Reprezentarea grafurilor neorientate

Pentru a putea prelucra un graf neorientat cu ajutorul unui program, trebuie mai întâi să fie reprezentat in programul respectiv.

Pentru a reprezenta un graf, într-un program, există mai multe modalităti folosind diverse structuri de date:

- reprezentarea unui graf prin matricea de adiacentă;
- reprezentarea unui graf prin listele de adiacentă;
- reprezentarea unui graf prin sirul muchiilor.

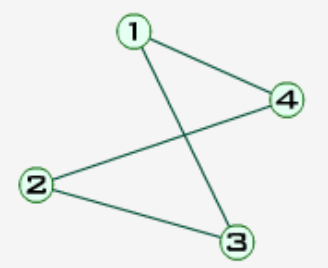
Reprezentarea unui graf prin matricea de adiacentă

Fie G=(V, M) un graf neorieritat cu n vârfuri (V={1,2, ..., n}) si m muchii.
Matricea de adiacentă, asociată grafului G, este o matrice pătratică de ordinul n, cu elementele definite astfel:

$$a_{i,j} = \begin{cases} 1, & daca\ [i,j] \in M \\ 0, & daca\ [i,j] \notin M \end{cases}$$

(altfel spus, $a_{i,j}$=1, dacă există muchie între i si j si $a_{i,j}$=0 dacă nu există muchie între i si j)

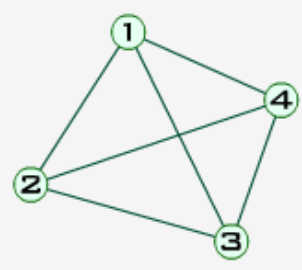
•**Exemplul 1**. Fie graful reprezentat in figura de mai jos:



Matricea de adiacentă, asociată grafului, este:

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{pmatrix} = \begin{pmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{pmatrix}$$

Exemplul 2. Fie graful reprezentat in figura de mai jos:



$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{pmatrix} = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$$

Comentarii:

Matricea de adiacentă este o matrice **pătratică**, de ordin n, si **simetrică** fată de diagonala principală (adică **a[i][j]=a[j][i]).**

Secventele de citire a matricei de adiacentă, sunt:

Metoda1:

```
int a[100][100];
.....
cout<<"n="; cin>>n;
for (i=1;i<=n-l;i++)  //se citesc valorile elementelor de deasupra diagonalei principale
   for (j=i+l; j<=n; j++)
        {
              cin>>a[i][j];      //si se transferă si sub diagonala principală
              a[j][i]=a[i][j];
        }
```

Metoda 2:

```
.....
cin>>n;
cin>>m;
for (i=1;i<=n;i++)
     for (j=1;j<=n;j++)
              a[i][j]=0;
              for (i=1;i<=m;i++)
              {
                    cout<<"dati extremitatile muchiei "<<i;
                    cin>>x >>y;
                    a[x][y]=1;
                     a[y][x]= 1;
                }
```

2. Matricea de adiacentă **are toate elementele de pe diagonala principală** egale cu **0**.
3. **Numărul elementelor egale cu 1 de pe linia i** (sau coloana j) este egal cu **gradul vârfului i**.
4. Dacă vârful **i** este un **vârf izolat pe linia i** (si coloana j) **nu sunt elemente egale cu 1**.

Probleme propuse:

Fie G un graf neorientat, cu n vârfuri si m muchii, reprezentat prin matricea de adiacentă. Să se realizeze programe, în C/C++, care:

a) afisează gradele tuturor vârfurilor;
b) afisează vârfurile de grad par;
c) afisează vârfurile izolate;
d) afisează vârfurile terminale;

e) verifică dacă graful are vârfuri izolate;
t) verifică dacă graful are vârfuri terminale;
g) verifică dacă graful are vârfuri interioare (nu sunt nici izolate nici terminale);
h) verifică dacă graful are toate vârfurile izolate;
i) verifică dacă graful are toate vârfurile interioare (nu sunt nici izolate nici terminale);

j) afisează gradul unui vârf dat;
k) afisează vecinii unui nod dat, vf;
l) verifică dacă un vârf dat este terminal, izolat sau interior;
m) afisează gradul cel mai mare si toate vârfurile care au acest grad
n) afisează frecventa vârfurilor:
izolate : n1
terminale : n2
interioare : n3

Indicatie:
Construiti subprogramul **grad** care returneaza gradul unui varf. Folositi aceasta functie pentru rezolvarea problemelor!