

PROIECTAREA ALGORITMILOR

Adrian Runceanu

Curs 6

Elemente de teoria grafurilor

Conținutul cursului

6.1. Definiții

6.2. Memorarea(reprezentarea) grafurilor

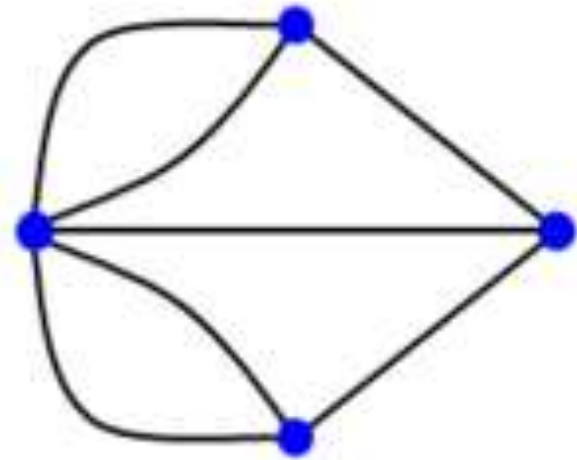
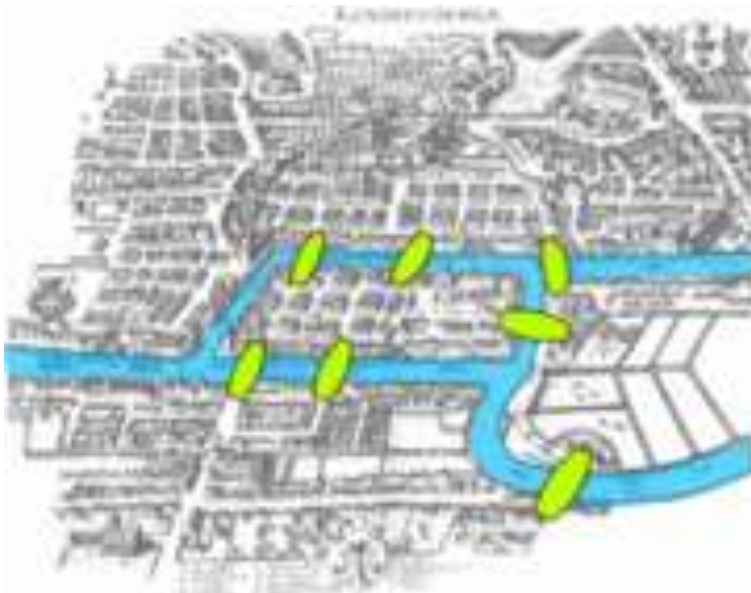
6.3. Parcurgerea grafurilor

Scurte considerații istorice

- Originile teoriei grafurilor se găsesc în rezolvarea unor probleme de jocuri și amuzamente matematice, care au atras atenția unor matematicieni de seamă, cum ar fi:
 - *Euler*
 - *Hamilton*
 - *Cazlyley*
 - *Sylvester*
 - *Birkoff*
- Data nașterii teoriei grafurilor este considerată a fi anul **1736**, când matematicianul *Leonhard Euler* a publicat un articol în care a clarificat **problema celor șapte poduri** și a prezentat o metodă pentru rezolvarea altor probleme de același tip.

Scurte considerații istorice

- În orașul Königsberg (Kalingrad) existau peste râul Pregel șapte poduri.
- Problema celor șapte poduri era:



- *Se poate face o plimbare peste toate cele șapte poduri, trecând o singură dată peste fiecare pod?*

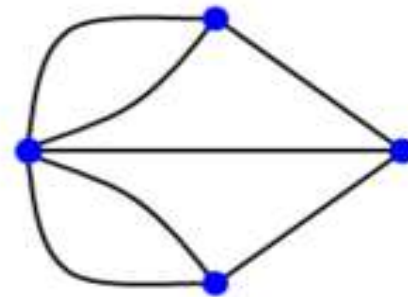
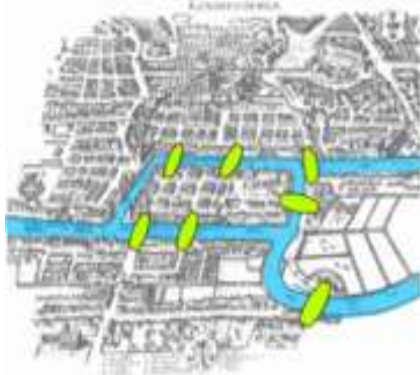
Problema celor șapte poduri

