Gebze Technical University
Computer Engineering

CSE 222 - 2018 Spring

HOMEWORK 6 REPORT

BATUHAN TOPALOĞLU
151044026

Course Assistant: Fatma Nur Esirci

1 Worst RedBlack Tree

1.1 Problem Solution Approach

Red- black tree BinarySearchTreeWithRotate den , BinarySearchTreeWithRotate ise BinarySearchTree den extends oluyor. BinarySearchTree de searchTree interface'ini implement ediyor

Bir tree yapısını worst case e düşürmek demek en az sayıda node ile en büyük yüksekliğe sahip ağacı oluşturmak demektir. RB tree'ler root'tan leaf'e kadar olan siyah node sayısı üzerinden balance sağladığı için RB tree yi normal edge sayısı yüksekliği bazından bir miktar balansı bozuk hale sokabilir. Bunu sağlamak için (en az sayıda node ile en büyük yükseklik) RB tree'lerin root'dan leaf'e giden bütün yollar üzerinde aynı sayıda siyah node olma koşulundan tree'nin bir tarafının minumum yükseklikte tutmak için ardarda siyah nodelar koyup diğer tarafında en az sayıda siyah node ile maximum yüksekliğe ulaşmak için siyah ve kırmızı nodelar peş peşe koymak gerektiği kanısına vardım.Tabiki tree'ye eklenecek node'un yerini ve rengine biz karar vermediğimiz için eklediğimiz elemanların birbirleriyle ilişkilerini bu senaryoyu sağlayacak şekilde seçmemiz gerekti. Tree'nin en az dengeli durumda olması için en az sayıda rotasyon yapması gerektiği gerçeği üzerinden de gidince çok basit ancak bulabildiğimi en basit yöntem olan küçükten büyüğe veya büyüktem küçüğe sıralı vaziyette elemanlar eklemek oldu. Sıralı 22 eleman eklenince ağaç 6 yüksekliğe ulaşıyor.

Generic olarak sıralı veya ters sıralı olarak eleman ekleme yöntemi bulamadım o yüzden örnek olarak Integer tipinde veriler tutan verilen yükseklik için en az sayıda node ile ağaç oluşturuğ return eden method'un pseudocode'u şu şekilde.

>>>

static RedBlackTree<Integer> createRBTreeInteger(int height)

Integer veriler tutan RedBlackTree tipinde rbt oluştur.

Integer değişken oluştururuz.

Ağaçın yüksekliği height den küçük olduğu sürece tekrarla

Ağaça o değişkeni ekleyip değişkenin değerini arttırız

rbt'yi return ederiz.

1.2 Test Cases

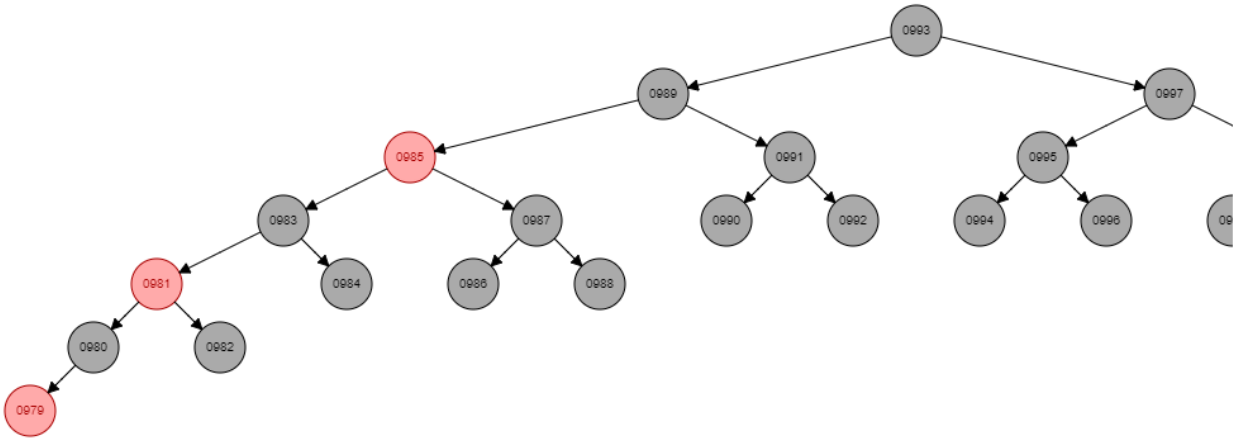
Integer ve String için 6 yükseklikte ağaçları en az sayıda node ile oluşturan methodlardan ınteger için olanının oluşturduğu ağaç ile visualizatin sayfasındaki ağaçları kıyasladım. Tree'ye girilen değerleri sıralı şekilde değilde düzendiz şekilde gönderdiğimiz 6 yüksekliğe çok daha fazla node girişinde ulaşabiliyor. Bunun testleri el ile bir çok farklı tipte deneyerek gözlemledim.

