Veri Yapıları Laboratuvarı Ödev 4

Öğr.Gör.Şevket Umut ÇAKIR
CENG 215 - Veri Yapıları

15 Aralık 2019

1 Çizge Temsili ve Sınıflar

Çizge temsili için düğüm(vertex) listesi ve her bir düğümdeki kenarların(edge) listesi bulunmaktadır. Şekil 1'de `Vertex` ve Şekil 2'de `Edge` sınıflarının yapısı bulunmaktadır.

```
import java.util.ArrayList;
import java.util.List;
public class Vertex<T> {
    public T value;//Düğümün adı
    public List<Edge<T>> edges;//Kenar listesi
    public Vertex(T value) {
        this.value = value;
        edges=new ArrayList<>();
    }
}
```

Şekil 1: Vertex sınıfı

Düğüm ve kenarları içinde barındıran çizge için dokümanın sonunda `Odev4AbstractGraph` sınıfı bulunmaktadır. Bu sınıfta temel çizge işlemleri yapılmaktadır.

Ödevde komşuluk matrisi hesaplayan `adjacencyMatrix` metodu ve parametre olarak verilen çizgenin mevcut çizgeye eşitliğini test eden `equals` metodunun yazılması istenmektedir.

2 Komşuluk Matrisi

Mevcut çizgenin komşuluk matrisinin temsilini hesaplayıp geri döndüren metodun yazılması istenmektedir. Metodun imzası `public int[][] adjacencyMatrix()` şeklindedir. Düğümlerin matristeki yerleşimleri `vertices` listesindeki sırayla aynı olmalıdır. Örneğin vertices listesinde düğümler P,A,U sıralamasına sahipse matrisin yapısı Tablo 1'deki gibi olmalıdır. `Odev4Graph` sınıfı `Odev4AbstractGraph` sınıfından türetilerek gerçekleştirilir. Bu nedenle

```java
public class Edge<T> {
    Vertex<T> from; //Kenarın çıktığı düğüm
    Vertex<T> to; //Kenarın girdiği düğüm
    int weight; //Kenarın ağırlığı
    public Edge(Vertex<T> from, Vertex<T> to) {
        this.from = from;
        this.to = to;
        weight=1;
    }
    public Edge(Vertex<T> from, Vertex<T> to, int weight) {
        this.from = from;
        this.to = to;
        this.weight = weight;
    }
}
```

Şekil 2: Edge sınıfı

`Odev4AbstractGraph` sınıfının bütün özelliklerini devralır. Kodunuzun doğru olduğunu kontrol eden test metotları `Odev4AbstractGraph` sınıfında bulunan düğüm ve kenar ekleme metotlarını kullanarak rastgele bir çizge oluşturur. Sizde istenen her çalıştırmada değişen bu rastgele çizgeye ait komşuluk matrisini iki boyutlu tamsayı dizisi şeklinde geri döndürmenizdir.

Tablo 1: Örnek komşuluk matrisi

	P	**A**	**U**
P	0	w_{pa}	w_{pu}
A	w_{ap}	0	w_{au}
U	w_{up}	w_{ua}	0

3 Eşitlik Testi

İmzası `public abstract boolean equals(Odev4AbstractGraph<T> g2)` olan ve parametre olarak gönderilen `g2` nesnesinin mevcut nesneye eşit olup olmadığını test eden metodu yazmanız istenmektedir. Gönderilen `g2` parametresinin türü `Odev4AbstractGraph` olarak gözükmektedir. `Odev4AbstractGraph` soyut bir sınıf olduğu için ve bu sınıftan bir nesne oluşturulamayacağı için, aslında, gönderilen nesne bir `Odev4Graph` örneğidir. `Odev4Graph` nesneleri bu sınıfın temel sınıfı olan `Odev4AbstractGraph` olarak temsil edilebilmektedir.

Eşitlik testi için rastgele olarak oluşturulan 10 durum ardı ardına `equals` metoduna gönderilmektedir. Bu testin başarılı olabilmesi için bu 10 durum için yapılan kontrollerin hepsi başarılı olmalıdır.

