Grafuri Neorientate

Grafuri Neorientate

Adiacenta.Incidenta.Grad

Definitie

- Un **graf neorientat** este o pereche ordonată de mulţimi (X,U), unde:
  - -X este o mulţime finită şi nevidă de elemente numite **noduri** sau **vârfuri**
  - -U este o mulţime de perechi neordonate din X, numite **muchii**

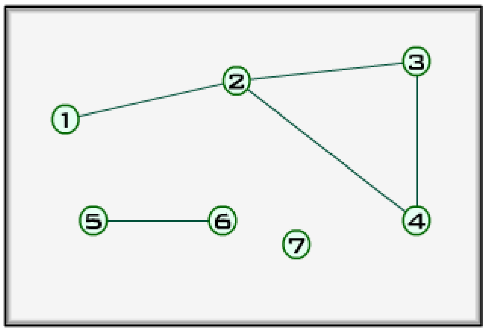
Teorie

TEOREMĂ

Într-un graf G=(X,U) cu **n** vârfuri şi **m** muchii, suma gradelor tuturor vârfurilor este egală cu 2*numărul muchiilor, altfel spus:

$$\sum_{i=1}^{n} d(x_i)=d(x_1)+d(x_2)+\ldots+d(x_n)=2*m$$

EXEMPLU



Mulţimea nodurilor: X={1, 2, 3, 4, 5, 6, 7}
Mulţimea muchilor: U={(1,2), (2,3), (2,4), (3,4), (5,6)}
Gradul nodului 2: d(2)=3
Nod izolat: 7
Noduri terminale: 1, 5, 6
Nodurile 2 şi 3 se numesc **adiacente**. La fel nodurile 2 şi 4.
Nodul 3 şi muchia (3,4) se numesc **incidente**.

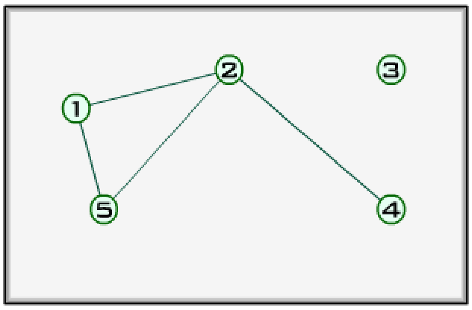
- Gradul unui nod **x**, notat d(x) reprezinta numarul muchiilor care trec prin nodul x(incidente cu x)
- Un nod care are gradul 0 se numeste nod izolat
- Un nod care are gradul 1 se numeste nod terminal
- Daca exista muchia u=(x,y) atunci vom spune ca nodurile x si y sunt **adiacente**
- Daca exista muchia u=(x,y) atunci vom spune ca nodul x si muchia u sunt i**ncidente**. La fel nodul y si muchia u

Grafuri Neorientate

EXERCITIUL 1

➤ În graful de mai jos avem:

i



a) Nodurile 2 şi 5 se numesc **adiacente** ✓

b) Nodul 4 este un nod **terminal** ✓

c) Pentru nodul 2 putem spune că 3 reprezintă **gradul** său ✓

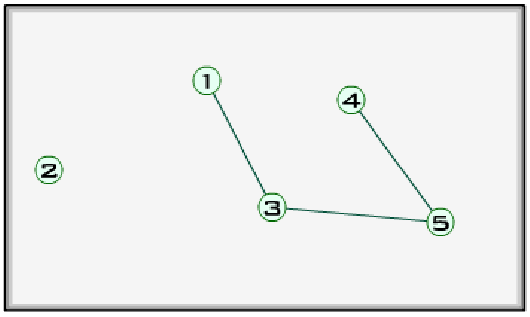
d) Nodul 3 este un nod **izolat** ✓

e) Muchia (2,5) şi nodul 2 se numesc **incidente** ✓

EXERCITIUL 2

➤ Desenaţi un graf cu 5 noduri, în care nodul 2 să fie **nod izolat**, să aibă un **vârf terminal**, iar **gradul** nodului 3 să fie 2.

i



Corect !

Adaugă noduri

Mută noduri

Trasează muchii

Şterge graful

Verifică

Grafuri Neorientate

Reprezentare

Teorie

REPREZENTĂRI

- O metodă de reprezentare a unui graf neorientat foarte folosită este matricea de adiacenţă. Aceasta are proprietatea că a[i,j]=a[j,i] oricare ar fi i,j∈ {1,2,3,...,n}, cu i ≠ j. Adică matricea de adiacenţă a este **simetrică** faţă de diagonala principală.
- Fiecare muchie a grafului poate fi privită ca o înregistrare cu două componente: cele două vârfuri care constituie extremităţile muchiei. Notând aceste extremităţi cu **x** şi **y**, putem defini tipul de date **MUCHIE**, astfel:

```
typedef struct MUCHIE{
        int x,y;
        }
```

Astfel că putem reprezenta graful şi ca un *"vector de muchii "*, adică un vector cu elemente de tipul **MUCHIE**:

```
MUCHIE v[25];
```