1.3 Running Commands and Results

*6 yükseklikte ağaç oluşturmak için createRBTreeInteger() methodunu 6 ile çağırdımızda şu şekilde bir ağaç oluşturuyor. 22 adet eleman eklenmiş.

```
"C:\Program Files\Java\jdk1.8.0_151\bin\java" ...
Black: 993
  Black: 989
    Red  : 985
      Black: 983
        Red  : 981
          Black: 980
            Red  : 979
              null
              null
            null
          Black: 982
            null
            null
        Black: 984
          null
          null
      Black: 987
        Black: 986
          null
          null
        Black: 988
          null
          null
    Black: 991
      Black: 990
        null
        null
      Black: 992
        null
        null
  Black: 997
    Black: 995
      Black: 994
        null
        null
      Black: 996
        null
        null
    Black: 999
      Black: 998
        null
        null
      Black: 1000
        null
        null
```

*Aynı elemanların eklenmesi sonuçu oluşması gereken ağıç ile örtüşuyor ağaç şu şekilde daha rahat görülebilir. (Sağ kısmı ekrana sığmadı).



*String için method arka planda küçük ‘a’ dan başlayarak ascıı karşılığını bir arttırığ ağaça ekliyor. Onun 22 karakter ile oluşturduğu ağaç :

```
"C:\Program Files\Java\jdk1.8.0_151\bin\java" ...
Black: Z
  Black: V
    Red  : R
      Black: P
        Red  : N
          Black: M
            Red  : L
              null
              null
            null
          Black: O
            null
            null
        Black: Q
          null
          null
      Black: T
        Black: S
          null
          null
        Black: U
          null
          null
    Black: X
      Black: W
        null
        null
      Black: Y
        null
        null
  Black: ^
    Black: \
      Black: [
        null
        null
      Black: ]
        null
        null
    Black: `
      Black: _
        null
        null
      Black: a
        null
        null


Process finished with exit code 0
```

*Methodlar istenilen yükseklikte ağaç oluşturabiliyor mu diye yapıtığım testin sonuçları:

2 binarySearch method

2.1 Problem Solution Approach

Btree class'ı searchTree<E> interface'ini implement ediyor. Ben iki farklı binarySearch methodu implement ettim. Birinicisi kitap kodlarında eksik olan insert methodu içerisinde kullanılan methdod ikincisi ise(binarySearchInBtree(E item)) parametre olarak E tipinde bir item alarak onu ağaç içerisinde her node'un data array'i içerisinde binarySearch yapmak koşulu ile arayan ve item tree içerisinde ise onun içinde bulunduğu node'u return eden bir method tanımladım. Bunlara yardımcı olarak binarySearchInNode ve binarySearchInBtree(Node<E> Root, int first, int last, E item) methodlarını tanımladım.