- Pentru a enunța fără echivoc cerința logică a problemei, soluțiile care implică fie:
 - 1) ajungerea pe o insulă sau într-o parte de oraș altfel decât trecând podurile, sau
 - 2) accesarea oricărui pod fără a-l traversa până la celălalt capăt sunt în mod explicit inacceptabile.
- *Euler a demonstrat că problema nu are soluție.*
- Dificultatea cu care s-a confruntat a fost să dezvolte o tehnică adecvată de analiză, și teste ulterioare, care să enunțe această afirmație cu rigoare matematică.

https://ro.wikipedia.org/wiki/Problema_podurilor_din_K%C3%B6nigsberg

Problema celor șapte poduri

- În primul rând, Euler a subliniat că alegerea drumului prin interiorul fiecărei porțiuni de oraș este irelevantă.
- Singura caracteristică importantă a unui traseu șirul de poduri traversate.
- Acest lucru i-a permis să reformuleze problema în termeni abstracți (punând bazele **teoriei grafurilor**), eliminând toate caracteristicile, cu excepția listei de mase de uscat și podurile de legătură.
- În termeni moderni, a înlocuit fiecare masă de uscat cu un „**nod**” sau vârf abstract, iar fiecare pod cu o legătură abstractă, o „**muchie**”, care servește doar pentru a înregistra care pereche de noduri (mase de uscat) este conectată prin pod. Structura matematică rezultată se numește **graf**.



https://ro.wikipedia.org/wiki/Problema_podurilor_din_K%C3%B6nigsberg

6.1. Definiții

- În scopul descrierii unor activități din cadrul unui proces de producție sau a relațiilor existente între elementele unei structuri organizatorice se pot folosi **imagini grafice** gen **diagrame**, **schițe**, **grafice**, etc.
- O reprezentare dintre cele mai utilizate este cea prin **grafuri**.
- Acestea sunt utilizate în special pentru ***vizualizarea sistemelor și situațiilor complexe***.

6.1. Definiții

În general, vom reprezenta:

- componentele acestora prin **puncte** în plan
- relațiile (legăturile, dependențele, influențele, etc) dintre componente prin **arce de curbă** cu extremitățile în punctele corespunzătoare.

6.1. Definiții

Între două puncte pot exista:

- *unul sau mai multe segmente* (în funcție de câte relații dintre acestea, care ne interesează, există)
- iar segmentelor:
 - li **se pot asocia sau nu orientări** (după cum se influențează cele două componente între ele),
 - **numere** care să exprime **intensitatea relațiilor dintre componente** etc.

6.1. Definiții

Definiție

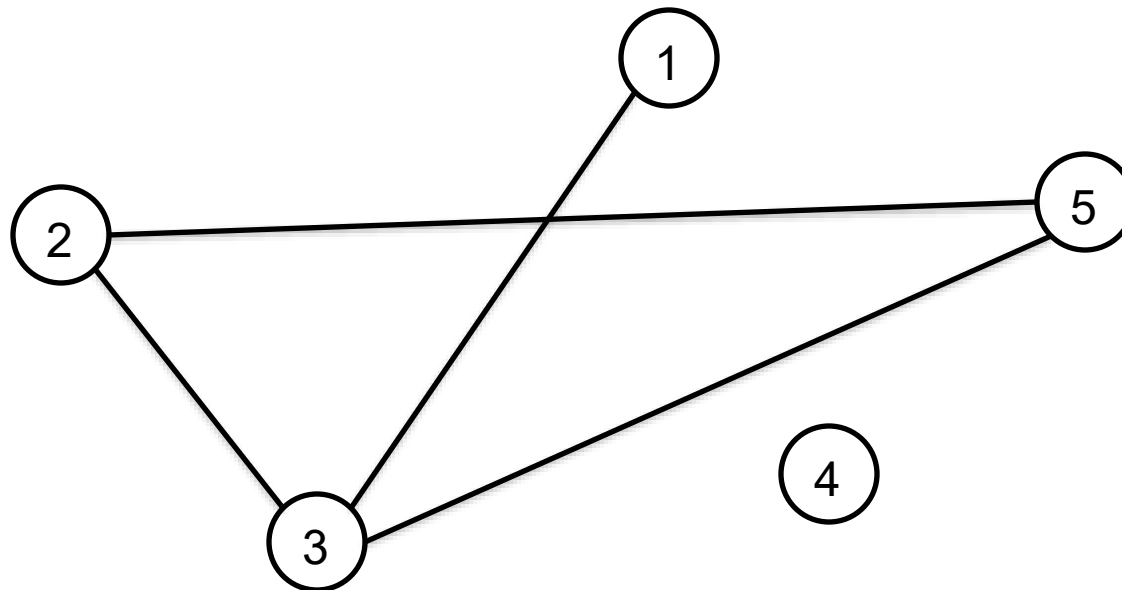
Se numește **graf neorientat** o pereche ordonată de mulțimi (X, U) , X fiind o mulțime finită și nevidă de elemente numite **noduri** sau **vârfuri**, iar U o mulțime de perechi neordonate din X , numite **muchii**.