GRAFURI NEORIENTATE | Obiective

REPREZENTARE

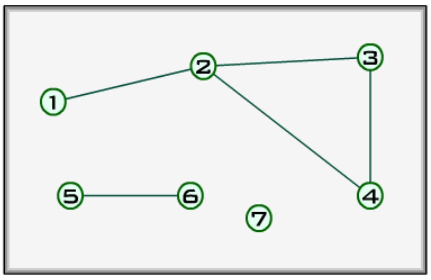
DEFINIŢII

- **Matricea de adiacenţă** este o matrice **a** cu **n** linii şi **n** coloane, în care elementele **a[i,j]** se definesc astfel:

$$a[i,j]=\begin{cases}1, \text{ dacă } \exists \text{ muchia } [i,j] \text{ cu } i\neq j\\ 0, \text{ în caz contrar}\end{cases}$$

- **Lista vecinilor nodului x** cuprinde toate nodurile care sunt extremităţi ale muchiilor ce trec prin nodul **x**.

EXEMPLU



Matricea de adiacenţă

	1	2	3	4	5	6	7
1	0	1	0	0	0	0	0
2	1	0	1	1	0	0	0
3	0	1	0	1	0	0	0
4	0	1	1	0	0	0	0
5	0	0	0	0	0	1	0
6	0	0	0	0	1	0	0
7	0	0	0	0	0	0	0

Lista vecinilor

nodul	*lista vecinilor*
1	2
2	1,3,4
3	2,4
4	2,3
5	6
6	5
7	

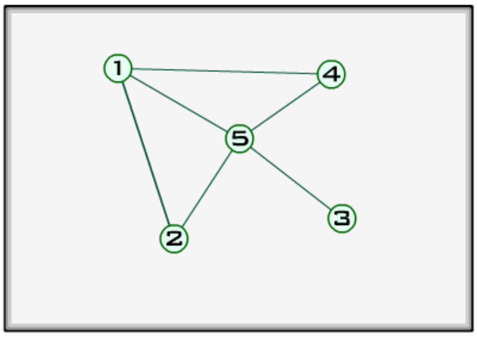
Vectorul de muchii U={(1,2), (2,3), (2,4), (3,4), (5,6)}

Grafuri Neorientate

EXERCITIUL 1

➤ Desenaţi un graf şi urmăriţi cum se modifică matricea de adiacenţă, listele vecinilor şi vectorul de muchii.

i



Matricea de adiacenţă

	1	2	3	4	5
1	0	1	0	1	1
2	1	0	0	0	1
3	0	0	0	0	1
4	1	0	0	0	1
5	1	1	1	1	0

Lista vecinilor

nodul	*lista vecinilor*
1	4 5 2
2	1 5
3	5
4	1 5
5	1 2 4 3

Vectorul de muchii

U={(1,4), (1,5), (1,2), (5,2), (5,4), (5,3)}

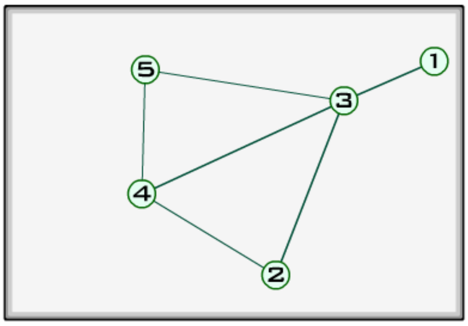
Adaugă noduri | Trasează muchii

Mută noduri | Şterge graful

EXERCITIUL 2

➤ Desenaţi graful corespunzător matricei de adiacenţă generate.
*Folosiţi butonul **Generare matrice** pentru a genera o altă matrice.*

i



Matricea de adiacenţă

	1	2	3	4	5
1	0	0	1	0	0
2	0	0	1	1	0
3	1	1	0	1	1
4	0	1	1	0	1
5	0	0	1	1	0

Corect !

Adaugă noduri | Trasează muchii

Mută noduri | Şterge graful

Verifică

Generare matrice

Grafuri Neorientate

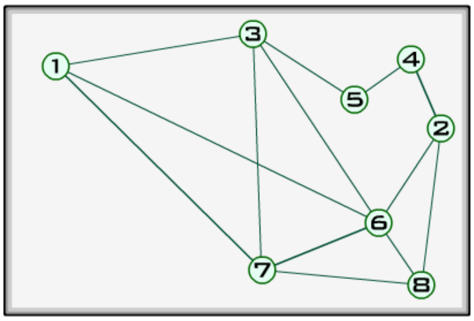
GRAFURI NEORIENTATE

Obiective

REPREZENTARE

EXERCITIUL 3

➤ Desenaţi graful corespunzător pentru lista vecinilor generată.
*Folosiţi butonul **Generare listă** pentru a genera o altă listă de vecini.*



Adaugă noduri | Trasează muchii

Mută noduri | Şterge graful

Verifică

Lista vecinilor

nodul	*lista vecinilor*
1	3 6 7
2	4 6 8
3	1 5 6 7
4	2 5
5	3 4
6	1 2 3 7 8
7	1 3 6 8
8	2 6 7

Corect !