1-> private int binarySearch(E item,E[] arr, int first, int last) methodu arr içerisinde first ve last indexler arasında item'ı binarySearch mantığı ile arayan ve item arr' üzerinde bulunması gereken index'i return eden bir method . İnsert içerisinde kullanılan method. Bu method içerisinde node içinde search yapmak içim binarySearchInNode methodunu implement ettim. Bu method item'ı arr üzerinden bulursa index'ini bulamazsa last'ın bir fazlasının negatifini return ediyor.

binarySearch Pseudocode :

```
int binarySearch(E item,E[] arr, int first, int last){

    result = binarySearchInNode(arr,first, last,item);
    Eğer result 0 dan küçükse:
        Eğer item arr'ın ilk elemanından küçükse :
            return 0

         for( i = arr'ın ilk index'in den last-1'inci indeksine kadar)
                Eğer item arr i'ninci elemanından büyük ve i+1 den küçükse:
                    return i+1

         return last

    Result sıfırdan büyük :
        return result
}
```

2->binarySearchInBtree Psuedocode :

Dışaradan çağrılabilecek olan binarySearchInBtree:-
>>

```
   Node binarySearchInBtree(E item)
        return binarySearchInBtree(root,0,root.size,item);
```

Sadece yukarıdaki methodun çağırabildiği ve recursive olarak aramayı yapan method:
>>

(sonraki sayfada)

```
Node  binarySearchInBtree(Node Root, int first, int last, E item){

    index = binarySearchInNode(Root.data, first, Root.size, item);

    Eğer index >=0 :
        return Root


    Eğer itemRoot.datasının ile elemanından küçükse:
        Eğer Root.child[0] boş değilse:
            return binarySearchInBtree(Root.child[0],0,Root.child[0].size,item)


    for(i = sıfırdan root'un size-1 ine kadar)
        Eğer item root.data[i] sinden büyük ve root.data[i+1]'inden küçükse :
            Eğer Root.child[i+1] boş değilse :
                return binarySearchInBtree(Root.child[i+1],0,Root.child[i+1].size,item)



    Eğer item Root.datasının son elemanından büyükse :
        Eğer Root.child[Root.size] boş değilse:
            return binarySearchInBtree(Root.child[Root.size],0,Root.child[Root.size].size,item)
         return null
```

2.2 Test Cases

Try this code to search least 4 element on 2 different BTree. Report all of situations.
Integer ve String veriler tutabilen iki farklı Btree oluşturdum. Treelere elemanlar ekleyerek eklemeleri doğru yerlere yapıp yapmadığı cs.usfca sitenide de aynı elemanları aynı sıra ile tree ye ekleyerek kontol ettim. Bu şekilde insert içinde kullanıcal binarySearch methodumun doğru çalışıp çalışmadığını test etmiş oldum.

Tree içerisinde item'ı binarySearch mantığı ile arayarak item'ın içinde bulunduğu node'u return eden method'u ise arama sonuçlarının ne olması gerektiğini daha önceden hesaplayarak fonksiyonun ürettiği sonuçlar ile kıyaslayarak bir test gerçekleştirdim. Bu testi her iki tip ağaç için ve her node da bir item'ı arayarak tam olarak fonksiyonalitisini kontrol ederek yaptım.

2.3 Running Commands and Results

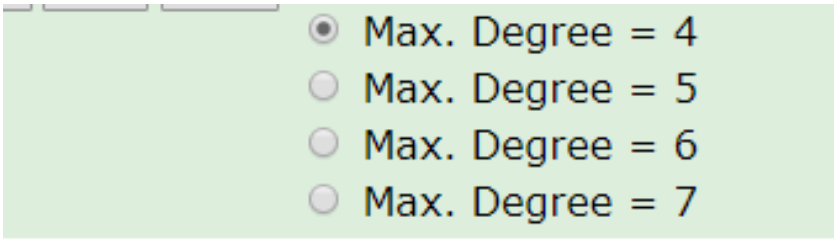
* Doğru yere insert edip etmediğini test ederek insert içindeki binarySearch methodunu test etmiş olacağım.