6.1. Definiții

Exemplu:

$$X = \{1, 2, 3, 4, 5\};$$

$$U = \{(1,3), (2,3), (2,5), (3,5)\};$$



Definiție:

- Două vârfuri legate printr-o muchie se numesc **adiacente**.

Definiție:

- Muchia $[x,y]$ este **incidentă** cu vârful x și vârful y .

Definiție:

- Se numește **gradul unui vârf x** și se notează **$d(x)$** numărul vârfurilor adiacente cu vârful x .

Observație:

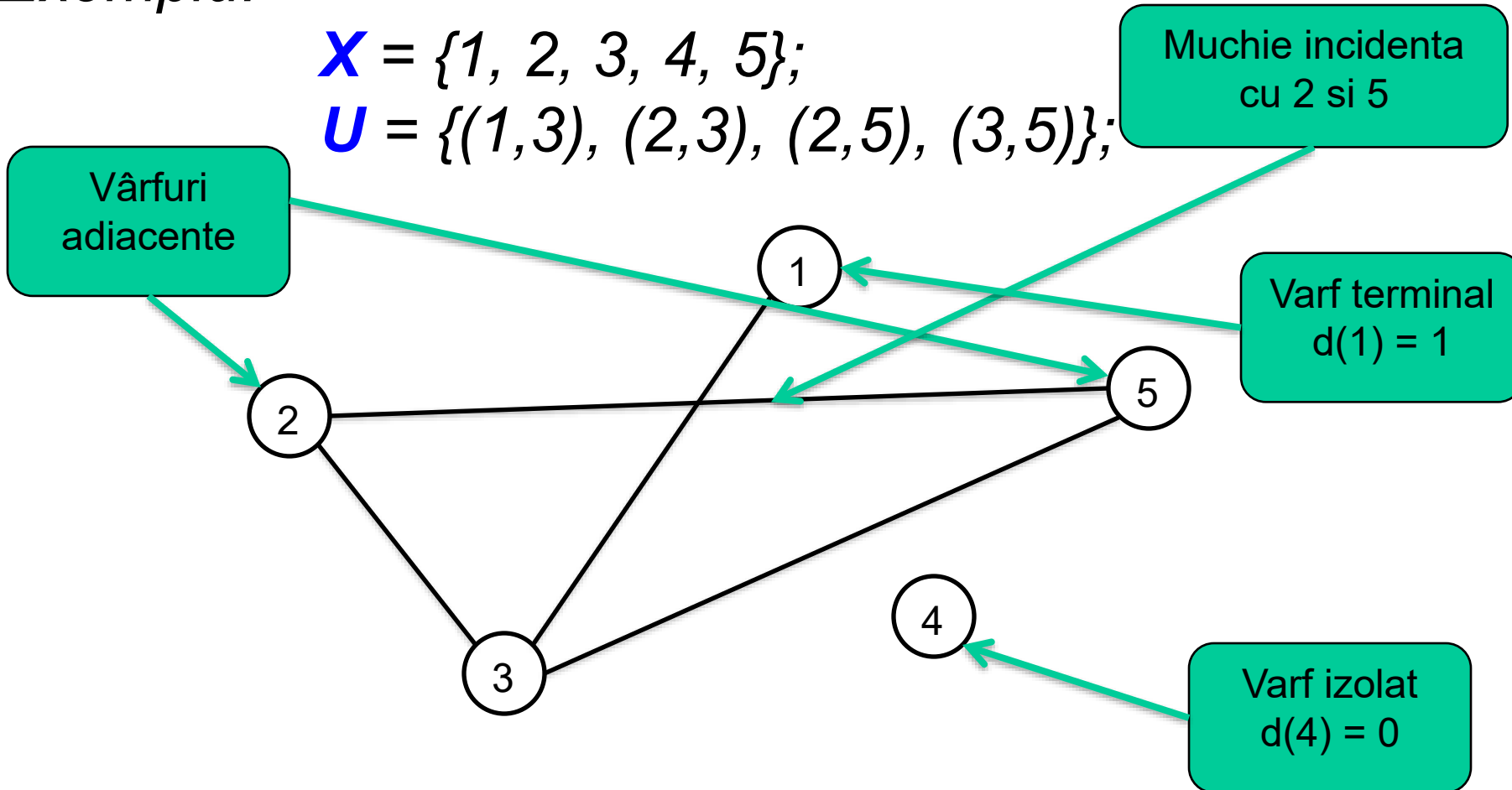
- Vârfurile cu grad 0 se numesc **izolate**
- Vârfurile cu grad 1 se numesc **terminale**

