

**Gebze Technical University
Computer Engineering**

CSE 222 - 2018 Spring

HOMEWORK 6 REPORT

**Furkan AKTAŞ
141044029**

Course Assistant: Fatma Nur Esirci

1 Worst RedBlack Tree

This part about Question1 in HW6

1.1 Problem Solution Approach

Red black tree sürekli kendini dengelediği için oluşabilecek max yüksekliğe($\log n$) bağlı olarak worst case senaryosu da dahil çalışma süresi $O(\log n)$ dir. Fakat worst case düşünüldüğünde rotate ve boyama işlemlerinin max seviyede olduğu senaryo akla getirebilir. Bunun için olabildiğince bu durumu elde edebilmek için ilk 2 elemandan sonraki elemanlar çapraz eklenerek, double rotate yada double renk değişimi sağlandı. 6 yüksekliğinde ağaç elde edebilmek için ağaca $63(2^6 - 1)$ node eklendi.

1.2 Test Cases

İlk durumda , root olarak 100 daha sonra right child olarak 1000 eklendi. Daha sonraki elemanlar sürekli küçülerek (999 dana geriye) eklendi. İlk rotate'ten sonra sol ağaçta eklenecek node'un çaprazına eklenerek ağaç devam etti. Bu sayede en az 2 rotate(ağaç uzadıkça üst node'larda daha fazla rotate olabilir) veya en az 3 renk boyama(kırmızı kardeşler siyah, grand parent kırmızı oldu, ayrıca ağaç uzadıkça üst node'larla bağlantılı olarak daha fazla renk değişimi olabilir). Çapraz eklenerek sırasıyla bu 2 durum tekrarlandı.

2. durumda ilk ağaç yapısının ters durumu yapıldı. Eklenen ilk node'dan(1000) sonra bu node'dan(100) küçük ve 2. eklenenden sonrakiler için büyüyerek(101, 102, ...) devam eden bir liste ile ağaç oluşturuldu.

Package içindeki main de 2 durum denendi. Bu yüzden test class yazılmadı.