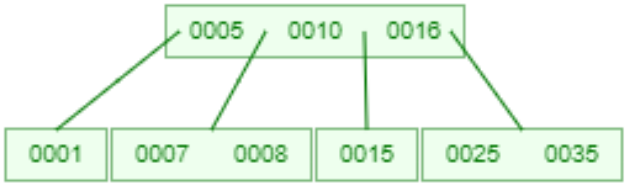
Integer ağaç için->>

5/10/ 15/ 25/7/ 16 /35 /1/ 8 inputlarını sıra ile verince benin yazdığım btree'nin oluşturduğu ağaç:

```
"C:\Program Fil
5, 10, 16
  1
    null
    null


  7, 8
    null
    null
    null


  15
    null
    null


  25, 35
    null
    null
    null
```

Benzetim programında oluşan ağaç:





Görüldüğü gibi elemanların yerleri ve node dağılımları birebir örtüşüyor.

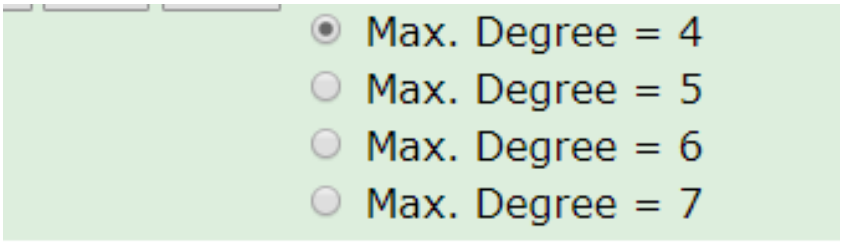
String ağaç için ->>

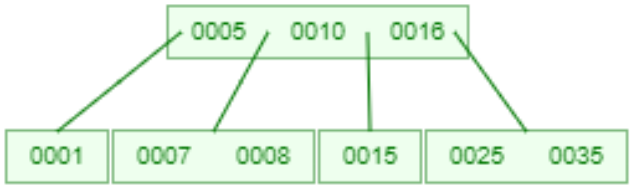
The / quick / brown / fox / jump / over / the / lazy / dog

inputlarını sıra ile verince benin yazdığım btree'nin oluşturduğu ağaç:

```
"C:\Program Files\J
fox, over
  The, brown, dog
    null
    null
    null
    null

  jumps, lazy
    null
    null
    null

  quick, the
    null
    null
    null
```

Benzetim programında oluşan ağaç:

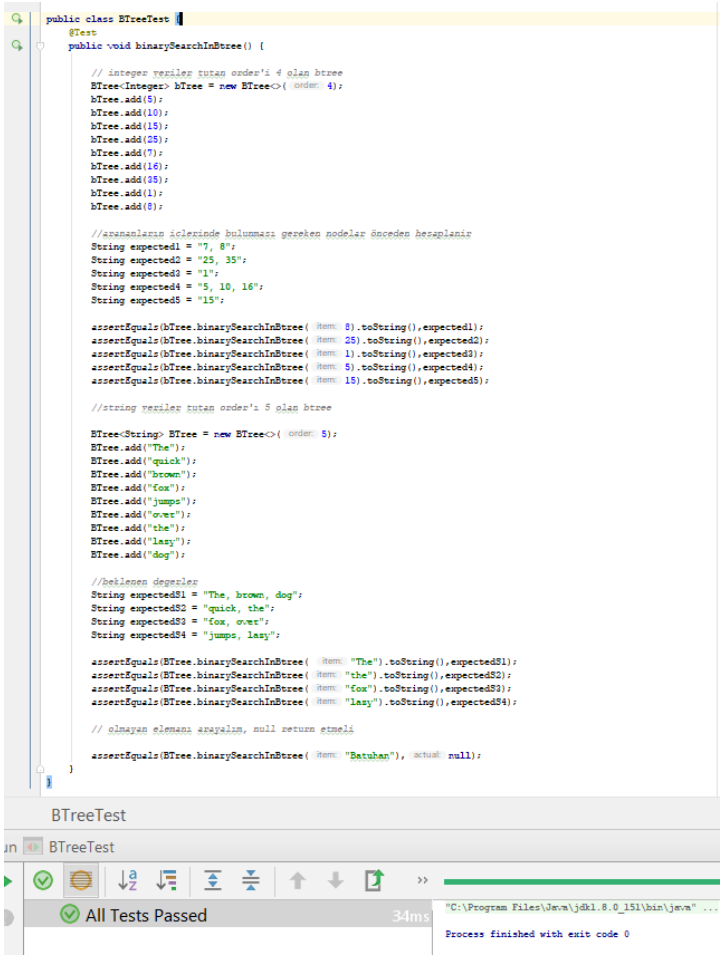




Görüldüğü gibi elemanların yerleri ve node dağılımları birebir örtüşüyor.

**İtem arayan search methodu :

(bütün testleri aynı anda geçtiğini göstermek için koydum detaylı olarak aşşağıda anlattım)



```
public class BTreeTest {
    @Test
    public void binarySearchInBtree() {

        // integer veriler tutan order'ı 4 olan btree
        BTree<Integer> bTree = new BTree<>( order: 4);
        bTree.add(5);
        bTree.add(10);
        bTree.add(15);
        bTree.add(25);
        bTree.add(7);
        bTree.add(16);
        bTree.add(35);
        bTree.add(1);
        bTree.add(8);

        //arananların içlerinde bulunması gereken nodelar önceden hesaplanır
        String expected1 = "7, 8";
        String expected2 = "25, 35";
        String expected3 = "1";
        String expected4 = "5, 10, 16";
        String expected5 = "15";

        assertEquals(bTree.binarySearchInBtree( item: 8).toString(),expected1);
        assertEquals(bTree.binarySearchInBtree( item: 25).toString(),expected2);
        assertEquals(bTree.binarySearchInBtree( item: 1).toString(),expected3);
        assertEquals(bTree.binarySearchInBtree( item: 5).toString(),expected4);
        assertEquals(bTree.binarySearchInBtree( item: 15).toString(),expected5);

        //string veriler tutan order'ı 5 olan btree

        BTree<String> BTree = new BTree<>( order: 5);
        BTree.add("The");
        BTree.add("quick");
        BTree.add("brown");
        BTree.add("fox");
        BTree.add("jumps");
        BTree.add("over");
        BTree.add("the");
        BTree.add("lazy");
        BTree.add("dog");

        //beklenen degerler
        String expectedS1 = "The, brown, dog";
        String expectedS2 = "quick, the";
        String expectedS3 = "fox, over";
        String expectedS4 = "jumps, lazy";

        assertEquals(BTree.binarySearchInBtree( item: "The").toString(),expectedS1);
        assertEquals(BTree.binarySearchInBtree( item: "the").toString(),expectedS2);
        assertEquals(BTree.binarySearchInBtree( item: "fox").toString(),expectedS3);
        assertEquals(BTree.binarySearchInBtree( item: "lazy").toString(),expectedS4);

        // olmayan elemanı arayalım, null return etmeli

        assertEquals(BTree.binarySearchInBtree( item: "Batuhan"), actual: null);
    }
}
```

BTreeTest

BTreeTest

All Tests Passed 34ms

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

Process finished with exit code 0

Tek tek durumları incelersek;

