    null
    null
  Red: 20
    Black: 18
      null
      null
    Black: 29
      null
      Red: 30
        null
        null
```

```
        ##########   INSERT 32   ###########
Black: 14
  Black: 12
    null
    null
  Red: 20
    Black: 18
      null
      null
    Black: 30
      Red: 29
        null
        null
      Red: 32
        null
        null

        ##########   INSERT 37   ###########
Black: 20
  Red: 14
    Black: 12
      null
      null
    Black: 18
      null
      null
  Red: 30
    Black: 29
      null
      null
    Black: 32
      null
      Red: 37
        null
        null
```

```
        ##########   INSERT 42   ###########
Black: 20
  Black: 14
    Black: 12
      null
      null
    Black: 18
      null
      null
  Black: 30
    Black: 29
      null
      null
    Black: 37
      Red: 32
        null
        null
      Red: 42
        null
        null
```

```
       ##########   INSERT 47   ############
Black: 20
  Black: 14
    Black: 12
      null
      null
    Black: 18
      null
      null
  Black: 30
    Black: 29
      null
      null
    Red: 37
      Black: 32
        null
        null
      Black: 42
        null
        Red: 47
          null
          null
```

```
       ##########   INSERT 55   ############
Black: 20
  Black: 14
    Black: 12
      null
      null
    Black: 18
      null
      null
  Black: 30
    Black: 29
      null
      null
    Red: 37
      Black: 32
        null
        null
      Black: 47
        Red: 42
          null
          null
        Red: 55
          null
          null
```

```
       ##########   INSERT 58   ############
Black: 20
  Black: 14
    Black: 12
      null
      null
    Black: 18
      null
      null
  Black: 37
    Red: 30
      Black: 29
        null
        null
      Black: 32
        null
        null
    Red: 47
      Black: 42
        null
        null
      Black: 55
        null
        Red: 58
          null
          null
```

```
       ##########   INSERT 62   ############
Black: 20
  Black: 14
    Black: 12
      null
      null
    Black: 18
      null
      null
  Red: 37
    Black: 30
      Black: 29
        null
        null
      Black: 32
        null
        null
    Black: 47
      Black: 42
        null
        null
      Black: 58
        Red: 55
          null
          null
        Red: 62
          null
          null
```

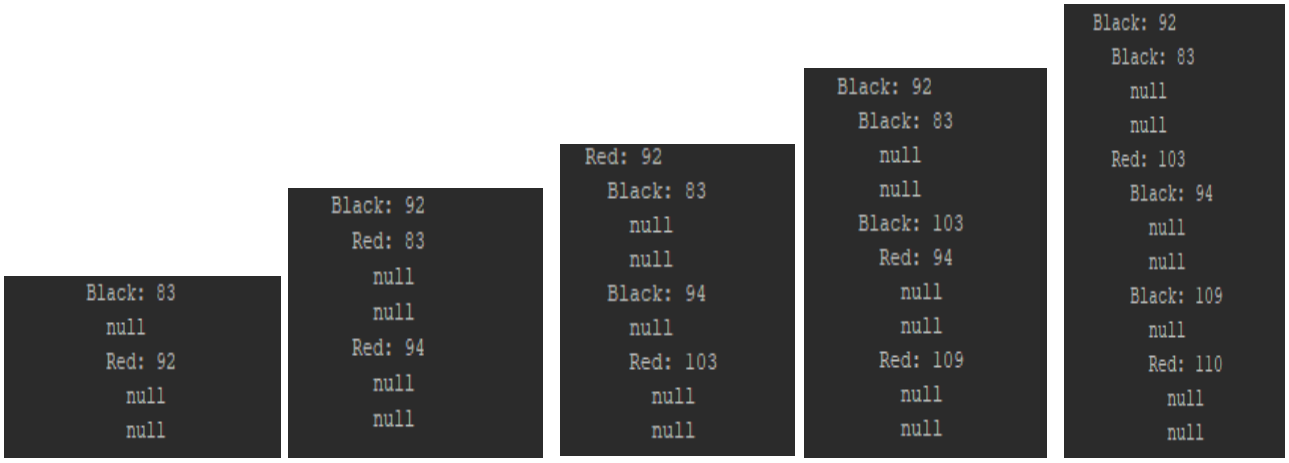
```
       ##########   INSERT 67   ###########
Black: 20
  Black: 14
    Black: 12
      null
      null
    Black: 18
      null
      null
  Red: 37
    Black: 30
      Black: 29
        null
        null
      Black: 32
        null
        null
    Black: 47
      Black: 42
        null
        null
      Red: 58
        Black: 55
          null
          null
        Black: 62
          null
          Red: 67
            null
            null
```

