Gebze Technical University
Computer Engineering

CSE 222 - 2018 Spring

HOMEWORK 04 REPORT

DENİZ BABAT
131044013

İçindekiler

Şekil Tablosu

1 PART1

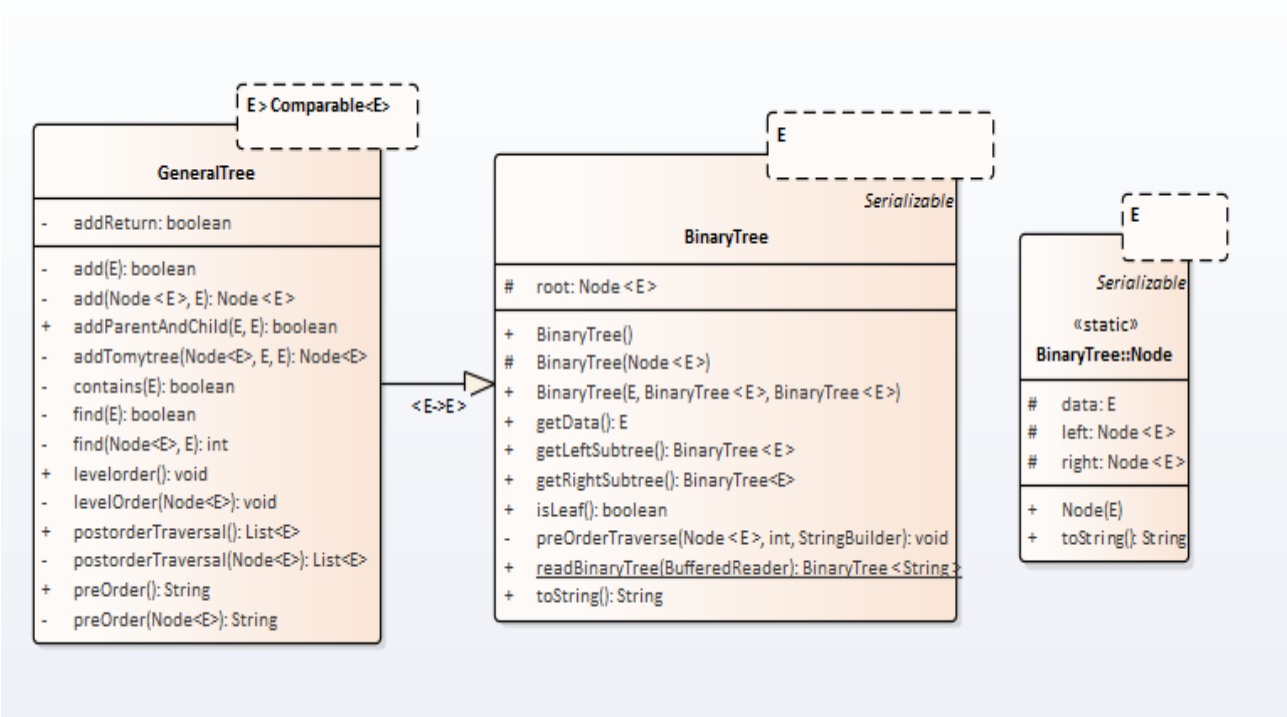
1.1 GİRİŞ

1.1.1 PROBLEM TANIMI:

Problem Binary Tree'ye iki ilişkili iki eleman eklemektir. Bu ilişki parent ve child ilişkisidir. Eklenecek parent ve child koşulu şöyledir; Eğer tree de parent varsa child eklenip return değeri true olarak dönecektir. Eğer parent yoksa return değeri false olarak dönecektir. İlişki bundan ibarettir. Bu problemin çözülmesi için GenerelTree adında bir class oluşturdum ve metod formatları ve açıklaması aşağıdaki gibidir.

