Graf Uygulamaları

Grafların komşu liste ve komşu matris şeklinde iki gösterimi vardır. Komşu liste gösteriminde her düğüm için bir bağlı liste vardır. Bu bağlı listede o düğümden çıkan kenarların bağlı olduğu düğümlerin sıra numarası yer alır. Eğer bir düğümün bağlı olduğu hiç kenar yoksa bağlı listesi boştur. İlk olarak komşu liste gösterimi için bağlı liste veri yapısı tanımlanmıştır.

```
class Eleman{
    Eleman ileri;
    int icerik;

    Eleman(int icerik){
        this.icerik=icerik;
        ileri=null;
    }
}

class Liste{
    Eleman bas, son = null;
```

Komşu liste ile graf tanımı için, düğüm sayısını belirten bir N değişkeni ve her düğüm için bir bağlı liste oluşturulacağından bu bağlı listeleri tutan kenar isimli N boyutlu bir bağlı liste dizisi tanımlanmıştır. Bu dizinin her elemanı bir düğümü gösterir ve her düğüm için bir bağlı liste oluşturulmuştur. Örneğin kenar[0], 0 numaralı düğümün bağlı listesini gösterir.

```
public class ListeGraf {
    Liste [] kenar;
    int N;
    public ListeGraf(int N){
        this.N = N;
        kenar = new Liste[N];
        for (int i = 0; i < N; i++) {
            kenar[i] = new Liste();
        }
    }
```

Grafa bir kenar eklemek için kenarın başlangıç ve bitiş düğümlerinin sıra numarası parametre olarak alınmıştır. Başlangıç düğümünün bağlı listesine kenar dizisinden ulaşılıp, bitiş düğümünün sıra numarası ListeyeEkle metodu ile yeni bir eleman olarak eklenmiştir. ListeyeEkle metodunda bağlı listenin sonuna ekleme işlemi yapılmıştır.

```
public void kenarEkle(int baslangic, int bitis){
    kenar[baslangic].ListeyeEkle(bitis);
}
```

```
public void ListeyeEkle(int eleman){
    Eleman e = new Eleman(eleman);
    if(bas==null){
        bas= e;
        son = e;
    }
    else{
        son.ileri = e;
        son = e;
    }
```

Graf üzerinde her düğümün bağlantılı olduğu düğümleri göstermek için kenarGoster isimli bir metot yazılmıştır. Bu metot ile graf üzerindeki her düğümün bağlı listesine ulaşılmış ve bu bağlı listedeki elemanların içeriği (bağlı olduğu düğümlerin sıra numarası) listeGoster metodu ile yazdırılmıştır.

```
public void kenarGoster(){
    for (int i = 0; i < N; i++) {
        System.out.print(i+" >> ");
        kenar[i].listeGoster();
        System.out.println();
    }
}
```

```
public void listeGoster(){
    Eleman e = bas;
    while(e!=null){
        System.out.print(e.icerik+" ");
        e = e.ileri;
    }

}
```

Oluşturulan yönsüz graftaki maksimum dereceli düğümü bulmak için dereceyi ve o dereceye sahip düğümün sıra numarasını tutan iki değişken tanımlanmıştır. Derece bir düğümden çıkan kenar sayısıdır. Dolayısı ile bir düğüm ne kadar fazla düğüm ile bağlantılı ise derecesi o kadar yüksektir. Bu yüzden her düğümün bağlı listesine ulaşılıp listede bulunan eleman sayısı yani o düğümün bağlı olduğu düğüm sayısı (kenar sayısı) elemanSayisi metodu ile bulunmuştur. Hangi düğümün bağlı olduğu düğüm sayısı fazla ise o düğümün sıra numarası maxDugum değişkenine, derecesi ise maxDerece değişkenine alınmıştır.

```java
public void maxDerece(){
    int maxDerece = 0;
    int maxDugum = 0;
    for (int i = 0; i < N; i++) {
        if(kenar[i].elemanSayisi()>maxDerece){
            maxDerece = kenar[i].elemanSayisi();
            maxDugum = i;
        }
    }

    System.out.println("Max dereceli düğüm: "+maxDugum+" >> "+maxDerece);
```

```java
public int elemanSayisi(){
    Eleman e = bas;
    int sayac = 0;
    while(e!=null){
        sayac++;
        e = e.ileri;
    }
    return sayac;
}
```

