**Gebze Technical University**

**ComputerEngineering**

**CSE 222 -2018 Spring**

**HOMEWORK 6 REPORT**

**BURAK ÖZDEMİR**

**141044027**

Course Assistant: Fatma Nur Esirci

# WorstRedBlackTree

## Problem Solution Approach

RedBlackTree ve RedBlackNode sınıfları kullanılmıştır.6 yüksekliğindeki RedBlackTree ağacı worst case durumunda en az 14 node ile gerçeklenebilir . Worst Case durumu agacı yüksekliği ile alakalı oldugu icin her durumda O(logn) karmasıklıgındadır . Yükseklikte normal Binary agactaki gibi hesaplanır .

## Test Cases

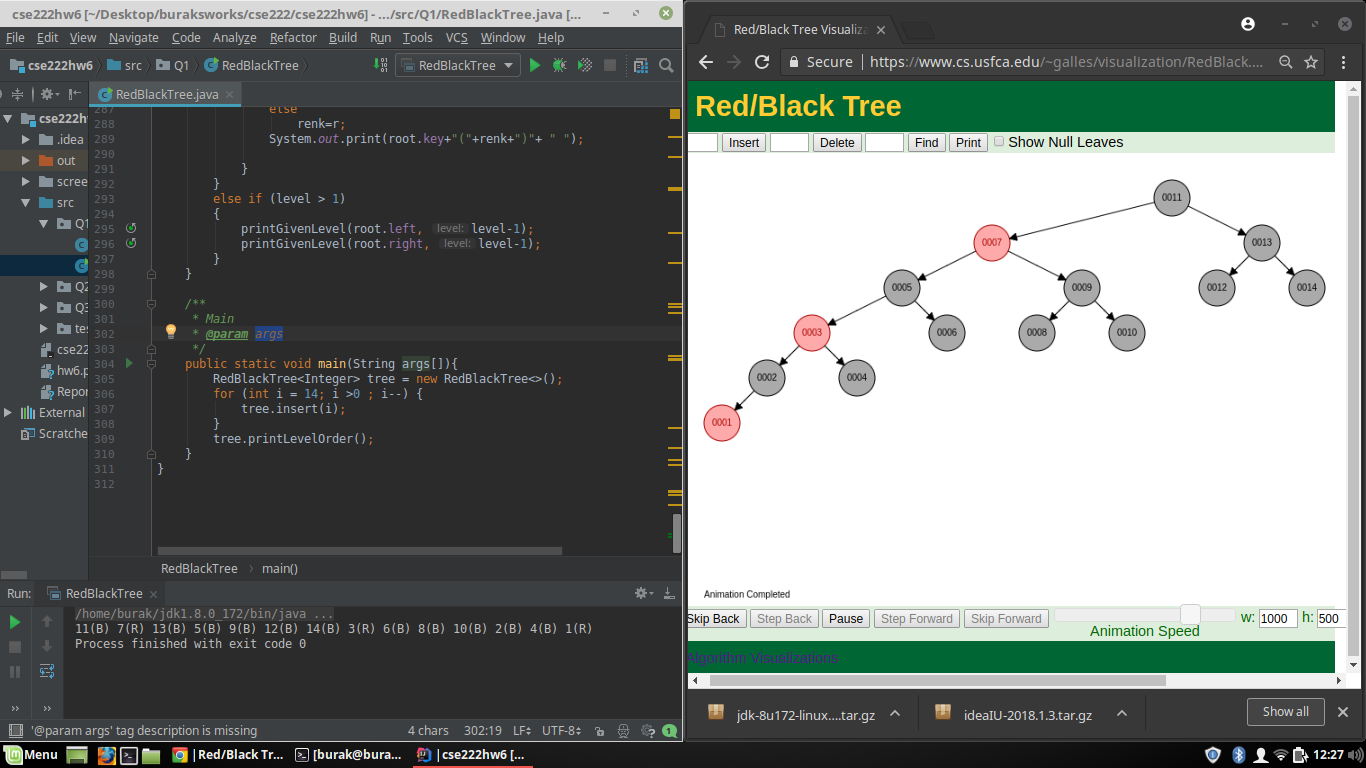
FirstTree:

1 den 14 e kadar nodelar sırasıyla eklenmiştir . Agac RedBlackTree genel kurallarına uyuyor.Visualization sitesindede denenmiştir. Yükseklik 6 dır. Worst Case durumunda yine O(logn) karmasıklıgı vardır.