Generare listă

➤ Construirea matricei de adiacenţă pe baza muchiilor citite de la tastatură

```
void cit_graf (matrice a, int& n, int& m)
{
int i, j, k, x, y;
//citeşte numărul de vârfuri n şi numărul de muchii m
  cout<<" Numarul de varfuri : ";   cin>>n;
  cout<<"Numarul de muchii : ";   cin>>m;
//iniţializează cu 0 toată matricea de adiacenţă
  for (i=1; i<=n; i++)
        for (j=1; j<=n;j++) a[i][j]=0;
//citeşte m perechi de numere întregi de forma (x,y)
  for (k=1; k<=m; k++)
   {
     cout<<"Dati muchia cu numarul de ordine "<<k;
     do
     {
          cin>>x>>y;
     }while (x<1 || x>n || y<1 || y>n || x==y);
/*pentru fiecare pereche, facem 1 elementele a[x][y]
şi a[y][x] care identifică muchia (x,y) */
     a[x][y]=1;
     a[y][x]=1;
    }
}
```

Grafuri Neorientate

➤ Determinarea muchiilor unui graf pornind de la matricea de adiacenţă

```
void afisare_graf()
{
/*afişează muchiile grafului pornind de la matricea de
adiacenţă a*/
int i,j;
 cout<<"Muchiile sunt: ";
/*matricea fiind simetrică, se parcurge numai
porţiunea de deasupra diagonalei principale*/
     for(i=1; i<n; i++)
         for(j=i+1;j <=n; j++)
               if(a[i][j]==1)
    /*afiseaza muchia (i,j), daca exista*/
                  cout<<i<<"  "<<j<<"\n";
}
```

➤ Citirea matricei de adiacenţă de la tastatură

```
void cit_matrice (matrice a, int& n)
{
int i, j;
/*citeşte numărul de vârfuri*/
  cout<<"Numarul de varfuri : "; cin>>n;
/*iniţializează cu 0 diagonala principală*/
  for (i=1; i<=n; i++) a[i][i]=0;
/*citeşte portiunea din matrice situată deasupra
  diagonalei principale*/
  for (i=1;i<n;i++)
   for (j=i+1;j<=n;j++)
    {
       cout<<"Exista muchia "<<i<<" - "<<j<<"1-
                                          Da, 0-Nu";
          do {
               /*citeşte elementul a[i,j] cu validare
                (el trebuie să fie 0 sau 1)*/
                cin>>a[i][j];
             } while (a[i][j]!=0 && a[i][j]!=1);
         a[j][i]=a[i][j];
      }
}
```

Grafuri Neorientate

➤ Citirea matricei de adiacenţă dintr-un fişier text

```
void cit_matr_fis (matrice a, int& n)
{
/*citeşte numărul de noduri şi matricea de adiacenţă
din fişierul text*/
int i,j;
char* nume_fis;
ifstream f;
     cout<<"Dati numele fisierului"; cin>>nume_fis;
     f.open (nume_fis, ios::in);
     f>>n;
     for (i=1; i<=n; i++)
      {
          for (j=1; j<=n; j++)
               f>>a[i][j];
      }
     f.close();
}
```

➤ Determinarea gradului unui vârf x folosind matricea de adiacenţă

```
int grad (int x)
/*returnează gradul lui x, adică numărul muchiilor care
trec prin vârful x */
{
     int j, gr;
     gr=0;
/*ne uităm în linia x şi, dacă avem o valoare egală cu
1, o numărăm*/
     for (j=1; j<=n; j++)
          if (a[x][j]==1)   gr++;
     return gr;
}
```

Grafuri Neorientate

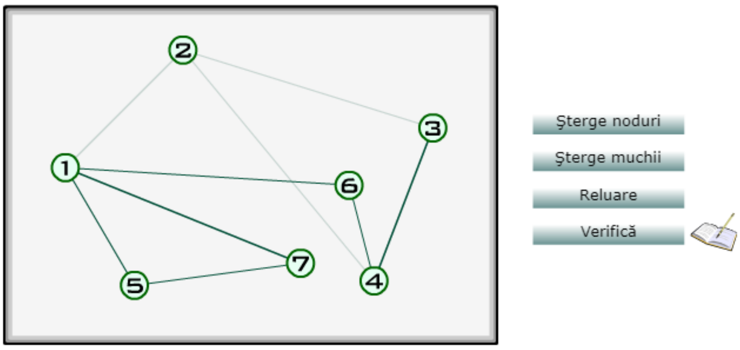
Graf Partial.Subgraf

Definitie

Fie graful $G=(X,U)$. Un **graf parţial** al lui G, este un graf $G_1=(X,V)$, cu $V \subseteq U$. Altfel spus, un graf parţial G_1 al lui G, este chiar G, sau se obţine din G păstrând toate vârfurile şi suprimând nişte muchii.

EXERCITIUL 1

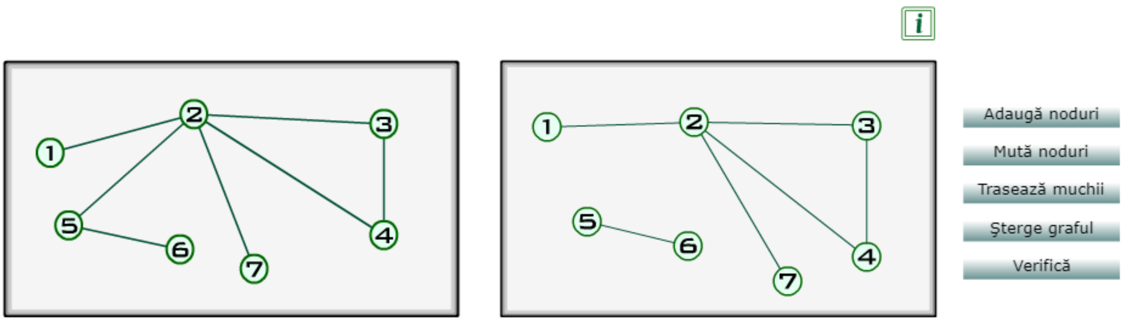
Precizaţi care este **graful parţial** obţinut prin eliminarea muchiilor ce trec prin nodul 2.