```
       ##########   INSERT 72   ###########
Black: 20
  Black: 14
    Black: 12
      null
      null
    Black: 18
      null
      null
  Red: 37
    Black: 30
      Black: 29
        null
        null
      Black: 32
        null
        null
    Black: 47
      Black: 42
        null
        null
      Red: 58
        Black: 55
          null
          null
        Black: 67
          Red: 62
            null
            null
          Red: 72
            null
            null
```

```
       ##########   INSERT 80   ###########
Black: 20
  Black: 14
    Black: 12
      null
      null
    Black: 18
      null
      null
  Red: 37
    Black: 30
      Black: 29
        null
        null
      Black: 32
        null
        null
    Black: 58
      Red: 47
        Black: 42
          null
          null
        Black: 55
          null
          null
      Red: 67
        Black: 62
          null
          null
        Black: 72
          null
          Red: 80
            null
            null
```

```
       ##########   INSERT 83   ###########
Black: 37
  Red: 20
    Black: 14
      Black: 12
        null
        null
      Black: 18
        null
        null
    Black: 30
      Black: 29
        null
        null
      Black: 32
        null
        null
  Red: 58
    Black: 47
      Black: 42
        null
        null
      Black: 55
        null
        null
    Black: 67
      Black: 62
        null
        null
      Black: 80
        Red: 72
          null
          null
        Red: 83
          null
```

INSERT 92 INSERT 94 INSERT 103 INSERT 109 INSERT 110




# 2 binarySearch method

This part about Question2 in HW6

## 2.1 Problem Solution Approach

BTree nodeları arraylerden oluştuğu için tree üzerinde binary search yapılamaz bundan dolayı her bir her bir nodeda bulunan array üzerinde binar search yapılabilir. Bu yüzden BTree yapısındaki insert() metodu içinde kullanılan binarySearch() metodu implement edildi. Bu metod kendisine gelen item, tree içinde ise bu itemin bulunduğu indexi return eder yoksa bu itemin tree ye eklenmesi için uygun bir yer arayan findAvailableIndex() metodu devreye girer ve BTreedeki nodelarda bulunan dataların sıralı halinin korunmasını sağlayan uygun bir index üretir.

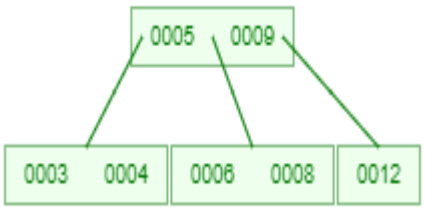
```
binarySearch()
        calculate middle index of array
        if end index > start index
                if item is equal to middle item of array
                        return middle index
                else if item is less than middle item of array
                        search item in left half of array recursively
                else
                    item is greater than middle item of array
                        search item in right half of array recursively

        if item is not in the array find available index for insert



findAvailableIndex()
        calculate size of node data array
        for i=0 to size
                if item is less than data[i]
                        return i for inserting to previous index

        if item is greater than all elements in data
```

return size of data array for inserting to next index

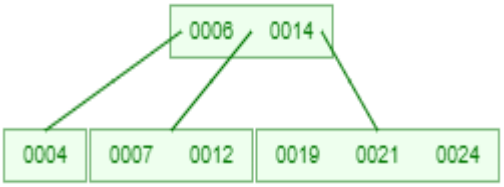
## 2.2 Test Cases

3 ordera sahip bir BTree oluşturulup elemanlar treeye eklenmiştir.Ardından simülasyon üzerinde aynı orderlı tree oluşturulup aynı elemanlar eklenmiş ve çıktıları karşılaştırılıp onaylanmıştır. Ardından aynı işlem 4 orderlı BTree üzerinde de test edilmiştir.

3 ORDER




4 ORDER




## 2.3 Running Commands and Results