6.1. Definiții

Exemplu:

$$X = \{1, 2, 3, 4, 5\};$$

$$U = \{(1,3), (2,3), (2,5), (3,5)\};$$



6.1. Definiții

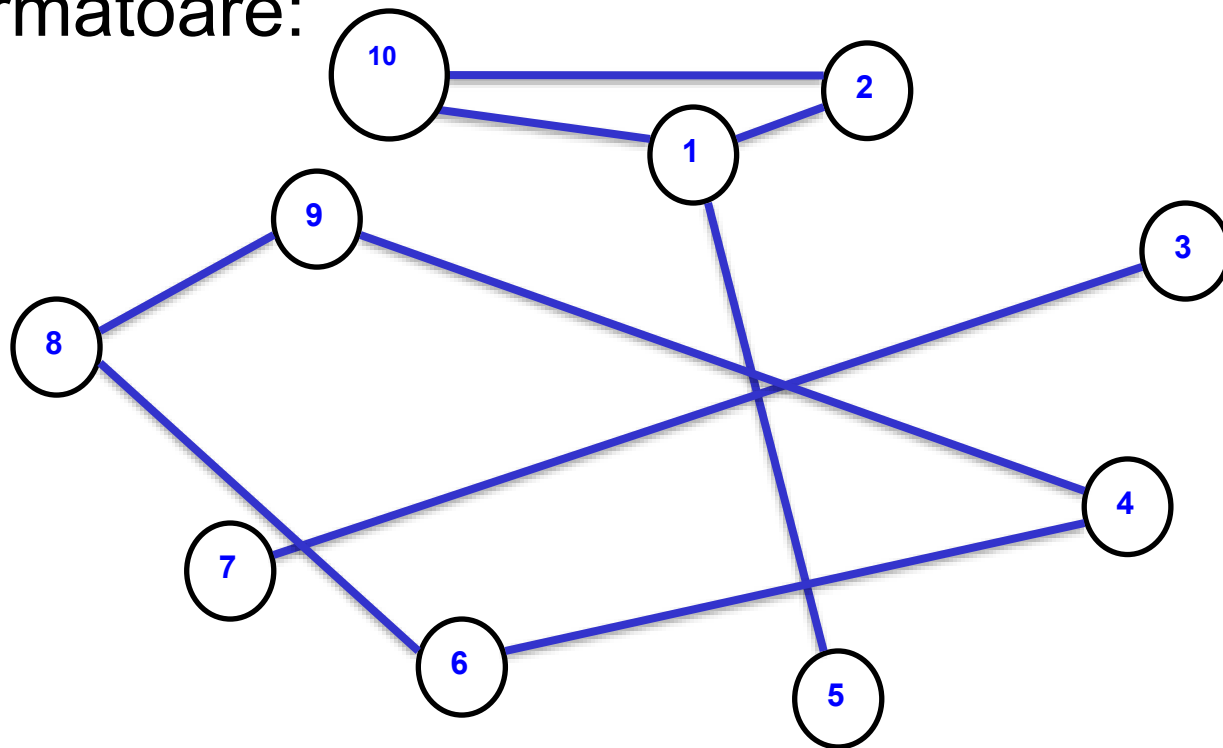
- Un prim element atrăgător al teoriei grafurilor este *aspectul geometric sau grafic* al subiectelor.
- Astfel *un graf poate fi reprezentat cu ajutorul unei figuri plane in care fiecărui vârf i se asociază un punct și fiecărui muchii (x,y) o linie curbă care unește în plan punctele ce corespund vârfurilor x și y .*
- Al doilea aspect interesant este *dezvoltarea naturală a teoriei grafurilor*, de îndată ce definiția unui graf a fost prezentată, noțiunile și rezultatele par să se nască singure și în mod spontan, astfel încât cel care studiază acest domeniu pare să aibă impresia că ar fi putut fi chiar el însuși creatorul acestui domeniu.