Corect!

EXERCITIUL 2

Pentru graful $G=(X,U)$ de mai jos construiţi alăturat un **graf parţial** obţinut prin eliminarea a cel mult 3 muchii.



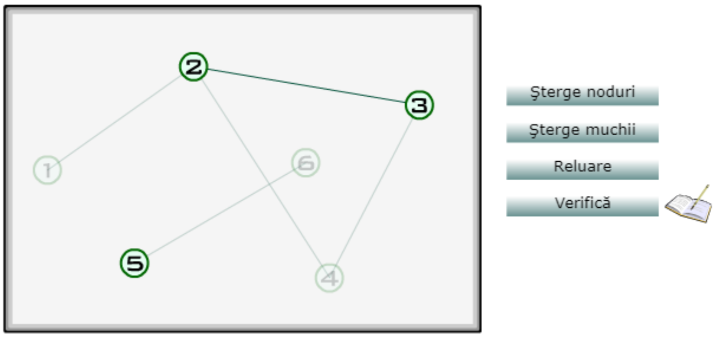
Corect!

Grafuri Neorientate

Definitie

Fie graful $G=(X,U)$. Un **subgraf** al lui G, este un graf $G_1=(Y,V)$, unde $Y \subset X$, iar V va conţine toate muchiile din U care au ambele extremităţi în Y. Altfel spus, un subgraf al unui graf se obţine eliminând nişte noduri şi toate muchiile incidente acestor noduri.

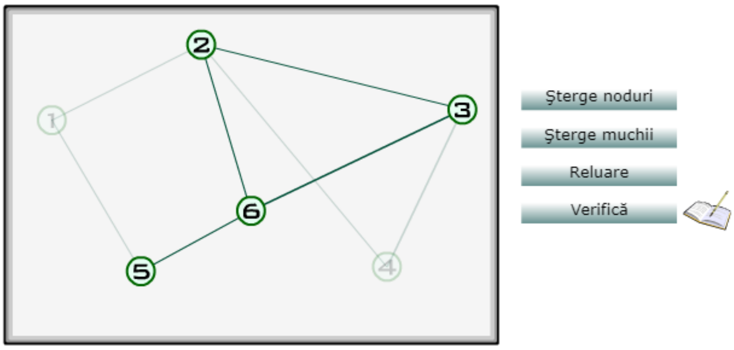
➤ Obtineţi **subgraful** generat de mulţimea de noduri Y={2,3,5}



Corect !

EXERCITIUL 4

➤ Construiţi **subgraful** obţinut prin eliminarea nodurilor 1 şi 4.



Corect !

Grafuri Neorientate

Tipuri de grafuri

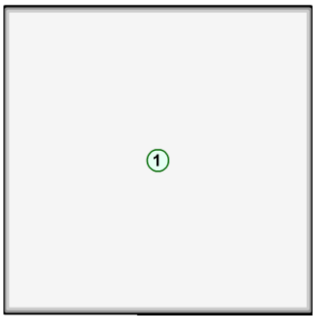
Definitie

Se numeşte **graf complet** cu n vârfuri, notat K_n, un graf G=(X,U) cu proprietatea că oricare două vârfuri sunt adiacente, adică
$\forall\ x, y \in X \Rightarrow \exists$ muchia $[x, y] \in U$.

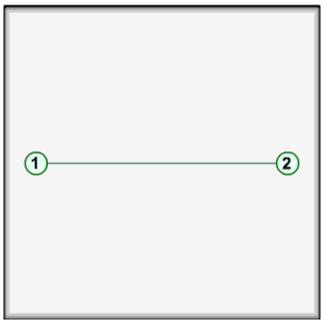
- Un graf complet cu **n** vârfuri, are $\frac{n \cdot (n-1)}{2}$ muchii.

Exemple :

