Gebze Technical University
Computer Engineering

CSE 222 - 2018 Spring

HOMEWORK 4 REPORT

İslam Göktan SELÇUK
141044071

Course Assistant: Mehmet Burak KOCA

1 INTRODUCTION

1.0 Problem Definition

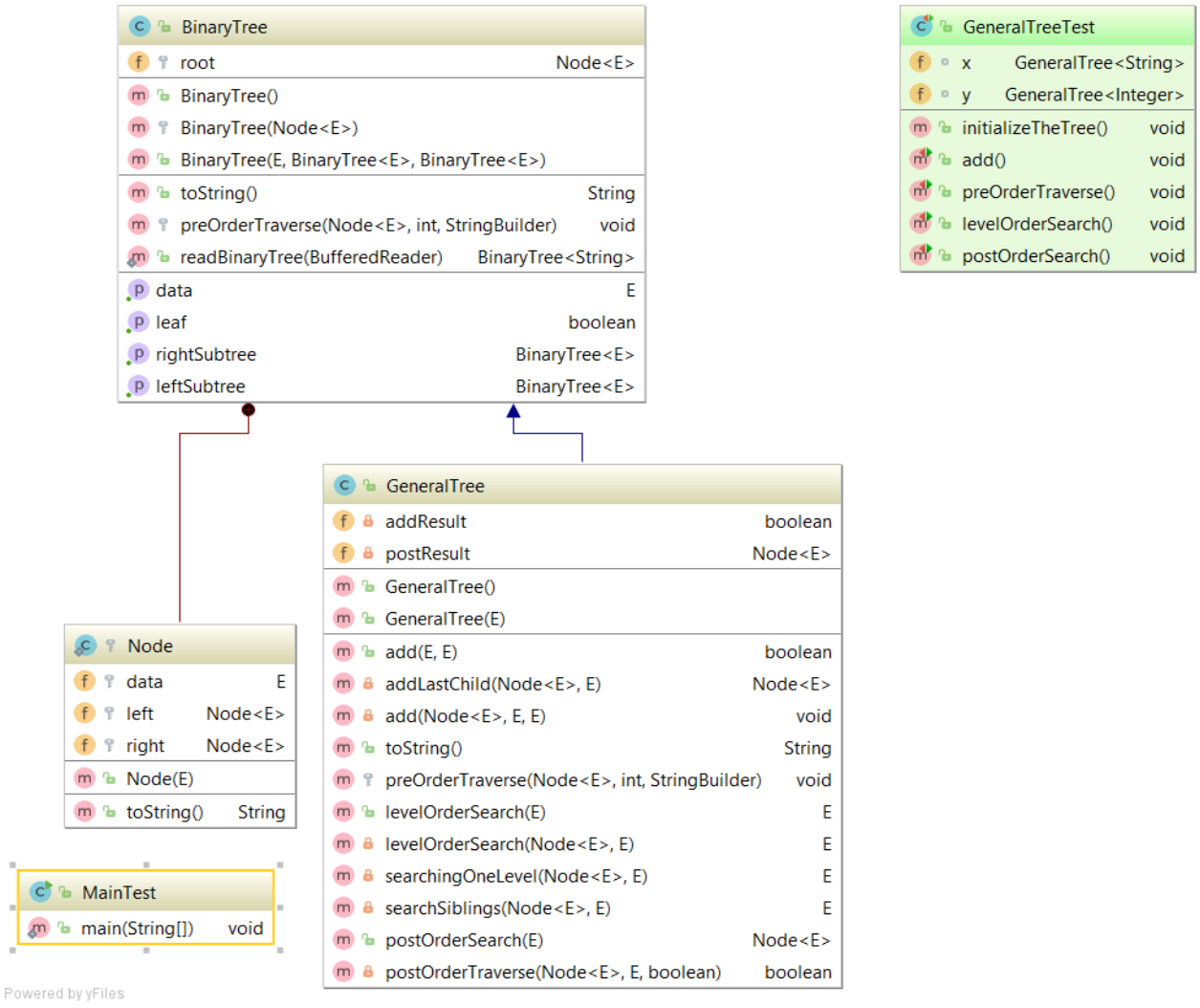
BinaryTree sınıfından faydalanarak general tree yapısını olusturmak. General tree içerisinde preOrderTraverse, postOrderTraverse ve levelOrderTraverse yöntemlerini kullanarak ağaç yapısı üzerinde gezinmek, ağaç üzerinde arama yapmak.

1.1 System Requirements

- Constructor ile boş ya da tek elemanlı bir general tree oluşturulur.
- Ağaça add methodu kullanılarak eleman eklenebilir. Eklenmek istenen child ve eklemenin yapılacağı parent itemleri add method'una parametre olarak verilir. Eğer parent elemani ağaç içerisinde bulunuyorsa ekleme işlemi başarılı olur.
- Ağaç üzerinde preOrderTraverse kullanılarak ağaç içerisindeki elemanlar print edilir.
- Ağaç üzerinde postOrderTraverse ve levelOrderTraverse kullanılarak arama yapılabilir.

2 METHOD

2.0 Class Diagrams



2.1 Problem Solution Approach

Generel tree oluşturulurken BinaryTree sınıfında yararlanıldı.

Arama yapmak için iki yöntem kullanıldı.

Birincisi: PostOrderTraverse kullanılarak ağaç soldan başlanarak taranıldı ve bulunan elemanın bulunduğu node döndürüldü

İkincisi: LevelOrderTraverse kullanılarak ağaç seviye seviye tarandı ve bulunan eleman return edildi.