Exemplu

Fie $G=(X,U)$ astfel încât $X=(1,2,3,4,5,6,7,8,9,10)$

$U=\{(1,5),(3,7),(4,6),(9,8),(10,2),(1,2),(9,4),(1,10),(6,8)\}$.

Reprezentarea în plan a acestui graf este dată în figura următoare:



6.1. Definiții

TEOREMA:

In orice graf (X,U) suma gradelor varfurilor este de doua ori numarul de muchii, adica

$$\sum d(x) = d(x_1) + d(x_2) + \dots + d(x_n) = 2 * m, x \in X,$$

Unde: m – nr. de muchii, n – nr. de varfuri.

Demonstratie:

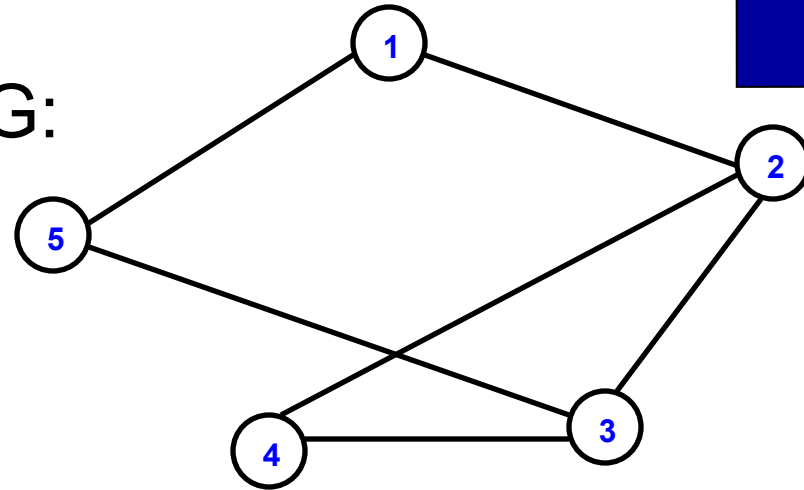
Fiecare muchie (x,y) contribuie cu o unitate la gradul lui x si cu o unitate la gradul lui y , deci contribuie cu 2 unitati la suma gradelor, si atunci fiind m muchii inseamna ca suma gradelor este de 2 ori nr. de muchii.