```
    ----------------------  FIRST TREE ORDER 3 ----------------------

5, 9
  3, 4
    null
    null
    null

  6, 8
    null
    null
    null

  12
    null
    null


    ----------------------   SECOND TREE ORDER 4    ----------------------

6, 14
  4
    null
    null

  7, 12
    null
    null
    null

  19, 21, 24
    null
    null
    null
    null
```

# 3 Project 9.5 in book

This part about Question3 in HW6

## 3.1 Problem Solution Approach

BinaryTree alan bir constructor oluşturuldu ve binaryTree classının constructurundan esinlenerek binaryTree AVL treeye aktarıldı. Sonrasında isBalanced() ve helper height() metodları kullanılarak binary treenin AVL tree olup olmadığı test edilip exception fırlatıldı.

```
AVLTree()
        Create a root using binaryTree root data

        İf left subtree of binaryTree is not equals to null
                Set left subtree root to root left
        Else
                Set null to root left

        İf right subtree of binaryTree is not equals to null
                Set right subtree root to root right
        Else
                Set null to root right

        İf tree is not balanced
                Throw exception


isBalanced()
        if localRoot equals to null
                return true

        calculate left subtree of local root height recursively
        calculate right subtree of local root height recursively

        if(difference of left and right subtree <= 1 or >= -1)
                find height of left and right subtrees and check is balanced recursively


height()
        if local root equals to null
                return 0
        return 1+ max height of left and right subtrees
```

Add() metodunun eksik item > data durumu tamamlandı. rebalanceRight ve incrementBalance metodları yazılıp rebalanceLeft ve decrementBalance metodları modify edildi. Tüm bu metodların kullanıldığı remove() metodu yazıldı. İncrease fieldının yanına decrease fieldı eklendi

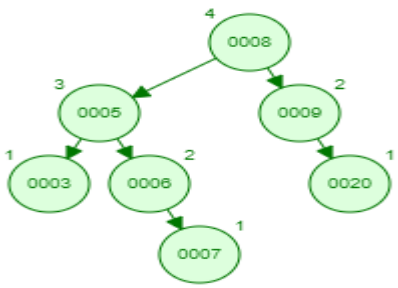
```
Add()
        /*
         *
         */
        Else if İtem is greater than local root data
```

```
            Recursively insert item in right subtree
            İf the height of the right subtree has increased
                İncrement balance
                İf balance is one, reset increase to false
                Perform a rebalance right

rebalanceRight()
    if right subtree has a negative balance
            if rightLeftChild has negative balance
                Set rightChild balance to 1
                Set rightLeftChild balance to 0
                Set localroot balance to 0
            Else if rightLeftChild has positive balance
                Set rightChild balance to 0
                Set rightLeftChild balance to 0
                Set localroot balance to -1
            Else
                Set rightChild balance to 0
                Set localroot balance to 0
            Set increase to false
            Set decrease to false
            Rotate the local root right
    Else
            Set rightChild balance to 0
            Set localroot balance to 0

            Set increase to false
            Set decrease to false

    Rotate the local root left



decrementBalance()
    decrement node balance by one
    if node balance is equals to 0
            Set increase false
    Else if node balance is equals to -1
            Set decrease false




incrementBalance()
    increment node balance by one
    if node balance is greater than zero
            Set increase true
            Set decrease false
    Else
            Set increase false
            Set decrease true
```

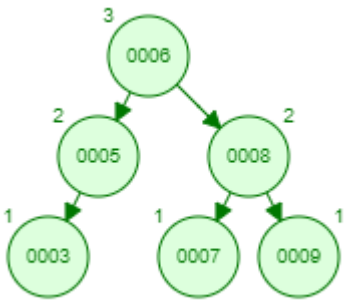
```
Remove()
        İf node equals to null
                Return null

