

**Gebze Technical University
Computer Engineering**

CSE 222 - 2018 Spring

HOMEWORK 6 REPORT

**EFKAN DURAKLI
161044086**

Course Assistant: Fatma Nur Esirci

1 Worst RedBlack Tree

Bu bölümde RedBlack tree için yüksekliği 6 olan en kötü ağaç oluşturuldu. RedBlack tree için kitabın kodları kullanıldı.

1.1 Problem Solution Approach

Kitabın RedBlackTree sınıfı içerisinde eksik metodlar ve tamamlanmamış metodlar vardı. Bunlar tamamlandı. Eksik olan metodlar rotateLeft(), moveBlackDown(). Add metodu içerisinde sol ağaç için ekleme yapan kısım yazılmıştı fakat, sağ ağaç için ekleme yapan kısım yazılmamıştı. Bu kısım yazıldı.

RedBlackTree için 6 yüksekliğindeki en kötü durumu bulmak için şu yollar izlendi.

- 6 yüksekliğindeki en kötü durum, en az elemanla 6 yüksekliğine ulaşabilmek demek.
- En kötü durum formülünden yararlanacak olursak;

$$2\log_2(n+1) = 6, \log_2(n+1) = 3, n+1 = 8, n = 7$$

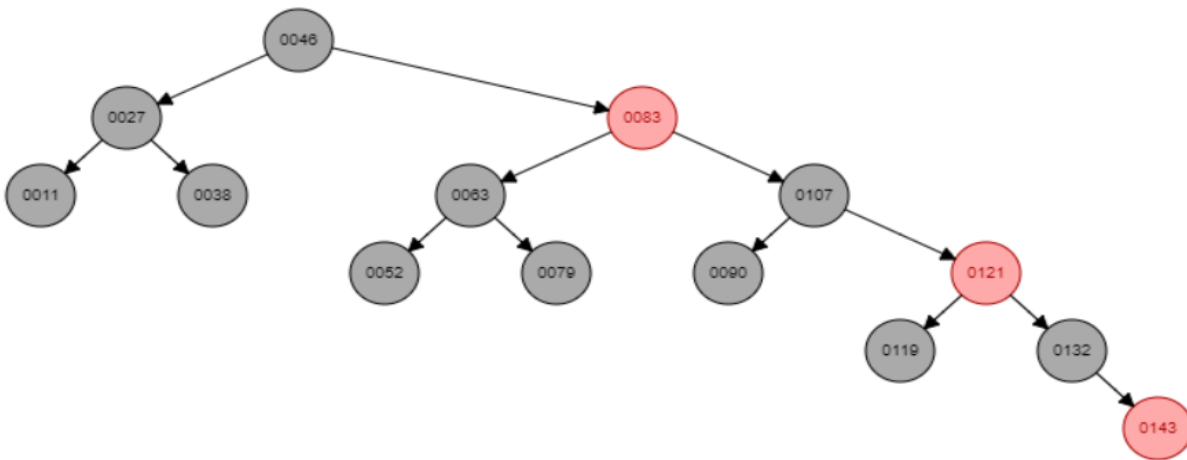
n : yaprak olamayan eleman sayısı

Eğer elemanları artan sırada veya azalan sırada ekleyecek olursak, en az 14 eleman ekleyerek en kötü duruma ulaşabiliyoruz. Yaprak olmayan eleman sayısı da 7 oluyor.

1.2 Test Cases

Case 1 :

Ağaca artan sırada 14 tane eleman eklendi. 6 yüksekliğine sahip en kötü ağaç oluşturuldu.

