Gebze Technical University
Computer Engineering

CSE 222 - 2018 Spring

HOMEWORK 4 REPORT

SERKAN SORMAN
151044057

Course Assistant: Mehmet Burak Koca

1 INTRODUCTION

1.1 Problem Definition

Part 1

Generic yapıdaki General bir treedeki elemanların binary bir treede depolanması ve bu treeye eleman eklenirken childların parent nodun soluna, siblinglerin ise parent nodun sağına eklenmesi. Tüm bu eklenen elemanlar üzerinde, General tree formunda level order, post order arama işlemlerinin yapılabilmesi ve tree üzerinde pre order olarak gezildiğindeki string halinin elde edilmesi.

Part 2

Birden fazla boyuta sahip real typedeki elemanlardan oluşan Multi dimensional BST nin tasarlanması. Bu treeye ekleme işlemi yapılırken her levelde bir sonraki boyuta göre karşılaştırma yapılıp mevcut binary search ekleme düzenine göre eklenecek elemanın üzerinde gezilen nodedan büyük değere sahipse sağ, küçükse sol tarafına eklenmesi. Implement edilen SearchTree interfacenin tüm methodlarının bu treeye uygun olarak implement edilmesi.

1.2 System Requirements

Part 1

General treenin extend edileceği bir BinaryTree class yapısına ihtiyaç duyuldu. Level order, post order search işlemleri sırasında traverse yapılırken üstünden geçilen nodeların belirtilmesi için String builder yapısı kullanıldı ve her node bu yapıya eklendi. Search işlemleri sırasında bulunan eleman direkt return edildiği için aranan elemana kadar olan elemanlar Stringe eklendi. (Tamamen level order veya post order sıralamanın elde edilmesi için son eleman veya treede olmayan bir elemanın search edilmesi gerekir). Searchler sırasında nodeların tutulması için Queue ve Stack veri yapıları kullanıldı.İlk add işlemi sırasında tree boş ise paren root yapılır ve child soluna eklenir. . Methodlar implement edilirken recursive çağrının yapıldığı overload edilmiş methodlar kullanıldı.