- **public boolean addParentAndChild(E parent, E child):** bu metodun bir parent bir de child parametresi alarak child'ı ağaca ekler ya da eklemez. Parent'ı olmayan child'lar için dalse return eder ve ekrana basar. Metod parent'ı bulmak için tüm ağacı dolaşır ve parent'ı bulduğunda ise; alt ağacı recursive olan başka bir fonksiyona parametre olarak verip çocuğu ağaca ekler. Bu yüzden T(n) = O(nlog(n)) dir.
- **public List<E> postorderTraversal():** Bu metod mevcut ağaçtaki dataları postOrder sıralamaya göre sıralar. Tüm ağacı dolaştığı için T(n) = O(n)dir.
- **public void levelorder():** Bu metod general tree ye göre ağaçtaki dataları level order'a göre ekrana basar yani search yapar. Tüm ağacı dolaştığı için T(n) = O(n)dir.
- **public String preOrder():** Bu metod ağaçtaki dataları preorder'a göre ekrana bir string generate eder. Tüm ağacı dolaştığı için T(n) = O(n)dir.

1.1.2 CLASS DİAGRAMI:



Part 1 class diagramı olarak şekil 1 de gözükmektedir. Yukarıda açıklaması yapılmıştır.

Şekil 1: Part1 Class Diagramı

1.1.3 PROBLEME YAKLAŞIM

Problemi çözmek için ilk olarak null olan ağaıcın rootuna ilk gelen parentı ve child ı ekledim. Daha Sonra eklenecek child'ın parent'ı ağaçta var mı yok mu diye aratıyorum. Yoksa false return ediyorum. Eğer varsa; çocuğun parent'ını bulup altındaki ağaç'ın (parent'ında leaf'i olabilir, Leaf yoksa ağacın leafine ekler.) çocuğu ekliyorum.

1.1.4 SONUÇ

```
Add kamil kaya: true
deniz fatih ahmet kamil pasa kaya dede
Add fatih kazım: true
deniz fatih kazım ahmet kamil pasa kaya dede
Add fatih berk: true
deniz fatih kazım berk ahmet kamil pasa kaya dede
kazım mehmet: true
deniz fatih kazım mehmet berk ahmet kamil pasa kaya dede
berk einstein: true
deniz fatih kazım mehmet berk einstein ahmet kamil pasa kaya dede
einstein kane: true
deniz fatih kazım mehmet berk einstein kane ahmet kamil pasa kaya dede
Add einstein aslan: true
deniz fatih kazım mehmet berk einstein kane aslan ahmet kamil pasa kaya dede
fatih ahmet: false
deniz fatih kazım mehmet berk einstein kane aslan ahmet kamil pasa kaya dede


                                    deniz

                  ahmet                                         fatih
         null                kamil                   berk                 kazım
    null      null      dede      pasa      null      einstein      null      mehmet
  null  null  null  null  null  null  kaya  null  null  null  aslan  kane  null  null  null  null


====LEVELORDER TRAVERSAL====
deniz,ahmet,kamil,dede,pasa,kaya,fatih,berk,einstein,aslan,kane,kazım,mehmet,
====PREORDER TRAVERSAL====
deniz fatih kazım mehmet berk einstein kane aslan ahmet kamil pasa kaya dede

====POSTORDER TRAVERSAL====
dede kaya pasa kamil ahmet aslan kane einstein berk mehmet kazım fatih deniz
```