Ağaca ekleme yapmak için add method'u oluşturuldu. Parametresinde parent ve child ikilisini alarak parent'ın ağaçta bulunduğu durum koşulunda eklemenin yapılması sağlandı.

3 RESULT

3.0 Time Complexities

add method'u için:

```
private void add(Node<E> root, E parent, E child) {
    if(root == null)
        addResult = false;
    else if(parent.compareTo(root.data) == 0) {
        if(root.left == null)
            root.left = new Node<>(child);
        else
            addLastChild(root.left, child);
        addResult = true;
    }
    else if(root != null) {
        add(root.right, parent, child);
        add(root.left, parent, child);
    }

}
```

addLastChild methoduyla birlikte fonksiyon zamanı O(N*N) olur.

O(N)

Son eleman bulunana kadar recursive çağrı devam eder.
N kadar eleman olduğu varsayılırsa t = O(N) olur.

Add method'unun yardımcı method'u:

```
private Node<E> addLastChild(Node<E> parent, E child) {

    if(parent.right == null) {
        parent.right = new Node<>(child);
        return parent.right;
    }
    else {
        return addLastChild(parent.right, child);
    }
}
```

O(N)

Son child'a kadar devam eder. N kadar child olduğu düşünülürse t = O(N) olur.

PreOrderTraverse method'u için:

```
protected void preOrderTraverse(Node < E > node, int depth,
                                StringBuilder sb) {
    for (int i = 1; i < depth; i++) {
        sb.append(" ");
    }
    if(node != null) {
        sb.append(node.toString());
        sb.append("\n");
        preOrderTraverse(node.left,  depth: depth + 1, sb);
        for(int i = sb.length()-1; i > 0 && (sb.charAt(i) == ' ' || sb.charAt(i) == '\n'); i--)
            sb.deleteCharAt(i);
        sb.append("\n");
        preOrderTraverse(node.right, depth, sb);
    }
}
```

Recursive çağrılar n kere yapılır.
Zaman O(N) olur

LevelOrderSearch method'u için

```
private E levelOrderSearch(Node<E> root, E target) {
    System.out.println("\nLevel Order Search for " + target + ":");
    System.out.print("My Tree: ");
    if(root != null) {
        E item = searchSiblings(root, target);
        if(item != null && item.compareTo(target) == 0) {
            return item;
        }
    }
    while(root != null) {
        E item = searchingOneLevel(root, target);
        if(item != null && item.compareTo(target) == 0) {
            return item;
        }
        root = root.left;
    }
    return null;
}
```

İki yardımcı fonksiyonun toplam süresi O(2N) olur.
Dolayısıyla iterative method'un çalışma süresi O(N)'dir.

O(N) kadar sürede çalıştırılırlar.

PostOrderTraverse method'u için:

```
private boolean postOrderTraverse(Node<E> root, E target, boolean found) {
    if(root != null) {
        found = postOrderTraverse(root.left, target, found);
        found = postOrderTraverse(root.right, target, found);
        if(target != null && target.compareTo(root.data) == 0) {
            found = true;
            postResult = root;
        }

        System.out.print(root.data + " ");
    }
    if(found)
        return found;
    return false;
}
```

O(N) = O(2N)

Method kendisini 2N defa çağırır.
Dolayısıyla O(2N) zaman sürer.

3.1 Running Results

Test 1: Add method'u kullanılarak ağaç oluşturulur ve eleman eklenmeme durumu test edilir.

```
Birinci Agac:
0
 1
  4
   10
  5
  11
 2
  6
  7
 3
  8
  9

Ikinci Agac:
-1
 -2
  -3
   -10
    -11
     -12
  -4
 -5
  -8
  -9
 -6
 -7

Agacta olmayan elemana(-19) child ekleme durumu: false
```

Test 2: PreOrderTraverse kullanılarak ağaç içerisindeki elemanlar seviyelerine göre print edilir.

```
Birinci agac:
0
 1
  4
   10
  5
  11
 2
  6
  7
 3
  8
  9

Ikinci agac:
-1
 -2
  -3
   -10
    -11
     -12
  -4
 -5
  -8
  -9
 -6
 -7
```

Test 3: LevelOrderSearch method'u ile ağaç içerisinde arama yapılır. Seviye seviye elemanlar aranır. Birinci durumda elemanın bulunmama durumu test edilirken ikinci durumda bulunma durumu test edilir. İki ağaçta arama yapılır.

```
Level Order Search for 111:
My Tree: 0 1 2 3 4 5 11 6 7 8 9 10
Level Order Search for 10:
My Tree: 0 1 2 3 4 5 11 6 7 8 9 Target found: 10

Level Order Search for -122:
My Tree: -1 -2 -5 -6 -7 -3 -4 -8 -9 -10 -11 -12
Level Order Search for -4:
My Tree: -1 -2 -5 -6 -7 -3 Target found: -4

Process finished with exit code 0
```

Test 4: PostOrderSearch method'u ile ağaç içerisinde arama yapılır. PostOrderTraverse ile

elemanlar aranır. Birinci durumda elemanın bulunmama durumu test edilirken ikinci durumda bulunma durumu test edilir. İki ağaçta arama yapılır.

```
Post Order Search for 111:
My Tree: 10 11 5 4 7 6 9 8 3 2 1 0
Post Order Search for 10:
My Tree: 10 11 5 4 7 6 9 8 3 2 1 0 Target found: 10

Post Order Search for 12345:
My Tree: -12 -11 -10 -4 -3 -9 -8 -7 -6 -5 -2 -1
Post Order Search for -4:
My Tree: -12 -11 -10 -4 -3 -9 -8 -7 -6 -5 -2 -1 Target found: -4

Process finished with exit code 0
```