Tam graf, her düğümün graftaki tüm düğümler ile bir kenara sahip olduğu graftır. Yani her düğümün derecesi N-1 dir. Verilen yönsüz grafın tam graf olup olmadığını belirlemek için tüm düğümlerin bağlı listelerine ulaşılıp elemanSayisi metodu ile bağlı olduğu düğüm sayıları bulunmuştur. Eğer tüm düğümlerin bağlı listesindeki eleman sayıları N-1 e eşit yani graftaki diğer tüm düğümler ile bağlantılı ise true değeri, değil ise false değeri geri döndürülmüştür.

```java
public boolean grafTamMı(){
    for (int i = 0; i < N; i++) {
        if(kenar[i].elemanSayisi()!=(N-1)){
            return false;
        }
    }
    return true;
}
```

Grafın komşu matris gösterimi için NxN boyutlu bir kenar matrisi oluşturulmuştur. Bu matrisin ilk değerleri grafta başlangıçta hiç kenar olmadığı için 0 olarak verilmiştir.

```java
public class MatrisGraf {
    int [][] kenar;
    int N;
    public MatrisGraf(int N){
        this.N=N;
        kenar = new int[N][N];
        for (int i = 0; i < N; i++) {
            for (int j = 0; j < N; j++) {
                kenar[i][j]=0;
            }
        }
    }
```

Grafa kenar eklemek için kenarların başlangıç ve bitiş düğümlerinin sıra numaraları parametre olarak alınmıştır. Komşuluk matrisinde başlangıç düğümünün satırında bitiş düğümünün sütun değeri 1 yapılarak iki düğüm arasında kenar oluşturulmuştur.

```java
public void kenarEkle(int baslangic, int bitis){
    kenar[baslangic][bitis]=1;
}
```

Her düğümün bağlı olduğu düğümleri göstermek için kenarGoster metodu yazılmıştır. Komşuluk matrisinde her düğümün satırında 1 değerine eşit olan sütunları o düğümün bağlı olduğu düğümlerin sıra numaralarını göstermektedir.

```java
public void kenarGoster(int [][]kenar){
    for (int i = 0; i < kenar.length; i++) {
        System.out.print(i+" >> ");
        for (int j = 0; j < kenar.length; j++) {
            if(kenar[i][j]==1){
                System.out.print(j+" ");
            }
        }
        System.out.println();
    }
}
```

Yönlü bir grafın tersini elde etmek için graftaki tüm kenarların yönünün değiştirilmesi gerekir. Verilen yönlü grafın ters grafını oluşturmak için NxN boyutlu bir komşuluk matrisi oluşturulmuş ve matrisin ilk değerleri 0 olarak verilmiştir. Kenarların yönü değiştirileceği için kenar matrisindeki değeri 1 olan elemanların satır ve sütunları yer değiştirilerek elde edilen elemanın değeri ters matriste 1 olarak güncellenmiştir. Bu şekilde kenar matrisinde tanımlanan bir kenarın başlangıç ve bitiş düğümleri yer değiştirilmiş ve ters graf için oluşturulan komşuluk matrisinde kenar tersi yönde tanımlanmıştır.

```
public void tersGraf(){
    int tersMatris [][] = new int[N][N];
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++) {
            tersMatris[i][j]=0;
        }
    }

    for (int i = 0; i < kenar.length; i++) {
        for (int j = 0; j < kenar.length; j++) {
            if(kenar[i][j]==1){
                tersMatris[j][i]=1;
            }
        }
    }

    kenarGoster(tersMatris);
```

Yönlü bir grafta düğüm derecesi giriş ve çıkış derecesi olarak belirlenir. Giriş derecesi o düğüme gelen kenar sayısını, çıkış derecesi ise o düğümden çıkan kenar sayısını göstermektedir. Yönlü bir grafta sıra numarası verilen bir düğümün derecesini bulmak için komşuluk matrisinde ilgili düğümün satırına bakılır. Satırdaki 1 sayısı o sıra numarasındaki düğümden çıkan kenar sayısını göstermektedir. Her 1 sayısı için tanımlanan derece değişkeni bir arttırılır ve tüm sütunlar kontrol edildikten sonra ilgili düğümün çıkış derecesi bulunur.

```
public void dugumDerece(int n){
    int derece = 0;
    for (int i = 0; i < kenar.length; i++) {
        if(kenar[n][i]==1){
            derece++;
        }
    }

    System.out.println(n +" numaralı düğümün derecesi: "+derece);
}
```