6.1. Definiții

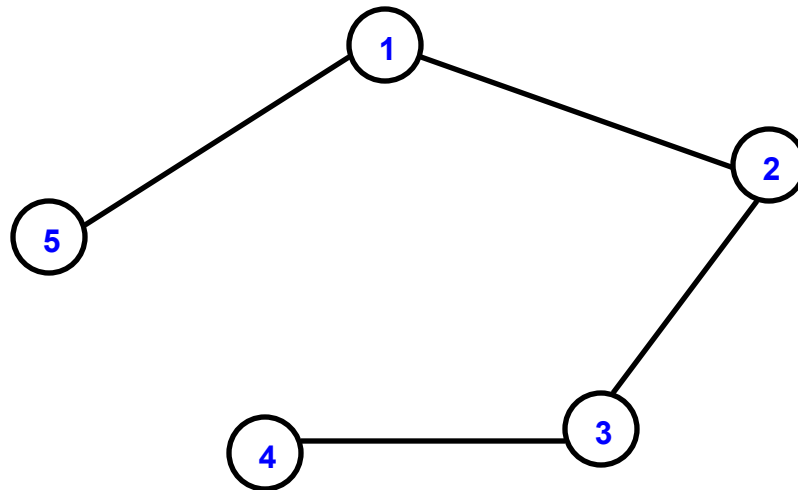
Definiție

- Un **graf parțial** al grafului $G=(X,U)$ este un graf $G_1=(X,V)$ astfel încât $V \subseteq U$, adică G_1 are aceeași mulțime de vârfuri ca G iar mulțimea de muchii V este chiar U sau o submulțime a acesteia.
- Cu alte cuvinte, *un graf parțial al unui graf se obține păstrând aceeași mulțime de vârfuri și eliminând o parte din muchii.*
- Se mai spune că un graf parțial $G_1=(X,V)$ este indus de mulțimea V de muchii.

Exemplu: Fie graful G:



Atunci graful G_1 este **graf partial** pentru graful G

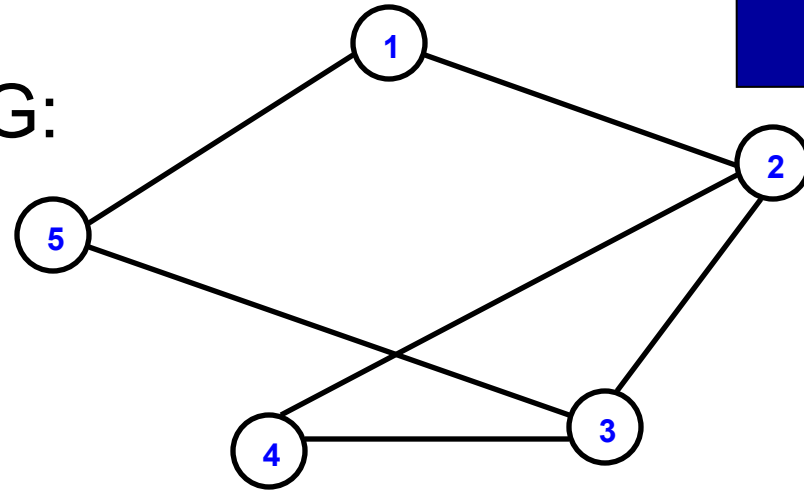


6.1. Definiții

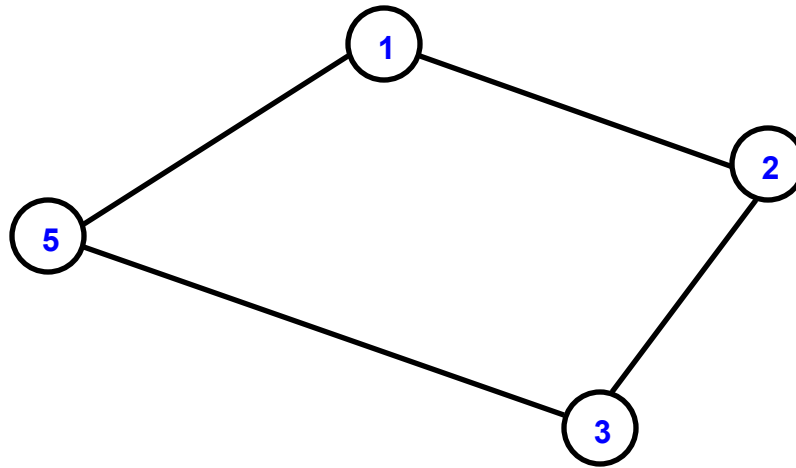
Definiție

- Un **subgraf** al unui graf $G=(X,U)$ este un graf $H=(Y,V)$ astfel încât $Y \subset X$ iar V conține toate muchiile din U care au ambele extremități în Y .
- Prin urmare, *un subgraf al unui graf se obține eliminând o parte din vârfuri și toate muchiile incidente cu acestea.*

Exemplu: Fie graful G:



Atunci graful G_1 este **subgraf** pentru graful G

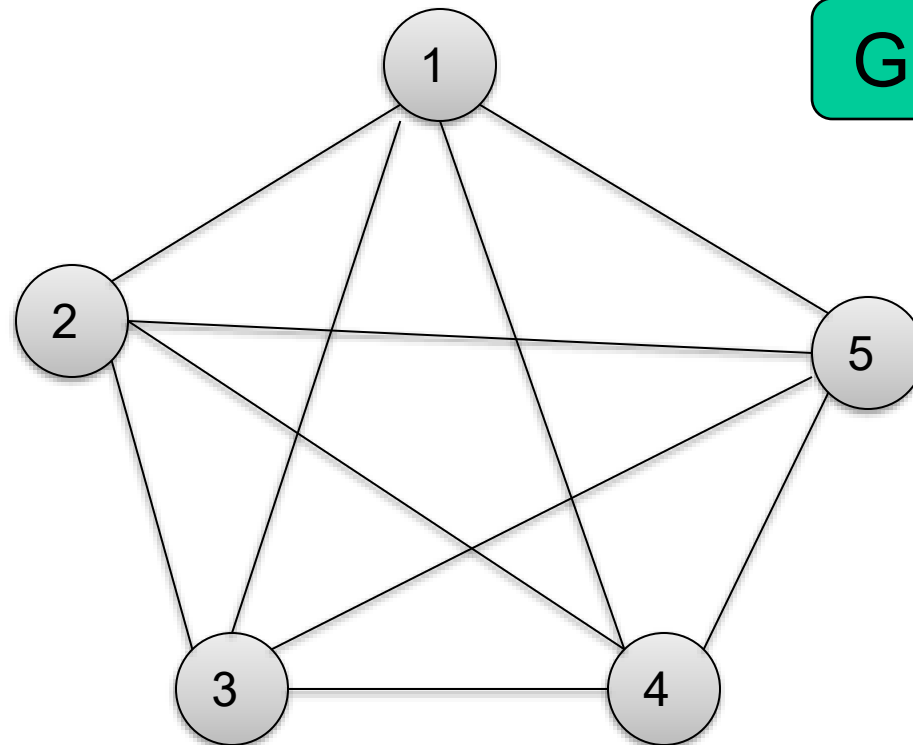


- Există tipuri de grafuri care au proprietăți speciale și care intervin în anumite categorii de probleme și raționamente.
- Ele au denumiri și notații speciale.

Definiție

Se numește **graf complet** cu n vârfuri, un graf care are proprietatea că orice două noduri diferite sunt adiacente.

6.1. Definiții



Graf complet

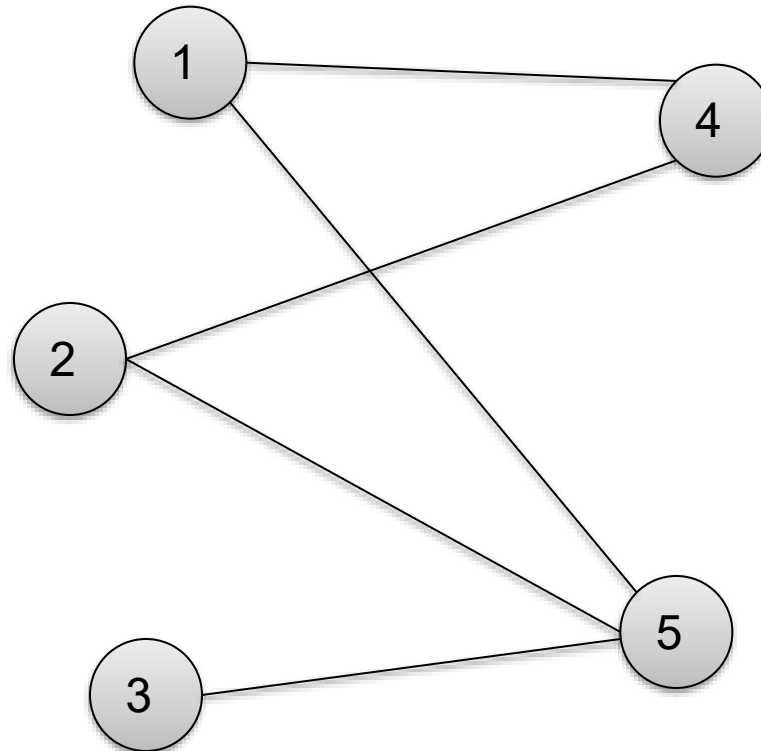
Definiție

*Un graf $G=(X,U)$ se numește **bipartit**, dacă există două mulțimi nevide A și B astfel încât $X=A\cup B$, și orice muchie u a lui G are o extremitate în A iar cealaltă în B .*