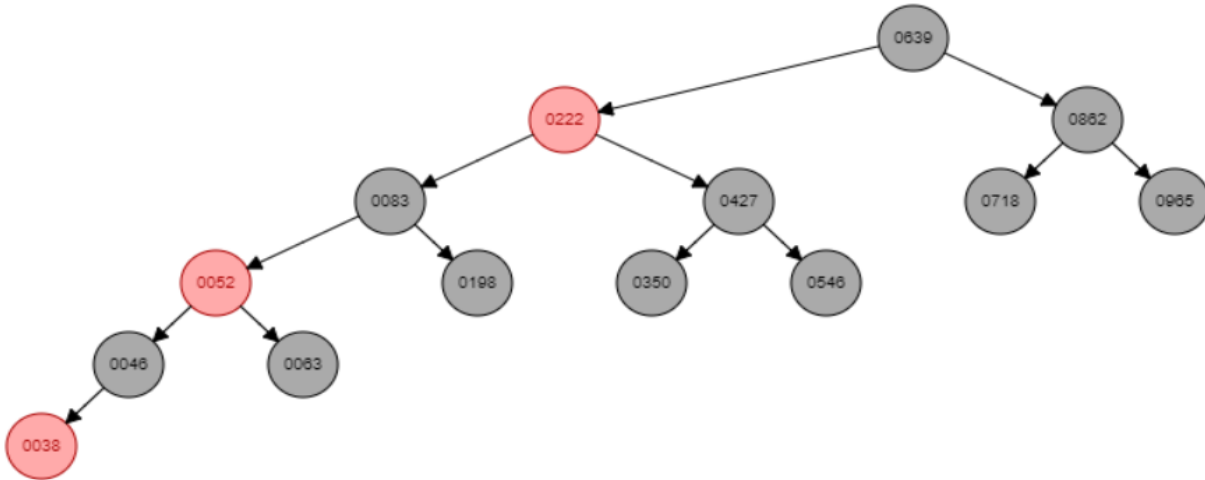


Internal node sayısı : 7

En kötü durumda ağacın yüksekliği : $2\log_2(n+1) = 2\log_2(8) = 6$

Case 2 :

Ağaca artan sırada 14 tane eleman eklendi. 6 yüksekliğine sahip en kötü ağaç oluşturuldu.



Internal node sayısı : 7

En kötü durumda ağacın yüksekliği : $2\log_2(n+1) = 2\log_2(8) = 6$

1.3 Running Commands and Results

```
TEST 1 :
After Adding 14 element to tree in ascending order
Black: 46
  Black: 27
    Black: 11
      null
      null
    Black: 38
      null
      null
  Red : 83
    Black: 63
      Black: 52
        null
        null
      Black: 79
        null
        null
    Black: 107
      Black: 90
        null
        null
      Red : 121
        Black: 119
          null
          null
        Black: 132
          null
          Red : 143
            null
            null
```

Case 1: Ağaca artan sırada 14 eleman eklendikten sonra ağacın durumu.

```

TEST 2 :
After Adding 14 element to tree in descending order
Black: 639
  Red : 222
    Black: 83
      Red : 52
        Black: 46
          Red : 38
            null
            null
            null
          Black: 63
            null
            null
        Black: 198
          null
          null
      Black: 427
        Black: 350
          null
          null
        Black: 546
          null
          null
    Black: 862
      Black: 718
        null
        null
      Black: 965
        null
        null

```

Case 2: Ağaca azalan sırada 14 eleman eklendikten sonra ağacın durumu.

2 binarySearch method

Bu bölümde kitabın BTree sınıfı içerisindeki insert metodu içerisinde implement edilmemiş olan binarySearch metodu implement edildi.

2.1 Problem Solution Approach

Bu metot verilen sıralı array içerisinde verilen elemanı binary olarak arar. Eleman array içerisinde bulunuyorsa bulundu index return edilir. Eğer eleman array içerisinde bulunmuyorsa eklenmesi gereken index return edilir.

binarySearch metodunun pseudo kodu aşağıdaki gibidir.

```

binarySearch(item, data, start, last)
if (last <= start)
  return start
middle = start + (last - start)/2
if ( data[middle] == item )
  return middle
if ( item < data[middle] )
  return binarySearch(item, data, start, middle-1)
return binarySearch(item, data, middle+1, last)

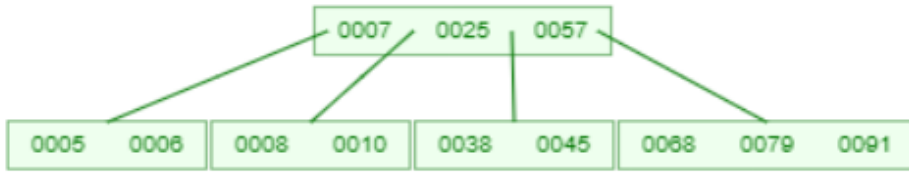
```

2.2 Test Cases

binarySearch metodu private bir metot ve add metodu içerisinde çağrılıyor. Bu metodu test etmek için add metoduyla elemanlar eklendi. Add metodunun doğru çalışması bu metodun doğru çalıştığı anlamına geliyor.