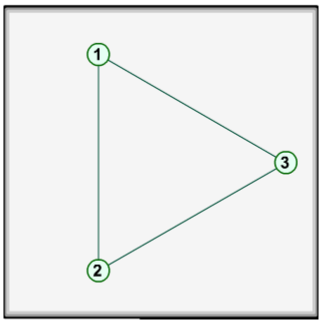
Graful K_1



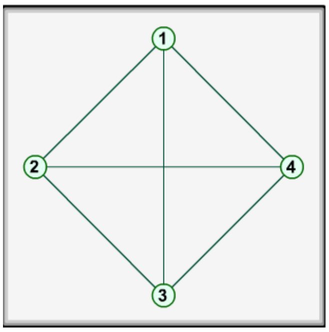
Graful K_2



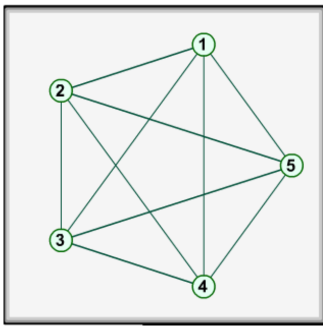
Graful K_3



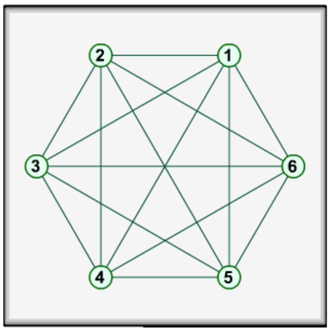
Graful K_4



Graful K_5



Graful K_6



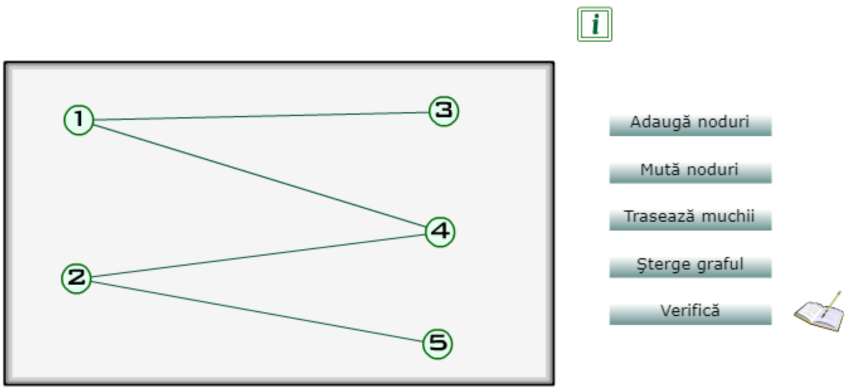
Grafuri Neorientate

Definite

Se numeşte **graf bipartit**, un graf G=(X,U) cu proprietatea că există două mulţimi A şi B incluse în X, astfel încât:

- $A \cap B = \varnothing$, $A \cup B = X$
- toate muchiile grafului au o extremitate în A şi cealaltă în B

➤ Desenaţi un **graf bipartit** cu minim 4 muchii generat pe baza mulţimilor A={1, 2} şi B={3, 4, 5}.



Corect!

Definite

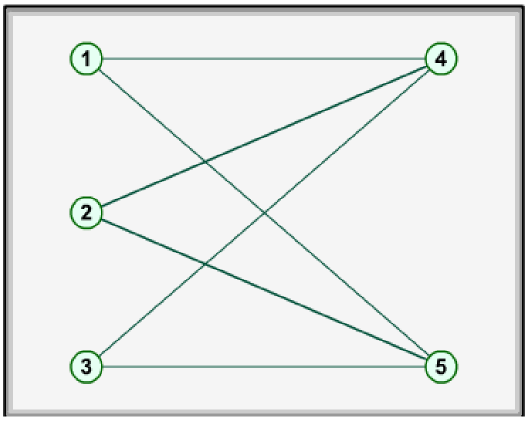
Se numeşte **graf bipartit complet**, un graf bipartit cu proprietatea că pentru orice vârf x din A şi orice vârf y din B, există muchia (x,y) (unde A şi B sunt cele două submulţimi care partiţionează mulţimea vârfurilor X).

➤ Un graf bipartit complet cu **p** vârfuri în prima mulţime şi **q** vârfuri în a doua mulţime are **pq** muchii.

Grafuri Neorientate

Exemplu :

Graful $K_{3\ 2}$

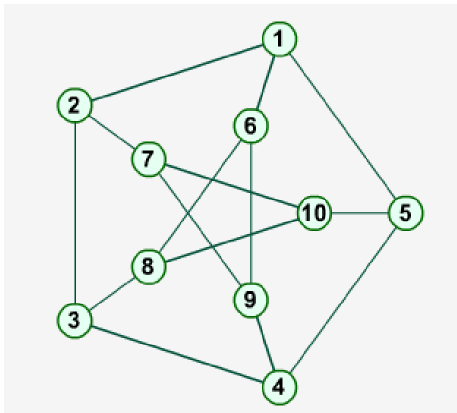


Definitie

Se numeşte **graf planar,** un graf cu proprietatea că există o reprezentare a sa în plan astfel încât oricare două muchii să nu se intersecteze.

Se numeşte **graf regulat**, un graf cu proprietatea că toate nodurile au acelaşi grad.

Graful Petersen



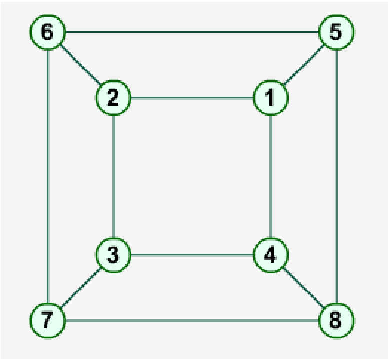
- Graful Petersen nu este un **graf planar**.

- După cum puteţi observa, graful Petersen este un graf în care fiecare nod are acelaşi grad şi anume gradul 3, deci graful Petersen este un **graf regulat**.