SecondTree:

14 den 1 e kadar nodelar sırasıyla eklenmiştir . Agac RedBlackTree genel kurallarına uyuyor.Visualization sitesindede denenmiştir. Yükseklik 6 dır. Worst Case durumunda yine O(logn) karmasıklıgı vardır.

## RunningCommandsandResults



Q1 Main Test

# binarySearchmethod

## Problem Solution Approach

Bu method 2 helper metodla calısır . İlk helper method Node un elemanlarının icinde binarysearch yaparak işlemi gerçekleştirir.Eğer bulursa return edip işlemi sonlandırır . Eğer bulamazsa Nodu alt cocuklarının ilk elemanları ile karşılaştırır . Yani cocuklarında hepsine bakmıyarak yine logn karmasıklıgında bır ıslem yaparak alt nodelara gecer ve tekrar ilk method cagırılır Bu sekılde devam eder .

-private boolean binarySearchInKeys(Bnode node ,int low,int high){

İf high is bigger than low

İnitiliaze mid and assign (low +(high-low)/2)

İf key of node[mid] is equal to data

Return true

İf key of node[mid] is smaller than data

Return call binarySearchInKeys(node,mid+1,high,data)

Else

Return call binarySearchInKeys(node,low,mid-1,data)

Return false

}

-private boolean binarySearchInChilds(Bnode root,int data){

Initialize res and assign call BinarySearchInKeys(root,0,root.count-1,data)

İf res is equal to true

Return true

İnitialize newChildNode and assign to -1

Inıtializee i and assing to zero

Inıtialize temp and ssign to root

While child of temp[i] is not equal to null

İf child of temp[i] is not equal to null

İf child of temp[i+1] is not equal to null

İf key of child of temp[i][0] is smaller than data

And key of child of temp[i+1][0] is bigger than data

Assign i to newChildNode

Break

Else

Assing i to newChildOne

İ plus plus

İf newchildNode is equal to -1

Return false

Return call binarySearchInChilds(root.getChild(newChildNode),data)

}

+public boolean binarSearch(int data){

Return binarySearchInChilds(root,data)

}

## Test Cases

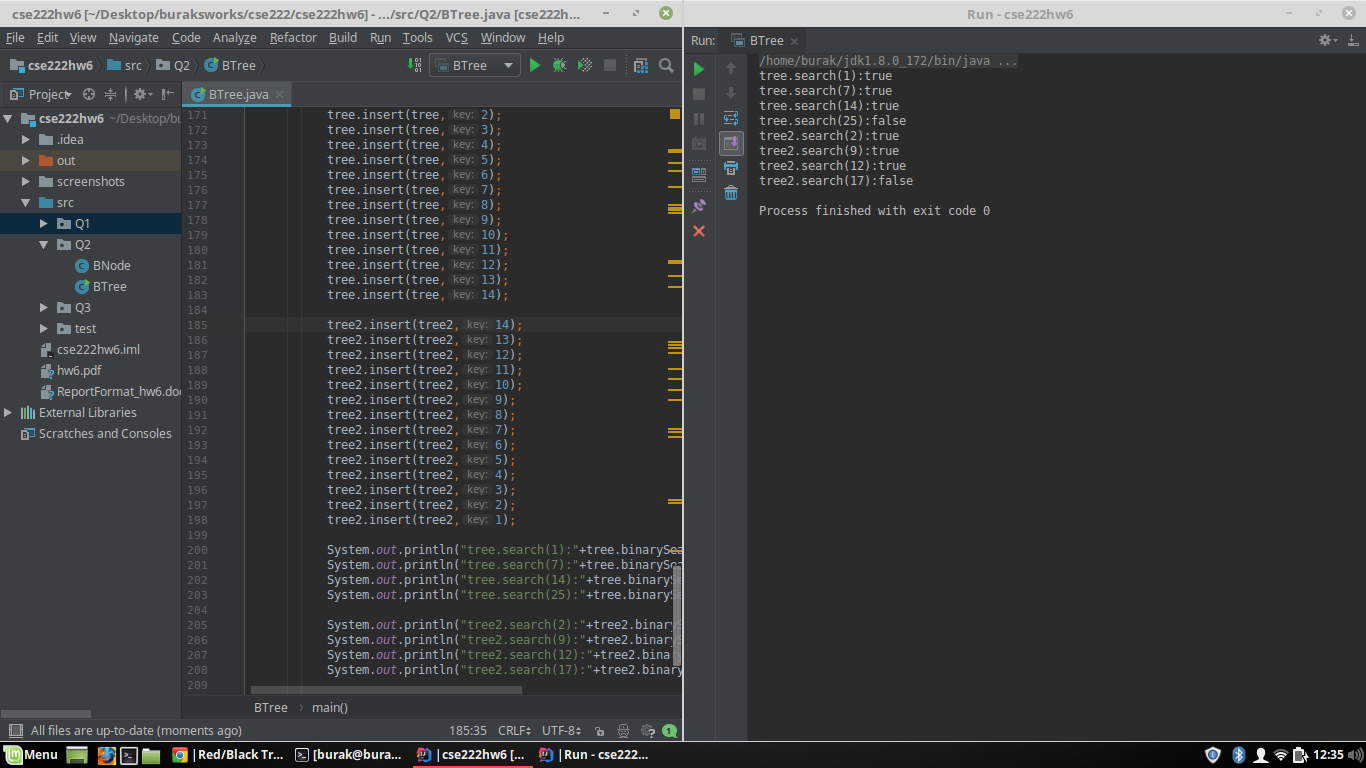
Tree1:

Ağaç 1 den 14 e kadar sayılarla doldurulmustur . Sırasıyla 1 7 14 25 sayıları search edılmıstır .(Basarılı)

Tree2:

Ağaç 14 den 1 e kadar sayılarla doldurulmustur . Sırasıyla 1 7 14 25 sayıları search edılmıstır .(Basarılı)

## RunningCommandsandResults



Q2 Main Test

# Project 9.5 in book

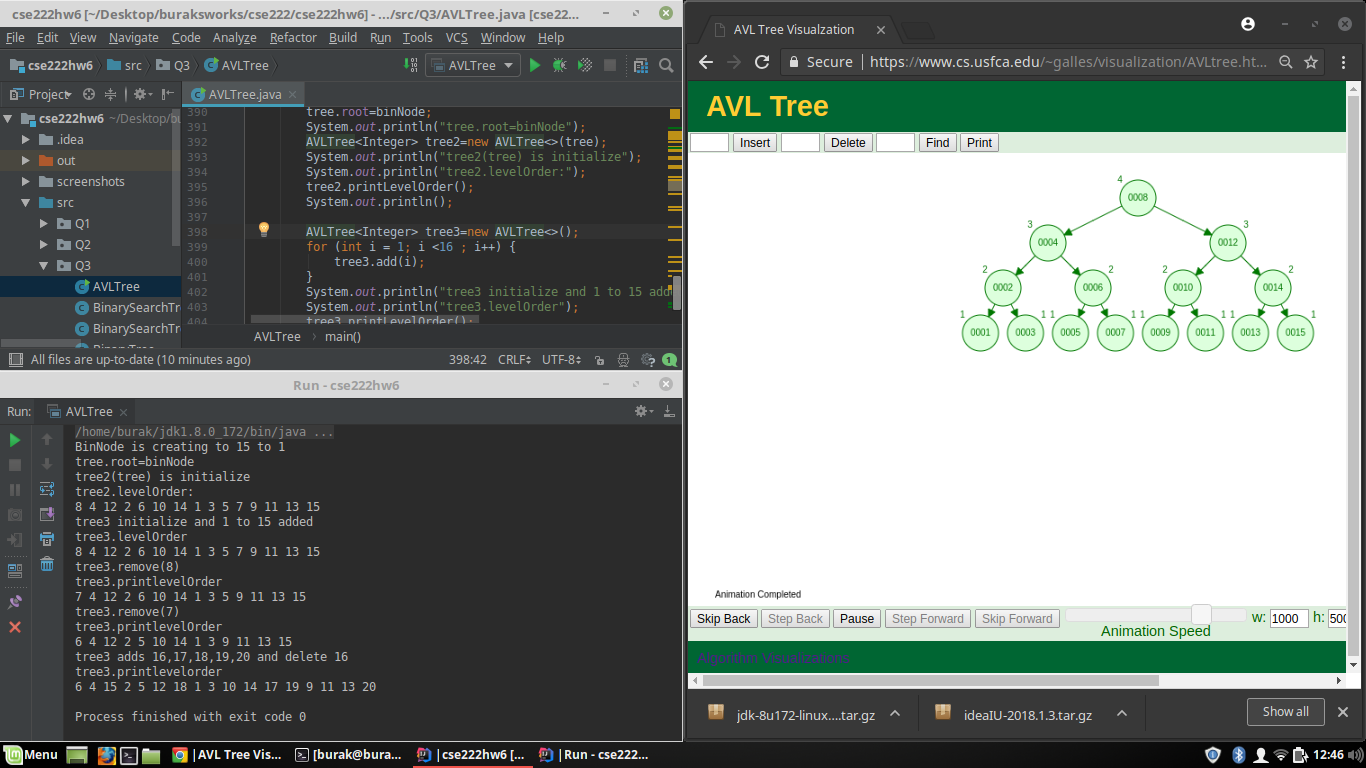
## Problem Solution Approach

AVLTree sınıfı implement edildi.Constructor baksa bir binary tree alıyor ve kendisine gore duzenlıyerek datafieldlarına yerlestrıyor.Gereklı metodlar ımplement edıldı.Binarytree kontrolu constructor ıcınde edılıyor ve sonuc ekrana basılıyor.