Part 2

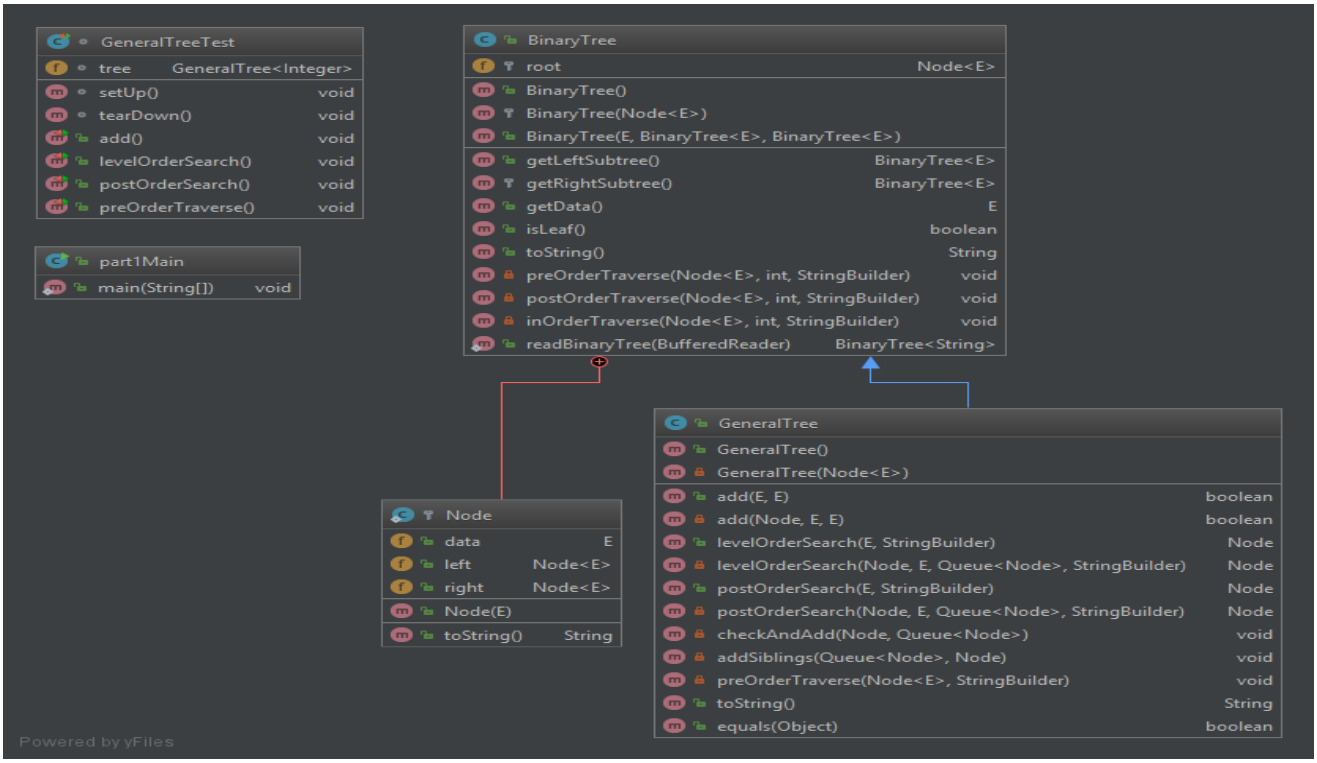
MDS treenin extend edildiği BinaryTree class yapısı ve implement edildiği SearchTree İnterfacesi kullanıldı. Tree deki elemanların dimensionları başlangıçta constructurın türüne göre belirlenebilir ya da default olarak dimension 3 olarak ayarlanır. Pozitif

olmayan dimensionlarda exception fırlatılır. Çok boyutlu eleman özelliği için vector kullanıldı ve elemanlar vektör olarak nodelarda tutuldu. Methodlar implement edilirken recursive çağrının yapıldığı overload edilmiş methodlar kullanıldı.