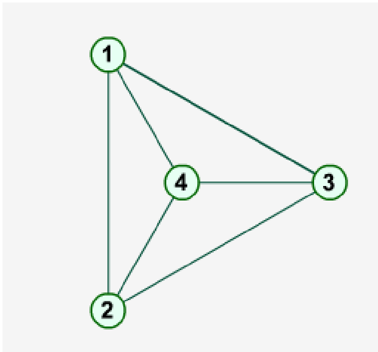
Grafuri Neorientate

Exemplu :

Cubul



Tetraedrul



Lant.Ciclu.Arbori

Definitie

- Se numeşte **lanţ** în graful G, o succesiune de vârfuri $L=(z_1,z_2,\dots,z_k)$, unde $z_1,z_2,\dots,z_k \in X$, cu proprietatea că oricare două vârfuri consecutive sunt adiacente, adică există muchiile $[z_1,z_2]$, $[z_2,z_3]$,…,$[z_{k-1},z_k] \in U$.

Elementar/Neelementar

Pentru un lanţ $L=(z_1,z_2,\dots,z_k)$, dacă vârfurile $z_1,z_2,\dots,z_k$ sunt distincte două câte două, atunci lanţul se numeşte **elementar**. În caz contrar, lanţul este **neelementar**.

Simplu

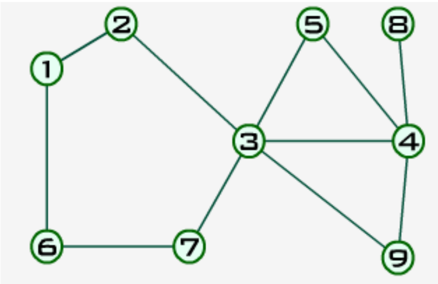
Dacă un lanţ nu conţine de mai multe ori aceeaşi muchie, atunci el se numeşte **simplu**.

Grafuri Neorientate

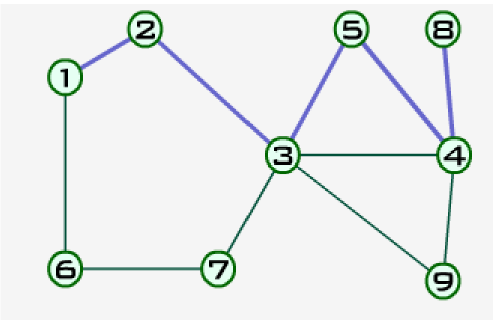
Lungime

Numărul de muchii care intră în componenţa unui lanţ reprezintă **lungimea** lanţului.

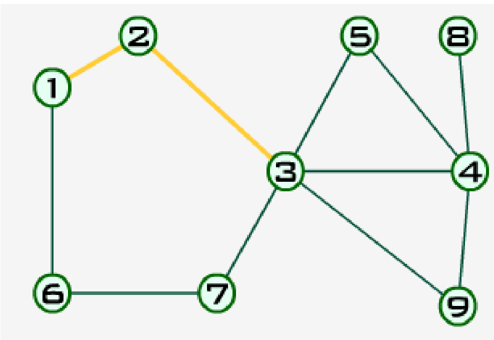
Exemplu :



L_1={1,2,3,5,4,8} care este **elementar**, **simplu** şi de **lungime** 5

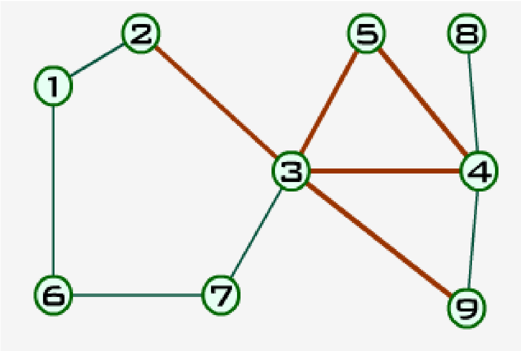


L_2={1,2,3,2} care este **neelementar** şi de **lungime** 3

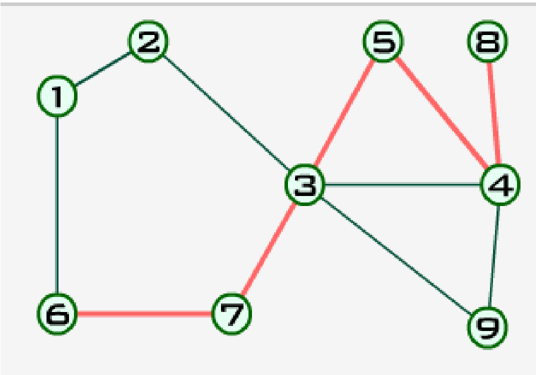


Grafuri Neorientate

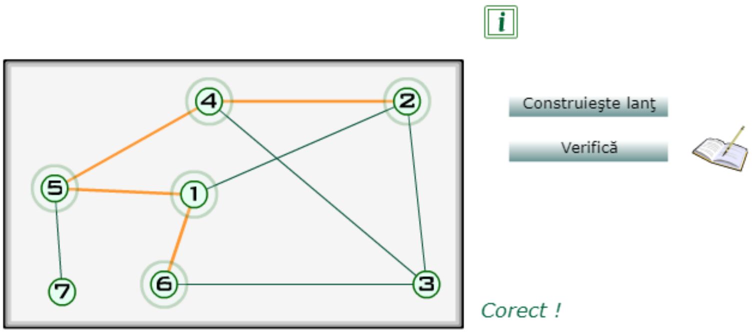
L_3={9,3,5,4,3,2} care este **neelementar**, **simplu** şi de **lungime** 5