        İf item is less than node data
                Recursively remove item from left sub tree
                İf the height of the left subtree has decreased
                        İncrement balance
                        İf node balance is greater than 1
                                Perform a rebalance right
        Else if item is greater than node data
                Recursively remove item from right sub tree
                İf the height of the right subtree has decreased
                        decrement balance
                        İf node balance is less than -1
                                Perform a rebalance left
        Else
                Set decrease to true
                Record node data to be deleted

                İf node has not left child
                        Perform with right child
                Else if node has not right child
                        Perform with left child
                Else
                        İf node has not left right child
                                Replace the data with the data in the left child
                                Replace the left child with its left child.

                                İncrement balance
                                İf node balance is greater than 1
                                        Rebalance right this node
                        Else
                                Recursively find node data that is inorder predecessor
                                Decrement balance
                                İf node balance is less than -1
                                        Rebalance left this node
```

## 3.2 Test Cases

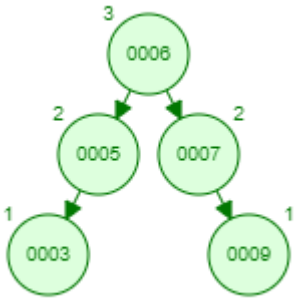
AVLTree oluşturulup rastgele sıradaki elemanlar treeye eklenmiştir ve treenin son hali ekrana basılmıştır.Aynı tree simülasyonda oluşturulup ağaçlar karşılaştırılmıştır.



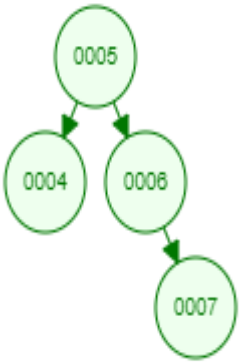
Ardından treede olmayan bir eleman remove edilmeye çalışıldı. Sonrasında 20 değerine sahip node treeden silinmeye çalışıldı başarıyla silindiğinin ve ağaçtaki değişimin gösterilmesi için tree ekrana basıldı. Aynı şekilde similasyonda da bu silme işlemi gerçekleştirildi ve treeler karşılaştırıldı.

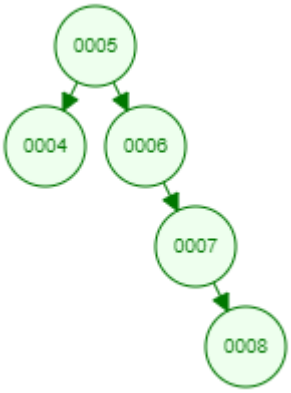



Devamında iki çocuğa sahip olan node 8 in silinmesi test edilip treenin çıktısı simülasyon ile karşılaştırıldı.




BinaryTree alan constructurun testinde ise dengeli bir BST oluşturulup try catch bloğu içinde constructura yollanarak test edildi. Ardından treenin dengesini bozmak için 8 numaralı node treeye eklendi ve aynı şekilde try catch bloğu içinde constructura yollanarak test edildi ve exception yakalandı.



## 3.3 Running Commands and Results

```
9 added
5 added
3 added
20 added
8 added
6 added
7 added
    -------------- AVL TREE ---------------
-1: 8
  1: 5
    0: 3
      null
      null
    1: 6
      null
      0: 7
        null
        null
  1: 9
    null
    0: 20
      null
      null
```

```
Element 99 can not found
Element 20 is removed!
    -------------- AVL TREE AFTER REMOVE 20 ---------------
0: 6
  -1: 5
    0: 3
      null
      null
    null
  0: 8
    0: 7
      null
      null
    0: 9
      null
      null

Element 8 is removed!
    -------------- AVL TREE AFTER REMOVE 8 ---------------
0: 6
  -1: 5
    0: 3
      null
      null
    null
  1: 7
    null
    0: 9
      null
      null
```

```
    --------------- CHECK BST IS AVL ----------------
5
  4
    null
    null
  6
    null
    7
      null
      8
        null
        null

This tree is not AVL tree
```