**Gebze Technical University**

**Computer Engineering**

**CSE 222 - 2018 Spring**

**HOMEWORK 6 REPORT**

**ZEYNEB AKDOĞAN**

**181041038**

Course Assistant: Fatma Nur Esirci

# Worst RedBlack Tree

RedBlackTree.java sınıfında red black binary search tree implement edildi. MoveBlack Down metodu iki kırmızının alt alta gelmeyeceği kuralını yerine getiren bir metotdur. Roota ve çocuklarına bakıp kırmızı olmaları durumunda ikisinin de siyaha çeviren metotdur. Aynı zamanda rotate left metodu da gerekli olduğu durumlarda ağacın dengelenmesin sağlayan metotdur. Koddaki 85-118 arasındaki kodlar itemın local roottan küçük ya da büyük olması durumunda çalışır. Bu ağacın worst case durumu ağaca sıralı değerlerin konulmasıdır. s

## Problem Solution Approach

Write pseudocode and explanation about code design. Indicate what you are using that interfaces, classes, structures, etc.

## Test Cases:

TEST 1:

Black: 23

Red : 15

Black: 11

Red : 7

Black: 5

Red : 3

Black: 2

Red : 1

null

null

null

Black: 4

null

null

Black: 6

null

null

Black: 9

Black: 8

null

null

Black: 10

null

null

Black: 13

Black: 12

null

null

Black: 14

null

null

Black: 19

Black: 17

Black: 16

null

null

Black: 18

null

null

Black: 21

Black: 20

null

null

Black: 22

null

null

Black: 27

Black: 25

Black: 24

null

null

Black: 26

null

null

Black: 29

Black: 28

null

null

Black: 30

null

null

TEST 2:

Black: 13

Black: 9

Black: 5

Red : 3

Black: 2

Red : 1

null

null

null

Black: 4

null

null

Red : 7

Black: 6

null

null

Black: 8

null

null

Black: 11

Black: 10

null

null

Black: 12

null

null

Black: 17

Black: 15

Black: 14

null

null

Black: 16

null

null

Black: 19

Black: 18

null

null

Black: 20

null

null

## Running Commands and Results

Sadece sayılar arttırılınca doğru cevabı bulduk. Sayılar artarak verilince ağaç konu mantıken aynı olmasına rağmen hatalı oldu. Sadece azalan sayılarda doğru cevabı buluyorum. s

# binarySearch method

BTree.java classı binary search metod ile yapıldı. Itemın datası 1.den küçük mü sonuncudan büyük mü aradakine binary tree uygulandı.

medium = (left + right) / 2; orta index bulunur

Eğer aradığım değerle aynıysa geriye döndürüyor. Büyükse linki küçükse rightu değiştiriyor.

yarılama yöntemiyle istenen indexi buluyor.

int index = binarySearch(item, root.data, 0, root.size);

ben bunu nereye insert edeyim. İnsert edilecek konumu bulur.

değerler hangi aralığa yerleşsin. childlar dolunca yukarıya ittiriliyor.

o da dolduysa root değişken oluyor.

## Problem Solution Approach

BTree.java classı binary search metod ile yapıldı. Itemın datası 1.den küçük mü sonuncudan büyük mü aradakine binary tree uygulandı.

medium = (left + right) / 2; orta index bulunur

Eğer aradığım değerle aynıysa geriye döndürüyor. Büyükse linki küçükse rightu değiştiriyor.

yarılama yöntemiyle istenen indexi buluyor.

int index = binarySearch(item, root.data, 0, root.size);

ben bunu nereye insert edeyim. İnsert edilecek konumu bulur.

değerler hangi aralığa yerleşsin. childlar dolunca yukarıya ittiriliyor.

o da dolduysa root değişken oluyor.

BTest Cases 7, 72

18, 33

0

null

null

20, 21, 30

null

null

null

null

36

null

null

52

45

null

null

58

null

null

81, 90

74, 76

null

null

null

84

null

null

92, 98

null

null

null

## Running Commands and Results

Sayılar rastgele seçildi.