1- order'ı 4 olan integer btree için :

```
@Test
public void binarySearchInBtree() {

    // integer veriler tutan order'i 4 olan btree
    BTree<Integer> bTree = new BTree<>( order: 4);
    bTree.add(5);
    bTree.add(10);
    bTree.add(15);
    bTree.add(25);
    bTree.add(7);
    bTree.add(16);
    bTree.add(35);
    bTree.add(1);
    bTree.add(8);

    //arananların içlerinde bulunması gereken nodelar önceden hesaplanir
    String expected1 = "7, 8";
    String expected2 = "25, 35";
    String expected3 = "1";
    String expected4 = "5, 10, 16";
    String expected5 = "15";

    assertEquals(bTree.binarySearchInBtree( item: 8).toString(),expected1);
    assertEquals(bTree.binarySearchInBtree( item: 25).toString(),expected2);
    assertEquals(bTree.binarySearchInBtree( item: 1).toString(),expected3);
    assertEquals(bTree.binarySearchInBtree( item: 5).toString(),expected4);
    assertEquals(bTree.binarySearchInBtree( item: 15).toString(),expected5);
```

2- Order'ı 5 olan string veriler tutan btree için "The quick brown fox jump over the lazy dog" cümlesini kelimelerini ağaça eklendi.

```
//string veriler tutan order'ı 5 olan btree

BTree<String> BTree = new BTree<>( order: 5);
BTree.add("The");
BTree.add("quick");
BTree.add("brown");
BTree.add("fox");
BTree.add("jumps");
BTree.add("over");
BTree.add("the");
BTree.add("lazy");
BTree.add("dog");

//beklenen degerler
String expectedS1 = "The, brown, dog";
String expectedS2 = "quick, the";
String expectedS3 = "fox, over";
String expectedS4 = "jumps, lazy";

assertEquals(BTree.binarySearchInBtree( item: "The").toString(),expectedS1);
assertEquals(BTree.binarySearchInBtree( item: "the").toString(),expectedS2);
assertEquals(BTree.binarySearchInBtree( item: "fox").toString(),expectedS3);
assertEquals(BTree.binarySearchInBtree( item: "lazy").toString(),expectedS4);
```

3- Btree de olmayan bir item search edilirse null return etmesi durumunu kontrol ettiğim test case'i :

```
// olmayan elemanı arayalım, null return etmeli

assertEquals(BTree.binarySearchInBtree( item: "Batuhan"), actual: null);
```

3 Project 9.5 in book

*Bu kısımda avl tree'nin methodlarını tamamlayıp binartTree alan constructor'ı yazdım remove methodunu implement etmedim.Vakit sıkıntısı yaşadığım için raporda açık olmayan kısımlar kalmış olabilir ama muhtemelen kode da bu açıkların cevapları vardır.

3.1 Problem Solution Approach

Avl tree binarSearchWithRotate 'den extends oluyor . Avltree'nin rebalancerRight , incrementBalance methodlarını implement ederek doğru bir eleman eklemesini sağladım . HighOfTree , isAVL , getMin ,getMax ,PostOrder gibi yardımcı methodlar tanımladım. Ardından parametre olarak BinaryTree alan isAvl methodunun kullanarak parametre olarak binaryTree alan bir constructor tanımladım .Bu constructor parametre olarak gelen tree nin balance'lık durumunu kontrol ediyor eğer avl tree'nin balancelık durumunu sağlıyorsa Ekrana bunun bir avl tree olduğu yönünde mesaj bırakıyor(ödev pdf'inde bu kısım açıktı ben de bu şekilde bıraktım) eğer gelen ağaçın balance'lık durumu avl tree olma koşulunu sağlamayacak durumda ise