2 METHOD

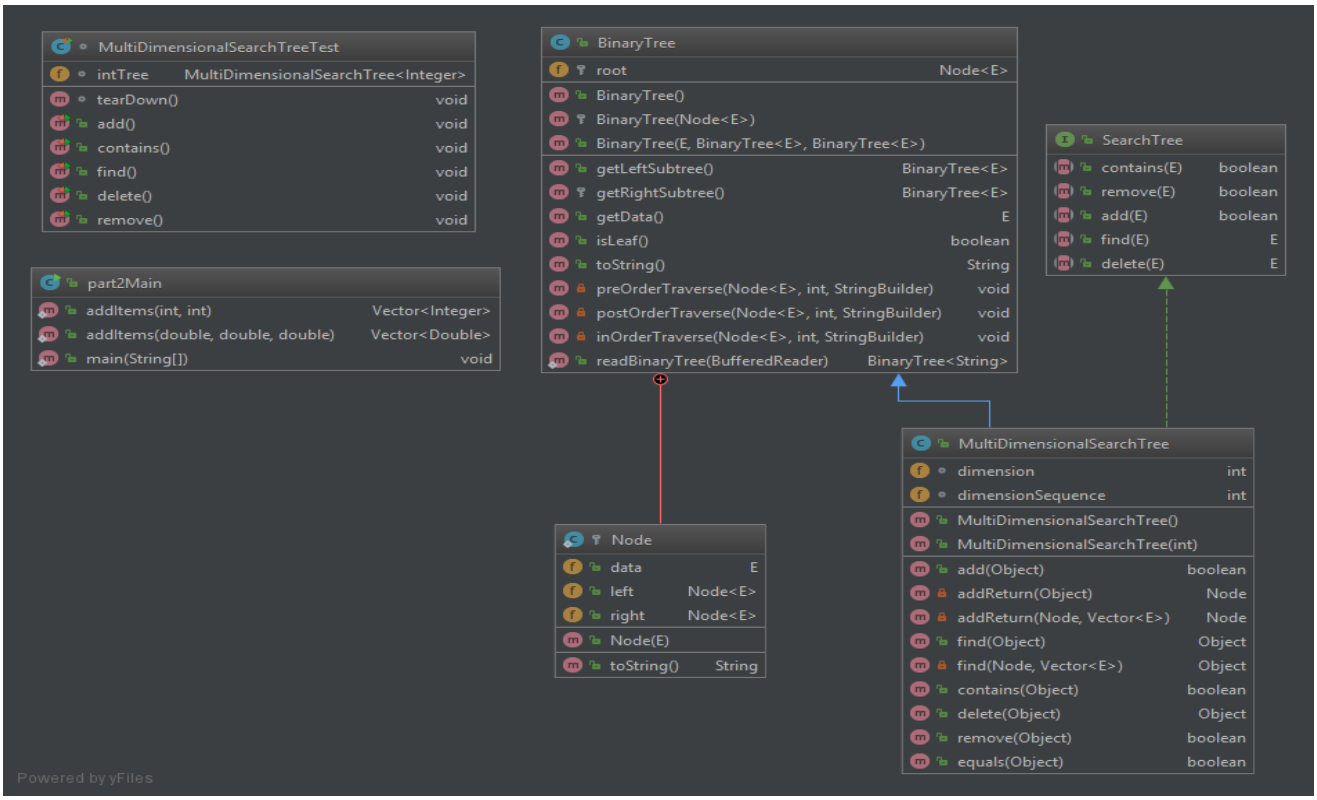
2.1 Class Diagrams

Part 1



- GeneralTree classı BinaryTree classından extend edildi.
- Add metodu ile parametrede verilen parent treede buluyorsa verilen child bu parenta eklendi.
- Level order search metoduyla eleman level order traverse ile arandı ve geçilen nodelar String buildera eklendi.
- Post order search metoduyla eleman post order traverse ile arandı ve geçilen nodelar String buildera eklendi.
- checkAndAdd ve addSiblings gibi methodlar arama methodları içinde yardımcı methodlar olarak kullanıldı.
- Methodların recursive çağrılarının yapıldığı node parametresi içermeyen overloadları tasarlandı.

Part 2



- MDSTree classı BinaryTree classından extend edilip SearchTree classını implement etmiştir.
- dimension fieldında treedeki tüm elemanların sahip olduğu boyut tutulmuştur.
- dimensionSequence fieldında ise her levelde değişen boyut karşılaştırması için dinamik bir counter tutulmuştur.
- Add metodunda AddReturn metodları kullanılarak parametre olarak gelen çok boyutlu eleman olan vector sahip olduğu dimensiona göre treede uygun konuma eklenir.
- Find metodu ile aranan eleman varsa return edilir yoksa null return edilir
- contains metodu içinde find metodu kullanılarak eleman bulunursa true return edilir.
- Delete metodunda silinmek istenen eleman haricindeki diğer elemanlar geçici bir tree ye eklenir ve eski tree silinerek bu tree ana tree yapılır.
- Remove metodu içinde delete metodu kullanılarak eleman başarıyla silindiyse true return edilir.

TIME COMPLEXİTY

Part 1

Add() T(n) = O(n^3)

levelOrderSearch() T(n) = O(n^3)

postOrderSearch() T(n) = O(n^5)

checkAndAnd() T(n) = O(n)

addSiblings() T(n) = O(n^2)

preOrderTraverse() T(n) = O(n)

Part 2

Add() T(n) = O(n)

Find() T(n) = O(n)

Contains() T(n) = O(n)

Delete() T(n) = O(n^2)

Remove() T(n) = O(n^2)