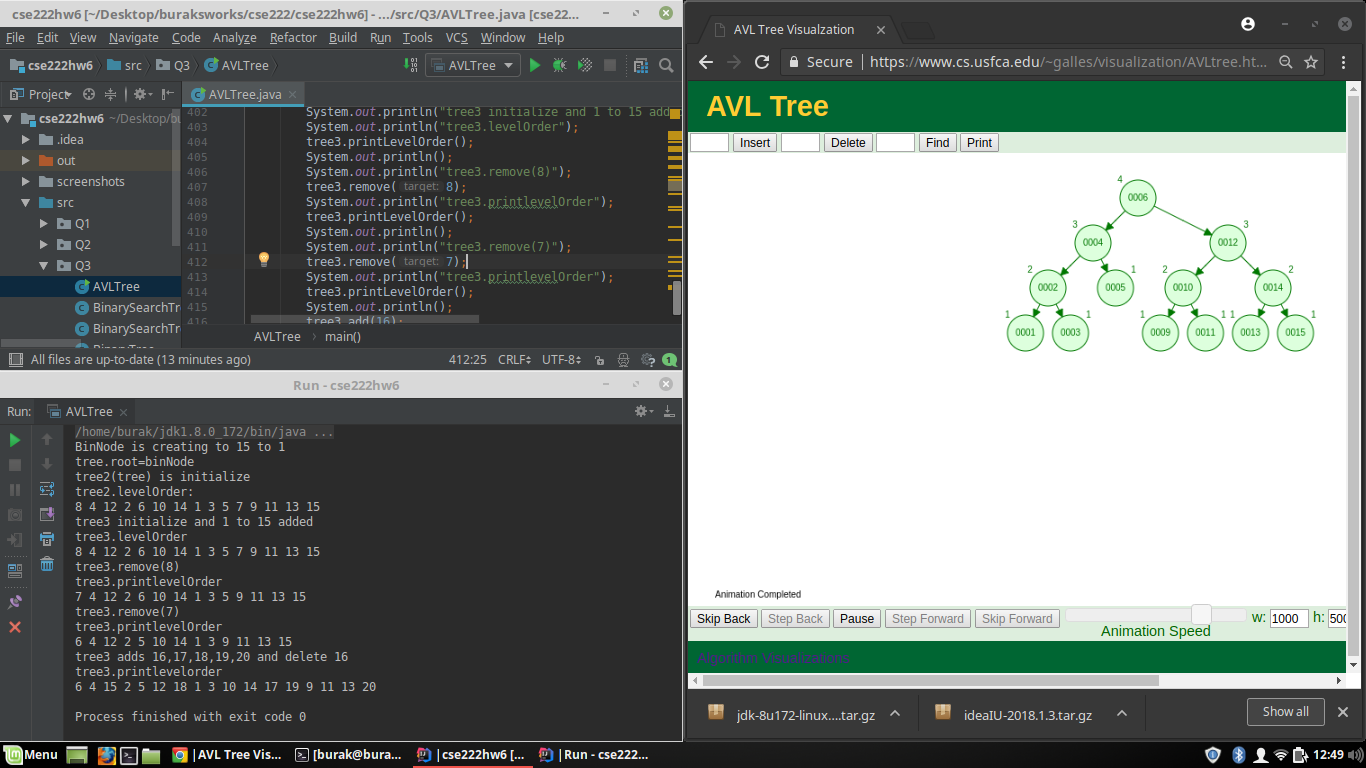
## Test Cases

Tree degiskenıne sureklı sol tarafa dogru gıden bır node yapısı verıldı . Daha sonra bu tree degıskenı tree2 yanı AVLTree yapısına constructor parametresı olarak verıldı . Sınıf bu yapıyı duzenlıyerek kendısıne uygun hale getırdi . Daha sonra tree3 degıskenı ıle sınıfın add ve remove metodları test edıldı .