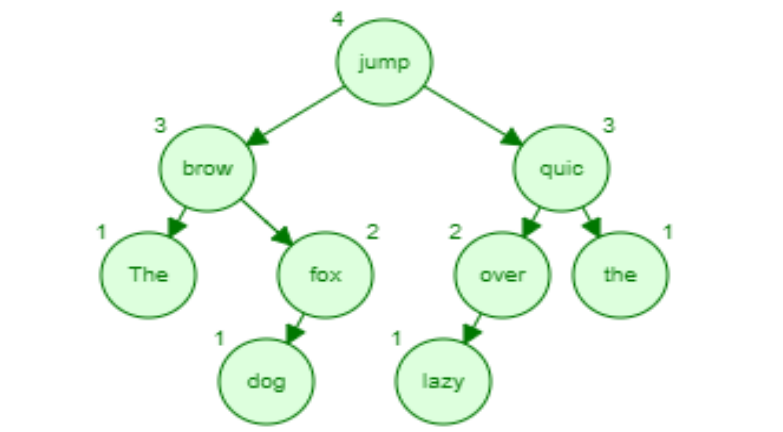
constructor exception fırlatıyor.

3.2 Test Cases

“The , quick , brown ,fox , jump ,over , the ,lazy, dog” ifadelerini bu sıralama ile tree ye ekleyince benim yazdığım kodun çıktısı :

```
"C:\Program Files\Java\
0: jump
  1: brown
    0: The
      null
      null
    -1: fox
      0: dog
        null
        null
      null
  -1: quick
    -1: over
      0: lazy
        null
        null
      null
    0: the
      null
      null
```

Benzetim programının çıktısı :



Görüldüğü gibi çıktılar birebir örtüşüyor.

BinaryTree şeklinde parametre alıp onun avl tree olup olmadığını kontrol eden konstructor' a iki durum için test ettim ilk olarak önceden oluşturulup eleman eklenmiş olan bir avltree 'yi yeni avl tre nin constructor'ına parametre olarak göndererek , ikinci ise bir avltree balance'ından uzak bir binarySearch Tree'yi constructor'a göndererek yaptım.

3.3 Running Commands and Results

Constructor'ın iki farklı durum için testleri :

Bu avlTree'yi yeni avl tree <isAvle> objesi oluştururken parametre olarak gönderiyorum .

```
"C:\Program File
0: jump
  1: brown
    0: The
      null
      null
    -1: fox
      0: dog
        null
        null
      null
  -1: quick
    -1: over
      0: lazy
        null
        null
      null
    0: the
      null
      null
```

Bir bir binarySearch tree objesi olan aşşağıdaki son derece dengesi tree'yi avl tree objesi olan <isAvle2> 'yi oluştururken parametre olarak gönderiyorum.

```
The
  null
  quick
    dog
      null
      fox
        null
        jump
          null
          null
    null
```

>> Bu iki testin sonuçunda ilk durum için bunun geçerli tree olduğunu belirten mesajı , ikinci durum için ise bunun bir avl tree olamayacağı yönündeki görmeyi bekliyorum . Ana test kodu ve çıktılar aşşağıda görülüyor. Sonuçlar beklentiler ile uyuşuyor constructor doğru çalışıyor.

```
    public static void main(String[] args) {

        AVLTree<String> avlTree = new AVLTree<>();
        avlTree.add("The");
        avlTree.add("quick");
        avlTree.add("brown");
        avlTree.add("fox");
        avlTree.add("jump");
        avlTree.add("over");
        avlTree.add("the");
        avlTree.add("lazy");
        avlTree.add("dog");

        BinaryTree<String> check1= avlTree;

        BinarySearchTree<String> bst = new BinarySearchTreeWithRotate<>();
        bst.add("The");
        bst.add("quick");
        bst.add("dog");
        bst.add("fox");
        bst.add("jump");

        BinaryTree<String> check2= bst;

        try {

            AVLTree<String> isAvle = new AVLTree<>(check1);

            AVLTree<String> isAvle2 = new AVLTree<>(check2);
        }
        catch (Exception E){
            System.out.println(E.getMessage());
        }
    }
```

AVLTree › main()

```
VLTree
"C:\Program Files\Java\jdk1.8.0_151\bin\java" ...
Bu bir avl tree
bu avl tree olamaz
```