2.2 Use Case Diagrams

2.3 Other Diagrams (optional)

2.4 Problem Solution Approach

Part 1

Genarel tree formunun korunması için parentın childları soluna siblingleri sağına insert edildi. Level order search yapılması için current nodun tüm siblingleri bir Queue ya atıldı Daha sonra Queue deki nodelar poll edilerek childları kontrol edildi ve childları yeni bir Queue ya aktarıldı ve bu işlem recursive call ile tekrarlanarak treede level order bir şekilde gezinilmiş oldu. Post order search için ise current rootun siblingleri bir queuya alındı ve bu queudaki her nodeun tüm childları Stacke atıldı. Daha sonra stackdeki nodelar sondan poll edilerek queuya eklendi. Bu işlem recursive olarak tekrar edilerek nodelar queuya post order bir şekilde atılmış oldu ve en son queu üzerinden post order search yapıldı.

Part 2

Çok boyutlu elemanların oluşturulması için vektör veri yapısı kullanıldı. Generic tipteki E Comparebledan extend edildi. Karşılaştırma sırasındaki boyut değişimlerini tutabilmek için dimensionSequence değişkeni oluşturuldu. Tree ye add işleminin yapılması için sol ve sağa recursive dallanmalar yapıldı her dallanma yapıldığında bir sonraki dimensiona göre karşılaştırma yapılması gerektiğinden dimensionSequence değeri bir sonraki boyuta kaydırıldı ve karşılaştırma diğer boyut üzerinden yapıldı. Her yeni ekleme işlemi bu şekilde bulunulan levele göre dimensionSequence vasıtasıyla boyut karşılaştırılarak yapıldı. Find ve contains

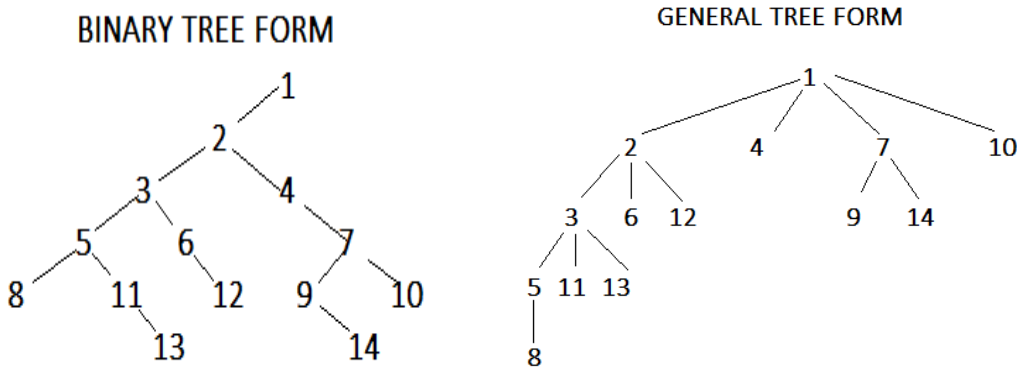
methodlarında recursive bir şekilde eleman arandı. Delete ve remove metodunda silinmek istenen eleman haricindeki diğer elemanlar geçici bir tree ye eklendi ve eski tree silinerek geçici tree ana tree yapılarak devam edildi.

3 RESULT

3.1 Test Cases

3.1.2 Main Testi

Part 1



Tüm methodlar mainde şekilde belirtilen tree üzerinde test edilmiştir.

Add metodu normal ekleme ve olmayan parenta ekleme denemesi yapılarak test edildi. levelOrderSearch metodu treede var olan eleman ve olmayan eleman ile test edildi. String builder yollanarak gezinilen nodelar stringe eklenip ekrana basıldı ve level order traverse doğrulandı. Aynı işlem postOrderSearch için de yapıldı ve gezilen nodelar ekrana basılarak postOrderTraverse doğrulandı. En son ise tree toString metoduyla ekrana basıldı ve pre order traverse gösterildi.