Şekil 2: Part-1 sonuç

2 PART 2

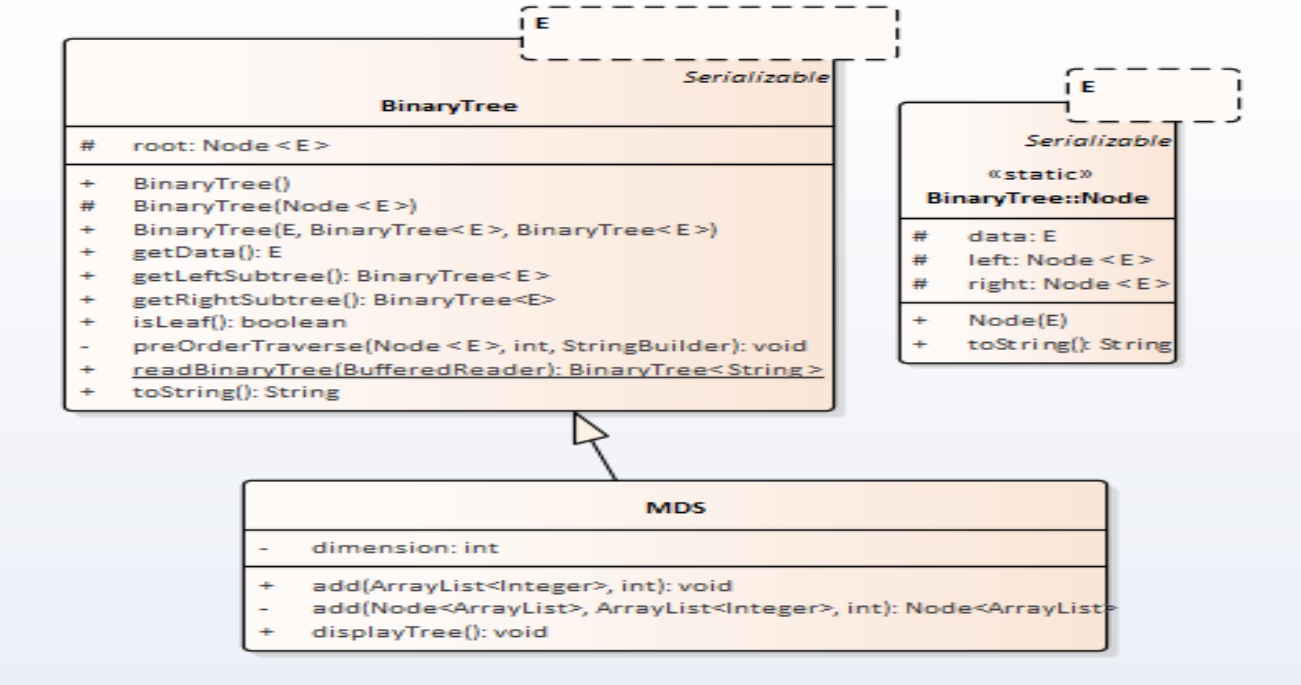
2.1 GİRİŞ

2.1.1 PROBLEMİN TANIMI

Hiper düzlem üzerindeki multidimensional noktalar düzlem eksenlerine göre ağaca eklenir. Root her zaman X eksini ile başlar ve ağaç y, z, k ... x, y … vs. olarak devam eder. Bu problem çözümü için MDS class'ı oluşturulmuştur. Metodları aşağıdaki gibidir.

- **public void add(ArrayList<Integer> points, int dimens):** Bu metod iki parametre alır ilk parametre değerlerden oluşan koordinat düzlemi, ikinci parametre kaç boyutlu bir düzlem olduğu ile ilgilidir. Bu metod eksenleri karşılaştırdığı için her geçtiği düzlemi compare eder. Bu yüzden normal ağaca add methodu gibi çalışır. Bu yüzden T(N) = O(log(n)) dir.

2.1.2 CLASS DİAGRAMI



Şekil 3: Part2 Class diagramı

2.1.3 PROBLEME YAKLAŞIM

Ağaca eklencek düzlemi ilk olarak kaç boyutta olaması için fonksiyona boyut girilmesi için parametre koydum. Bu paramtre yi her recursive kolda bir azaltıyorum. Böylece her düzlemi düzgün bir şekilde karşılaştırma yapabiliyorum. Eğer bu parametre 0 olursa tekrar parametre ilk halini recursive kola veriyorum, böylece düzlem noktasının ağacın leaf'ine eklenmesi için düzlemin eksenlerini tekrarlamasını sağlıyorum ve her ekseni kendi içinde karşılaştırıyorum.

2.1.4 SONUÇ

```
PART 2

                              [2, 5, 4]
                [0, 5, 4]                                   [3, 1, 0]
        null                  [1, 56, 24]                 [30, 0, 20]                 [8, 4, 41]
    null      null      null      null      null      null      [7, 56, 24]      null

Process finished with exit code 0
```

Şekil 4: Part-2 sonuç