1.3 Running Commands and Results

104

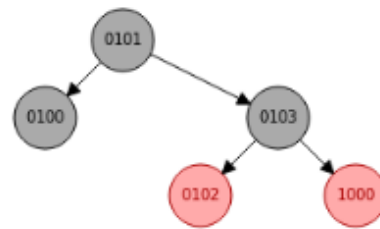
Insert

Delete

Find

Print

☐ Show Null Leaves



Red/Black Tree

105

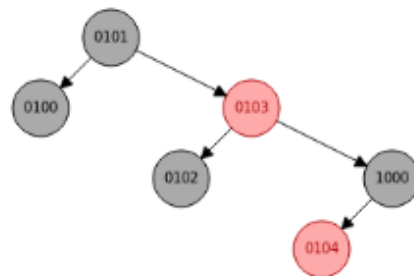
Insert

Delete

Find

Print

☐ Show Null Leaves



106

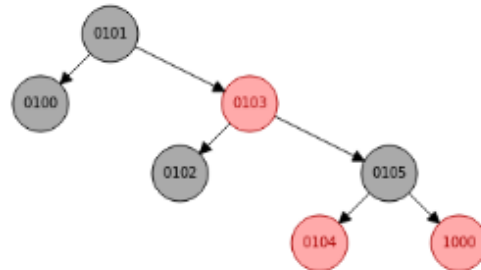
Insert

Delete

Find

Print

☐ Show Null Leaves



Red/Black Tree

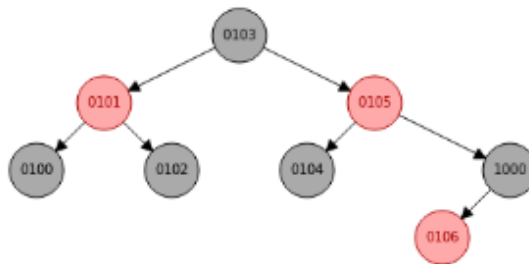
Insert

Delete

Find

Print

☐ Show Null Leaves



2 binarySearch method

This part about Question2 in HW6

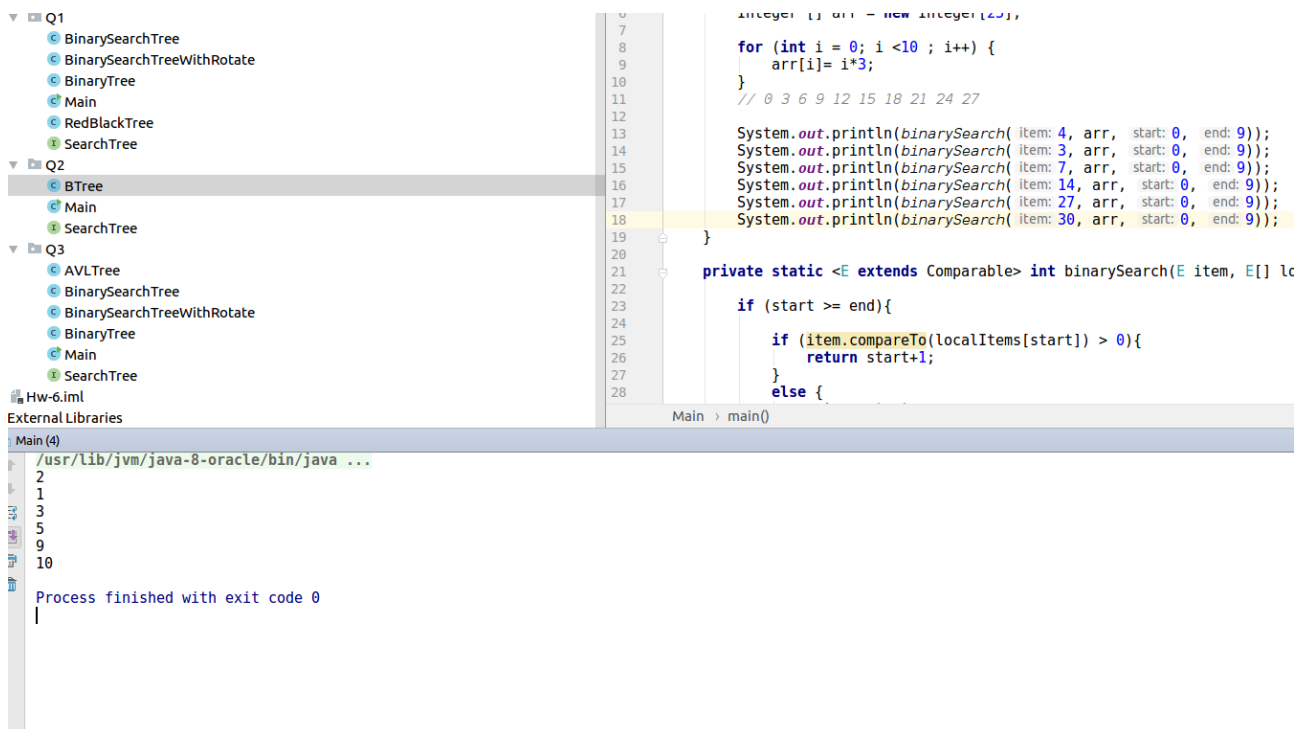
2.1 Problem Solution Approach

Eksik olan binarySearch fonksiyonu tamamlandı. Item varsa itemin indexi yoksa olması gereken index return edildi. Diğer Btree fonksiyonları için kitap kodları kullanıldı.

2.2 Test Cases

Kendi package'ındaki main de yazılan fonk , static olarak tekrar yazıldı. Ve basit 1 array üzerinde testi yapıldı.

2.3 Running Commands and Results



The screenshot shows an IDE with a project named 'Q1' containing several packages: 'BinarySearchTree', 'BinarySearchTreeWithRotate', 'BinaryTree', 'Main', 'RedBlackTree', and 'SearchTree'. The 'Q2' package is selected, showing 'BTree', 'Main', and 'SearchTree'. The 'Q3' package is also visible, containing 'AVLTree', 'BinarySearchTree', 'BinarySearchTreeWithRotate', 'BinaryTree', 'Main', and 'SearchTree'. The 'Main' class in the 'Q2' package is open, showing the following code:

```
Integer[] arr = new Integer[10],  
  
for (int i = 0; i < 10; i++) {  
    arr[i] = i*3;  
}  
// 0 3 6 9 12 15 18 21 24 27  
  
System.out.println(binarySearch( item: 4, arr, start: 0, end: 9));  
System.out.println(binarySearch( item: 3, arr, start: 0, end: 9));  
System.out.println(binarySearch( item: 7, arr, start: 0, end: 9));  
System.out.println(binarySearch( item: 14, arr, start: 0, end: 9));  
System.out.println(binarySearch( item: 27, arr, start: 0, end: 9));  
System.out.println(binarySearch( item: 30, arr, start: 0, end: 9));  
  
private static <E extends Comparable> int binarySearch(E item, E[] lc  
  
    if (start >= end){  
  
        if (item.compareTo(localItems[start]) > 0){  
            return start+1;  
        }  
        else {  