Part 2

Çok boyutlu elemanlar olarak kullanılacak vektöre eleman eklemek üzere örnek teşkil eden 2 ve 3 boyutlu vektör elemanları oluşturulması için helper addItems methodları kullanıldı. iki ve üç boyutlu elemanlardan oluşan treelere elemanlar eklendi ve son halleri ekrana basılarak doğrulandı Daha sonra contains ve find metodları treede olan ve olmayan elemanlar ile test edildi. Treeden remove ve delete metodları ile elemanlar silindi ve treenin son hali ekrana basılarak doğrulandı.

3.1.2 Unit Testler

Part 1

Main testlerindeki ile aynı şekilde add, postOrderSearch, levelOrderSearch ve preOrderTraverse methodları ayrı ayrı test edildi ve ardından gerekirse treenin son hali ekrana basıldı

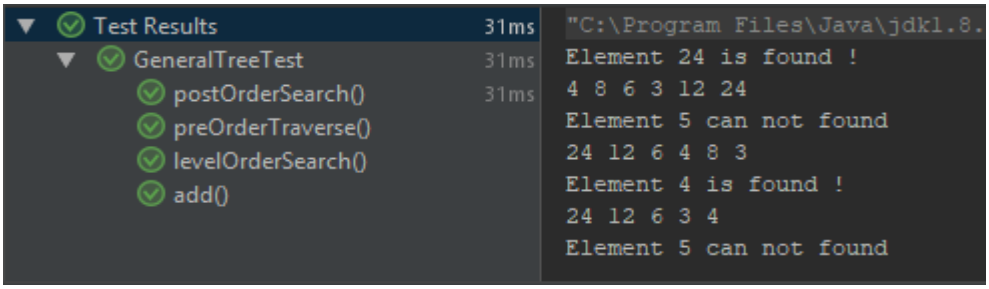
Part 2

Main testlerinden farklı olarak geçersiz dimension ile exception test edilip handle edildi. Add işlemlerinden sonra elemanların yerinin doğrulanması için ekrana basıldı ve doğrulandı.Tree de var olan elemanın tekrar eklenmesi test edildi.

3.2 Running Results

Part 1

```
Root and child added
Child added
Child added
Child added
Child added
Child added
Child added
Child added
Child added
Child added
Child added
Child added
Child added
Child added
Parent can not found
############ Level Order Traverse for 6 ############
1 2 4 7 10 3 6
############ Full Level Order Traverse ############
1 2 4 7 10 3 6 12 9 14 5 11 13 8
############ Full Post Order Traverse ############
8 5 11 13 3 6 12 2 4 9 14 7 10 1
```



```
############   Pre order Traverse   #############
1
  2
    3
      5
        8
          null
        null
      11
        null
      13
        null
      null
    6
      null
    12
      null
    null
  4
    null
  7
    9
      null
    14
      null
    null
  10
    null
  null
null
```

Part 2

```
############ Double Three Dimension Tree #############
[40.0, 45.0, 30.0]
  [15.0, 72.2, 24.8]
    null
    [18.0, 72.4, 26.7]
      [20.0, 80.0, 25.0]
        null
        null
      null
  [70.0, 10.0, 18.0]
    [66.0, 9.8, 29.0]
      null
      [52.0, 5.5, 29.5]
        null
        null
    [85.0, 90.0, 12.0]
      null
      null

############ Integer Two Dimension Tree #############
[40, 45]
  [15, 70]
    null
    null
  [70, 10]
    null
    [69, 50]
      [66, 85]
        null
        null
      [85, 90]
        null
        null
```

```
Check (66,85): [66, 85]
Check (99,99): null
Deleted Node: [15, 70]
############ Integer Two Dimension Tree After (15,70) deleted #############
[40, 45]
  null
  [70, 10]
    null
    [69, 50]
      [66, 85]
        null
        null
      [85, 90]
        null
        null
############ Integer Two Dimension Tree After Root (40,45) deleted #############
[70, 10]
  [69, 50]
    null
    [66, 85]
      null
      null
  [85, 90]
    null
    null
```