Case 1 :

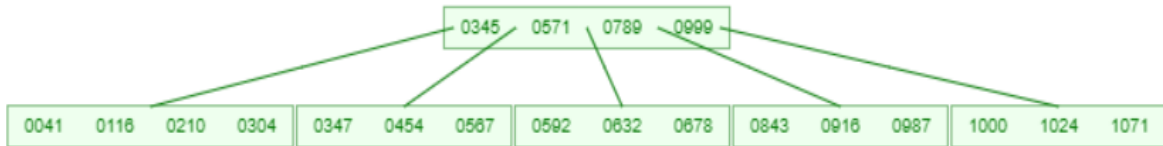
İlk teste order'ı 5 olan bir BTree oluşturuldu. Ağaca 12 farklı eleman eklendi. Aynı elemanı ekleyip eklemediğini test etmek için aynı eleman tekrar eklendi. Sonuçlar internetteki visualization ile karşılaştırıldı.



Ağaca eleman eklendikten sonra ağacın durumu.

Case 2 :

İlk teste order'ı 7 olan bir BTree oluşturuldu. Ağaca 20 farklı eleman eklendi. Sonuçlar internetteki visualization ile karşılaştırıldı.



Ağaca eleman eklendikten sonra ağacın durumu.

2.3 Running Commands and Results

Case 1:

```
TEST 1
After element to BTree
7, 25, 57
  5, 6
    null
    null
    null

  8, 10
    null
    null
    null

  38, 45
    null
    null
    null

  68, 79, 91
    null
    null
    null
    null
```

Case 2:

```
TEST 2
After element to BTree
210, 345, 843, 999
  41, 116, 304
    null
    null
    null
    null

  571, 454, 592
    null
    null
    null
    null

  347, 567, 632, 678
    null
    null
    null
    null
    null

  916, 789, 987
    null
    null
    null
    null

  1000, 1024, 1071
    null
    null
    null
```

3 Project 9.5 in book

Bu bölümde kitabın AVLTree kodları üzerine bazı eklemeler ve kodlarda bazı değişiklikler yapıldı. Bu eklemeler ve değişiklikler şunlardır.

- AVLTree sınıfına binarySearchTree alan bir constructor eklendi. Eğer gelen ağaç bir AVL ağacı ise bu ağacın elemanlarından oluşan bir AVL ağacı oluşturulur. Eğer gelen ağaç AVL ağacı değilse exception fırlatılır.
- rebalanceLeft metoduna isDelete adında boolean bir parametre eklendi. Bu parametre delete metodunda true değeriyle çağrılır, add metodunda false değeriyle çağrılır.
- rebalanceRight metodu ve incrementBalance metodu yazıldı.
- Delete metodu yazıldı.

3.1 Problem Solution Approach

Delete metodunu yazabilmek için aşağıdaki yollar izlendi.

- Sınıfın içerisine decrease adında bir boolean değer eklendi.
- Decrease flagini delete metodunun her recursive çağrısının dönüşünde kontrol edilir. Bu flagin doğru değere sahip olması alt ağacın yüksekliğinin değiştiği anlamına gelir. Bu durumda local root'un balansı bir azaltılır. Eğer local root'un balansı 0 olursa decrease flagi true yapılır. Eğer balans -1, +1, -2 yada 2 ise bu değer false yapılır. Bu durumda ağacın yeniden dengeye gelebilmesi için rebalance metodları çağrılır. Sonra local rootun balansı eğer 0 ise decrease flagi true yapılır.