L_4={6,7,3,5,4,8} care este **elementar**, **simplu** şi de **lungime** 5



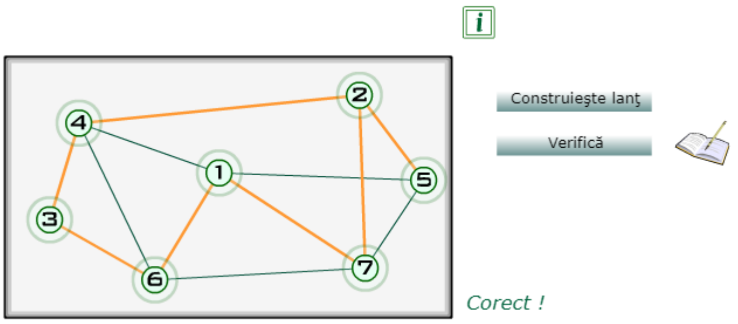
➤ Desenaţi un **lanţ elementar** de la nodul 2 la nodul 6.



L={2,4,5,1,6}

Grafuri Neorientate

➤ Desenaţi un **lanţ neelementar** de la nodul 2 la nodul 5.



L={2,4,3,6,1,7,2,5}

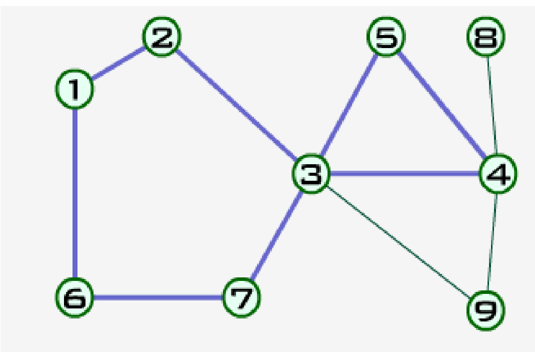
Definitie

➤ Se numeşte **ciclu** într-un graf, un lanţ $L=(z_1,z_2,\ldots,z_k)$ cu proprietatea că $z_1=z_k$ şi muchiile $[z_1,z_2]$, $[z_2,z_3]$,…,$[z_{k-1},z_k]$ sunt distincte două câte două.

Elementar/Neelementar

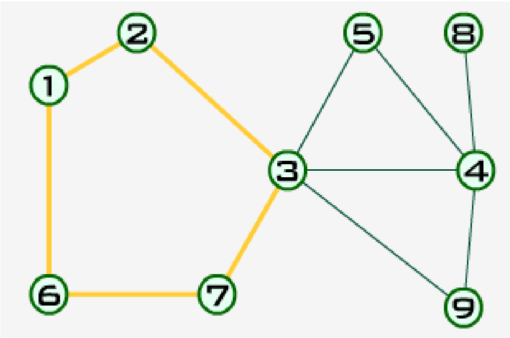
Dacă într-un ciclu, toate vârfurile cu excepţia primului şi a ultimului sunt distincte două câte două, atunci ciclul se numeşte **elementar**. În caz contrar, el este **neelementar.**

C_1={3,4,5,3,7,6,1,2,3} care este **neelementar**

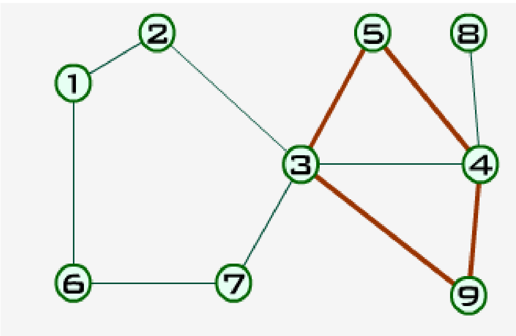


Grafuri Neorientate

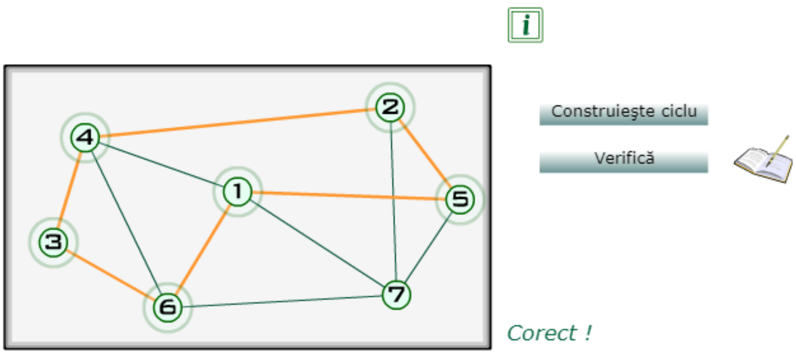
C_2={1,2,3,7,6,1} care este **elementar**