Önemli Tarihler :

Tablo 2: Önemli Tarihler

Olay	Tarih	Konum	Biçim
Ödev Teslimi	24.12.2019	bilmoodle.pau.edu.tr	Odev4Graph.java

Ödev Teslimi ile İlgili Açıklamalar

- Ödevler Programlama Ödevleri Moodle Sistemi(http://bilmoodle.pau.edu.tr/) üzerine kaynak kod yüklenecektir.
- Ödevde girdiler rastgele olarak her değerlendirmede üretilmektedir. Kodunuzda hata varsa bazı girdilerde çalışıp, bazılarında çalışmıyabilir. Dolayısıyla kodunuzu bir kaç defa değerlendirmeye göndermeniz önerilir.
- Ödevler teslim süresi bittikten sonra otomatik olarak değerlendirilecektir. Otomatik değerlendirme sonucu notunuzu belirleyecektir. Ödev teslim süresinden önce almış olduğunuz notlar yanıltıcı olabilir.
- Kaynak kod dosyasının en üstüne öğrenci numarası ve ad soyad açıklama satırı olarak eklenmek zorundadır.
- Ödevler bireysel olarak cevaplanacaktır. Kopya olduğu anlaşılan ödevlerin hepsine **0** puan verilecektir.

A Odev4AbstractGraph Sınıfı

```java
import java.lang.reflect.Array;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

public abstract class Odev4AbstractGraph<T> {
    /**
     * Düğümleri tutan liste
     */
    protected List<Vertex<T>> vertices;
    /**
     * Düğümlere kolay erişmek için kullanılan map <değer, düğüm>
     */
    protected Map<T, Vertex<T>> verticesMap;
    /**
     * Çizge yönlü mü
     */
    boolean directed=true;

    public Odev4AbstractGraph() {
        vertices=new ArrayList<>();
        verticesMap=new HashMap<>();
    }

    public Odev4AbstractGraph(boolean directed) {
        this();
        this.directed=directed;
    }

    /**
     * Çizgeye düğüm ekler
     * @param deger eklenecek düğümün değeri
     */
    public void addVertex(T deger){
        if(!vertices.contains(deger)) {
            Vertex<T> v=new Vertex<>(deger);
            vertices.add(v);
            verticesMap.put(deger, v);
        }
    }

    /**
```

```
44          * Çizgeye kenar ekler
45          * @param from kenarın çıktığı düğüm
46          * @param to kenarın girdiği düğüm
47          */
48         public void addEdge(T from, T to){
49             addEdge(from,to,1);
50         }
51 
52         /**
53          * Çizgeye kenar ekler
54          * @param from kenarın çıktığı düğüm
55          * @param to kenarın girdiği düğüm
56          * @param weight kenarın ağırlığı
57          */
58         public void addEdge(T from, T to, int weight) {
59             Vertex<T> f=verticesMap.get(from);
60             Vertex<T> t=verticesMap.get(to);
61             if (f!=null && t!=null){
62                 for (Edge<T> e:f.edges)
63                     if(e.to.value.equals(to))
64                         return;
65                 Edge e1=new Edge(f,t,weight);
66                 //edges.get(from).add(e1);//alttaki şekilde değişti
67                 verticesMap.get(from).edges.add(e1);
68                 if(!directed){
69                     Edge e2=new Edge(t,f,weight);
70                     //edges.get(to).add(e2);//alttaki şekilde değişti
71                     verticesMap.get(to).edges.add(e2);
72                 }
73             }
74 
75         }
76 
77         /**
78          * Çizgenin içeriğini ekrana yazdırır
79          */
80         public void print() {
81             System.out.println(toString());
82         }
83 
84         /**
85          * Düğüm ve kenarları içeren metinsel temsil
86          * @return Çizgenin metin temsilini verir
87          */
88         @Override
```

```
89      public String toString() {
90          StringBuilder sb=new StringBuilder();
91          sb.append("Vertices:\n");
92          for (Vertex<T> vertex:vertices)
93              sb.append(vertex.value+"\n");
94          sb.append("Edges:\n");
95          for (Vertex<T> vertex:vertices)
96              for (Edge<T> edge:vertex.edges)
97                  sb.append(edge.from.value+(edge.weight!=1?" -
                    ↪  "+edge.weight:"")+" -> "+edge.to.value+"\n");
98          return sb.toString();
99      }
100
101     public abstract int[][] adjacencyMatrix();
102     public abstract boolean equals(Odev4AbstractGraph<T> g2);
103 }
```