3.2 Test Cases

Main Test

Bu testte AVL ağacına elemanlar eklendi. Balansın bozulduğu 4 durum(left-left heavy, left-right heavy, right-heavy ve right-left heavy) durumlar test edildi. Eklemenin sonucunda oluşan ağaç visulation sitesinde'ki sonuçla karşılaştırıldı. Ağaçta bulunan elemanı tekrar ekleyip eklemediğini kontrol etmek için aynı eleman birden fazla kez eklendi. Ağaçtan elemanlar silindi. Eleman silindikten sonra ağacı dengesi hale getiren durumlar test edildi. Sonuçlar visulation sitesindeki sonuçlarla karşılaştırıldı.

AVL ağacı yazılan constructor test edildi.

Unit Test

Delete metodu için ünit test yazıldı. Test başarılı bir şekilde sonlandı.

3.3 Running Commands and Results

Main Test Results

After adding element to AVL Tree.

```
0: 453
  1: 135
    0: 98
      0: 86
        null
        null
      0: 102
        null
        null
    1: 218
      0: 152
        null
        null
      1: 324
        null
        0: 347
          null
          null
  0: 807
    -1: 645
      0: 534
        0: 502
          null
          null
        0: 614
          null
          null
      0: 791
        null
        null
    -1: 951
      -1: 900
        0: 873
          null
          null
```

After deleting element 453, 502, and 791 from AVL Tree.

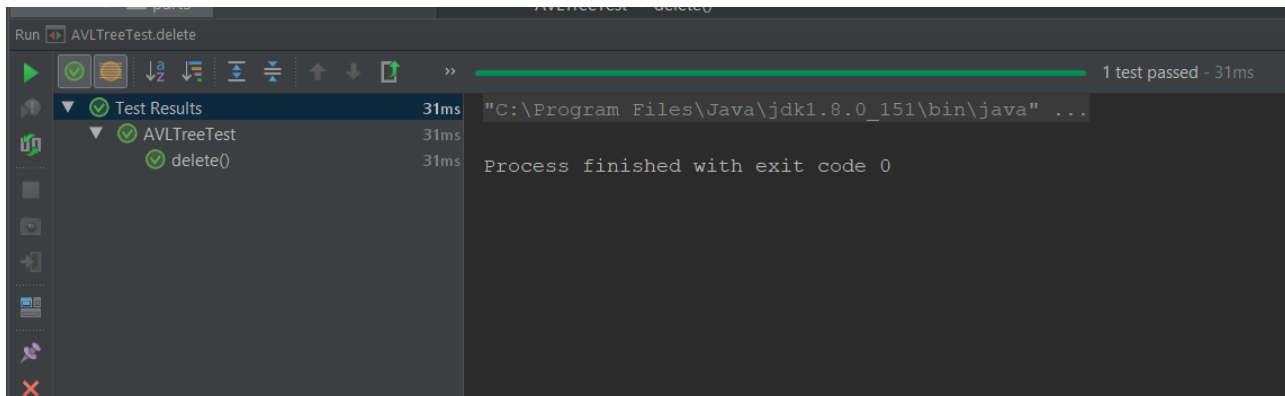
```
0: 347
  1: 135
    0: 98
      null
      0: 102
        null
        null
    1: 218
      0: 152
        null
        null
      1: 324
        null
        null
  0: 807
    -1: 645
      0: 534
        null
        0: 614
          null
          null
      null
    -1: 951
      -1: 900
        0: 873
          null
          null
      null
    0: 1041
      null
      null
```


Constructor Test Results

```
CONSTRUCTOR TEST
After creating new AVL tree from binaryTree
0: 86
  0: 73
    0: 61
      null
      null
    0: 79
      null
      null
  0: 98
    0: 90
      null
      null
    0: 105
      null
      null

EXCEPTION TEST
Given Binary Tree is not AVL Tree.
```

Unit Test Results



Run AVLTreetest delete

1 test passed - 31ms

Test Results	Time
AVLTreetest	31ms
delete()	31ms

"C:\Program Files\Java\jdk1.8.0_151\bin\java" ...

Process finished with exit code 0