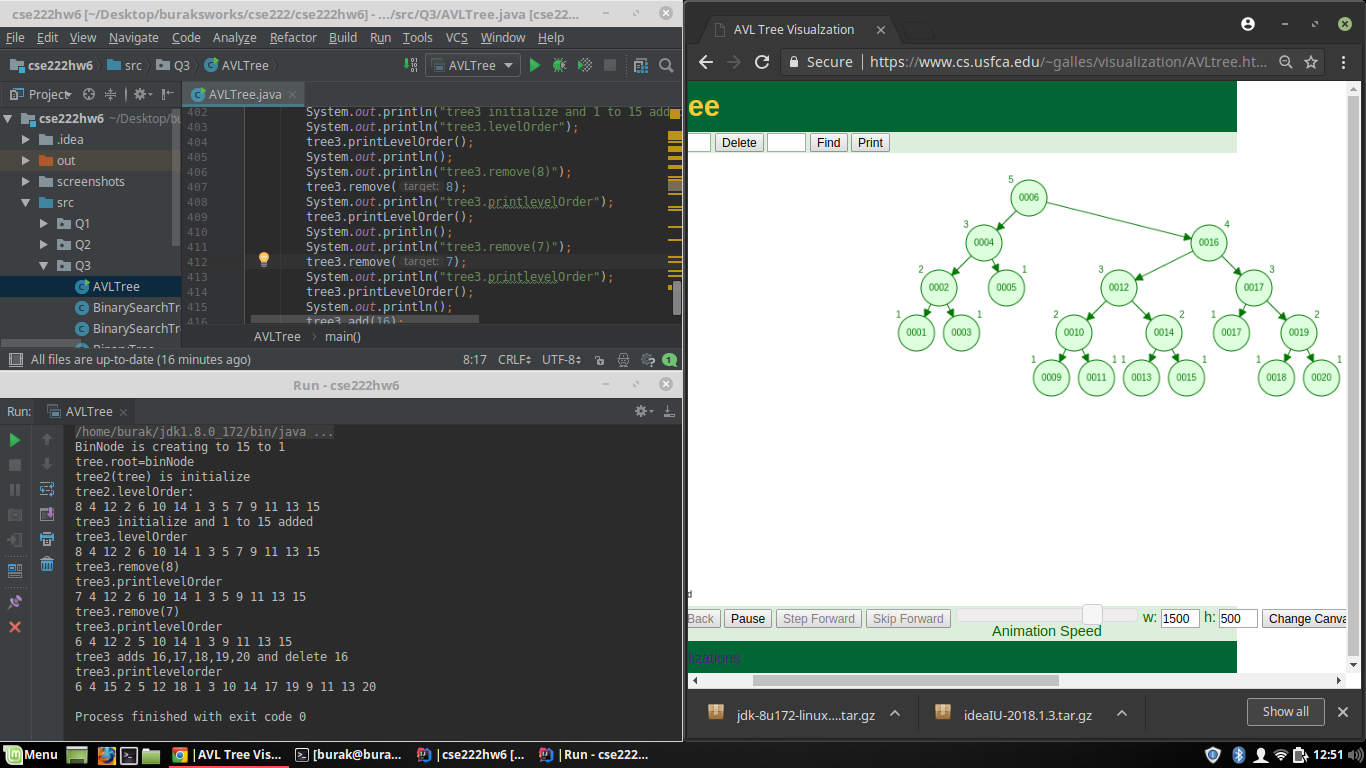
## RunningCommandsandResults



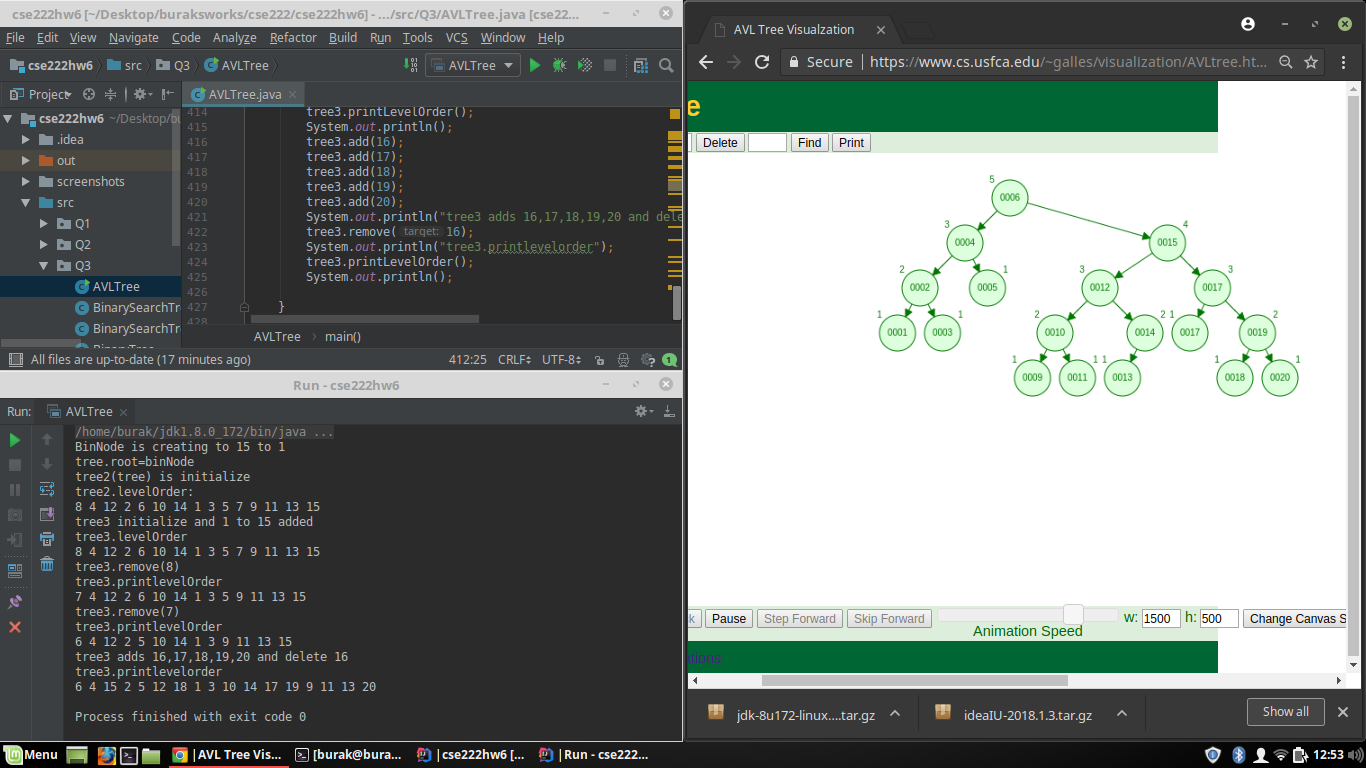
Q3 Main Test(1 den 15 e kadar ekleme)



Q3 Main Test(7 and 8 deleted)



Q3 Main Test(from 16 to 20 added)



Q3 Main Test(deleted 16)