```

The output window shows the command `/usr/lib/jvm/java-8-oracle/bin/java ...` and the results of the binary search tests:

```
2  
1  
3  
5  
9  
10  
Process finished with exit code 0
```

3 Project 9.5 in book

This part about Question3 in HW6

3.1 Problem Solution Approach

Verilen BinaryTree nin AVL tree olup olmadığını anlamak için **isAVL()** fonksiyonu yazıldı. Add ve Delete fonksiyonlarının testleri için isAVL fonksiyonu public yapıldı. isAVL() fonksiyonunda recursive olarak tüm node ların sağ ve sol node lar farkına ve aynı zamanda SearchTree şartını sağlayıp sağlamadığını anlamak için sol ve sağ değerlerin root'un data sına göre doğru yerde olup olmadığına bakıldı. Constructora gelen Binary tree eğer AVL ise verilen tree nin elemanları kullanılarak AVL tree oluşturuldu. Aynı tree' yi elde etmek için öncelikle level order gezilerek elemanlar queue'da tutuldu. Daha sonra queue dan alınarak AVL tree oluşturuldu. Zaten AVL tree olduğundan, bu şekilde eklendiğinde aynı tree elde edilmiş oldu.

Verilen linkteki rebalanceLeft fonksiyonunun left-right case'indeki durum yanlışdı, düzeltildi. Eksik kodlar(rebalanceRight vs.) için simetrik karşılıkları yazıldı ve delete methodu için eklemeler yapıldı. Delete için öncelikle klasik Search Tree delete işlemi yapıldı. Bu işlem ve bundan sonra eğer varsa ağaçtaki bozulmalar için rebalance edildi. Silerken eğer silinen node'un 2 çocuğu varsa sağın solunu başa alma mantığı kullanıldı(sağın sol'u null sa sil, değilse en sola git onu sil).

Rebalance fonksiyonlarında temel mantık ,sol ağaç için sol sol ise sağ rotate, sol sağ ise sol + sağ rotate sağ ağaç için sol durumun simetrik durumu geçerli. Delete için gerekli olan rotate durumuda aynı mantık üzerinde kurulu, add ve delete durumlar arasında boolean değişkeni (isDelete) ile seçim yapılır. Bu seçim sadece rotate 'den(lerden) sonraki node'ların yeni balance durumlarını belirliyor. Balance methodlarında kitap kodlarından yardım alındı.

3.2 Test Cases

Package içindeki main de denendi. Bu yüzden test class yazılmadı.

Random 1000 sayı tree ye eklendi. isAVL fonksiyonu ile kontrol edilerek yapıldı. Daha sonra random seçilen sayılar silindi.

3.3 Running Commands and Results

1	2	3	4	5	6	7
<pre>/usr/lib/jvm/java-8-oracle/bin/jav</pre>			<pre>/usr/lib/jvm/java-8-oracle/bin/java</pre>			
Still AVL !!!			Still AVL !!!			
245 will delete			585 will delete			
245 is deleted			585 is deleted			
110 will delete			272 will delete			
110 is deleted			272 is deleted			
721 will delete			490 will delete			
721 is deleted			490 is deleted			
702 will delete			627 will delete			
702 is deleted			627 is deleted			
85 will delete			7 will delete			
85 is deleted			7 is deleted			
367 will delete			442 will delete			
367 is deleted			442 is deleted			
401 will delete			619 will delete			
401 is deleted			619 is deleted			
180 will delete			267 will delete			
180 is deleted			267 is deleted			
182 will delete			431 will delete			
182 is deleted			431 is deleted			
351 will delete			289 will delete			
351 is deleted			289 is deleted			
Process finished with exit code 0			Process finished with exit code 0			
ll files are up-to-date (4 minutes ago)			. files are up-to-date (moments ago)			

```
/usr/lib/jvm/java-8-oracle/bin/java .
Still AVL !!!
521 will delete
521 is deleted
797 will delete
797 is deleted
827 will delete
827 is deleted
625 will delete
625 is deleted
132 will delete
132 is deleted
220 will delete
220 is deleted
215 will delete
215 is deleted
399 will delete
399 is deleted
105 will delete
105 is deleted
930 will delete
930 is deleted

Process finished with exit code 0
```