C_3={3,5,4,9,3} care este **elementar**



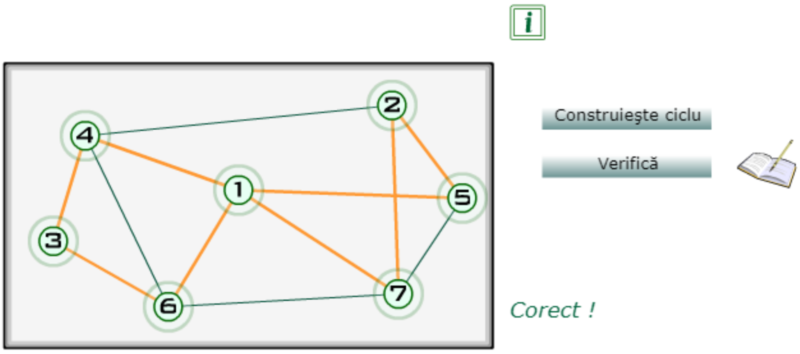
➤ Desenaţi un **ciclu elementar** care pleacă de la nodul 5 şi trece prin nodul 3.



C={5,2,4,3,6,1,5}

Grafuri Neorientate

➤ Desenaţi un **ciclu neelementar** care pleacă din nodul 3 şi trece prin cel puţin alte 4 noduri.



C={3,6,1,5,2,7,1,4,3}

Definitie

➤ Un graf **conex** şi fără cicluri se numeşte **arbore**.

Conex

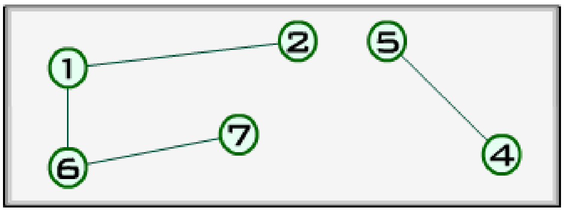
Un graf G se numeşte **conex** dacă oricare ar fi două vârfuri ale sale, există un lanţ care le leagă.

➤ Fie un graf G=(X,U). Următoarele afirmaţii sunt echivalente:

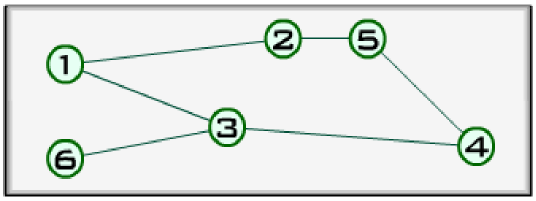
1. G este arbore.
2. G este un graf conex, minimal în raport cu această proprietate (eliminând o muchie oarecare, se obţine un graf ne-conex).
3. G este un graf fără cicluri, maximal în raport cu această proprietate (adăugând o muchie oarecare, se obţine un graf care are cel puţin un ciclu).

➤ Un arbore cu n vârfuri are n-1 muchii.

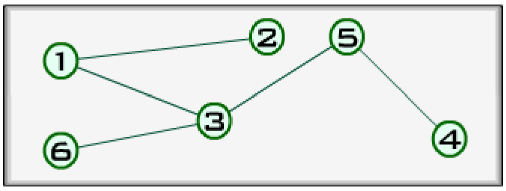
Grafuri Neorientate



Nu este un arbore deoarece, deşi nu are cicluri, nu e conex.



Nu este un arbore pentru că, deşi este conex, are cicluri.



Graful alăturat este un arbore pentru că e conex şi nu are cicluri.

➤ Completaţi frazele de mai jos folosind cuvintele disponibile.

a) Numărul de muchii are unui ***arbore*** cu n noduri este ***n-1***

b) Între oricare două noduri ale unui arbore există un unic ***lanţ***

c) Orice lanţ de lungime 0 conţine un singur ***nod***

d) Dacă un graf conţine un lanţ ce trece prin toate vârfurile, atunci el este ***conex***

- Verificarea pentru o secvenţă de vârfuri (citită dintr-un fişier) dacă reprezintă un lanţ elementar sau ne-elementar.

```
void secventa (matrice a, int n)
{
    ifstream f;
    int ok, i, j, k;
    f.open("lant.txt", ios::in);
{citeşte secvenţa de vârfuri din fişier în vectorul
z=(z[1],z[2],...,z[k])}
    k=0;
    while (!f.eof())
       {
          k++; f>>z[k];
       }
    f.close();
{afişează vectorul z ce conţine secvenţa}
    for (i=1; i<=k; i++) cout<<z[i]<<" ";
    cout<<"\n";
{verificăm dacă două vârfuri consecutive în secvenţă
sunt adiacente (dacă secvenţa e lanţ)}
    ok=1;
    for (i=1; i<=k-1; i++)
        if (a[z[i],z[i+1]]==0) ok=0;
    if (ok)  cout<<"Este un lant ";
        else cout<<"Secventa nu este un lant";
    if (ok)
        {
 {în cazul în care e lanţ, testează dacă vârfurile sunt
 distincte între ele (adică dacă e lanţ elementar)}
            for (i=1; i<k; i++)
                for (j=i+1; j<=k; j++)
                    if (z[i]==z[j]) ok=0;
            if (ok) cout <<"elementar";
                else  cout<<"ne-elementar";
        }
}
```