Definiție

*Un **graf bipartit** se numește **complet**, dacă pentru orice x din A și orice y din B , există în G muchia $[x,y]$.*

6.1. Definiții



Graf bipartit

Definiție

Se numeste **lanț** în graful G , o succesiune de vârfuri $L=(x_1,x_2,x_3,\dots,x_n)$, unde $x_1, x_2, x_3, \dots, x_n \in X$, cu proprietatea că oricare două vârfuri consecutive sunt adiacente, adică există muchiile $[x_{i-1},x_i]$, cu $i > 1$ și $i < n$.

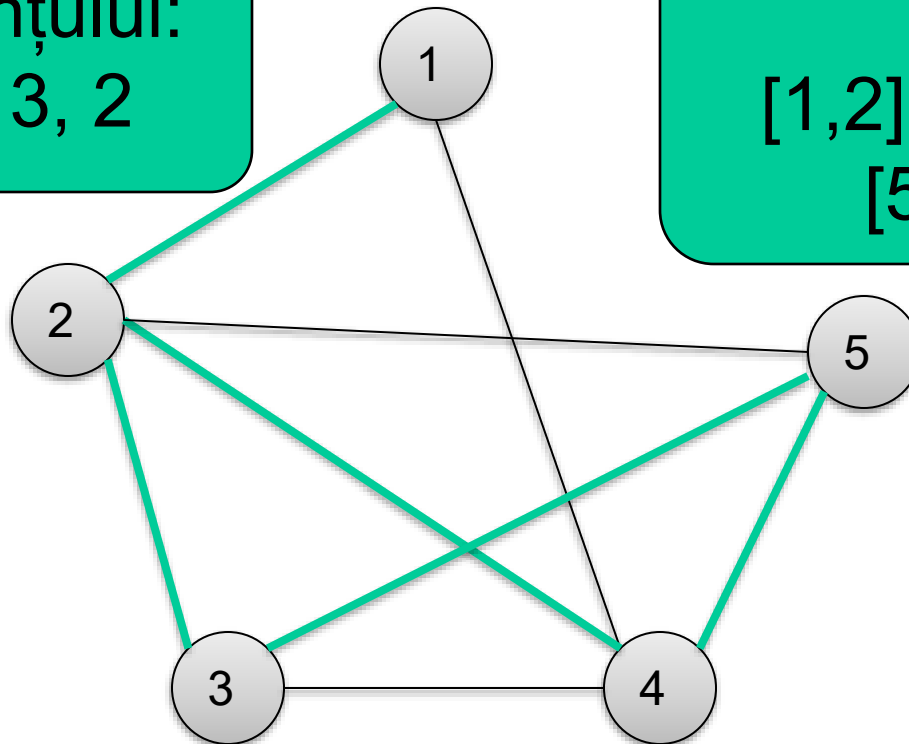
Observație:

Vârfurile x_1 și x_n se numesc **extremitățile lanțului**, iar numărul de muchii care intră în componența lanțului reprezintă **lungimea lanțului**.

6.1. Definiții

Vârfurile lanțului:
1, 2, 4, 5, 3, 2

Lanț
[1,2], [2,4], [4,5],
[5,3], [3,2]



Lungimea lanțului: 5

Observație:

Dacă vârfurile x_1, x_2, \dots, x_n sunt distincte două câte două lanțul se numește **elementar**, în caz contrar se numește **ne-elementar**.

Observație:

Un graf este **conex** dacă între oricare două vârfuri ale sale există un lanț care le leagă.

6.1. Definiții

Definiție

Se numeste **ciclu** într-un graf un lanț $L=(x_1, x_2, x_3, \dots, x_n)$ cu proprietatea că $x_1 = x_n$ și că muchiile $[x_{i-1}, x_i]$ sunt distincte două câte două.

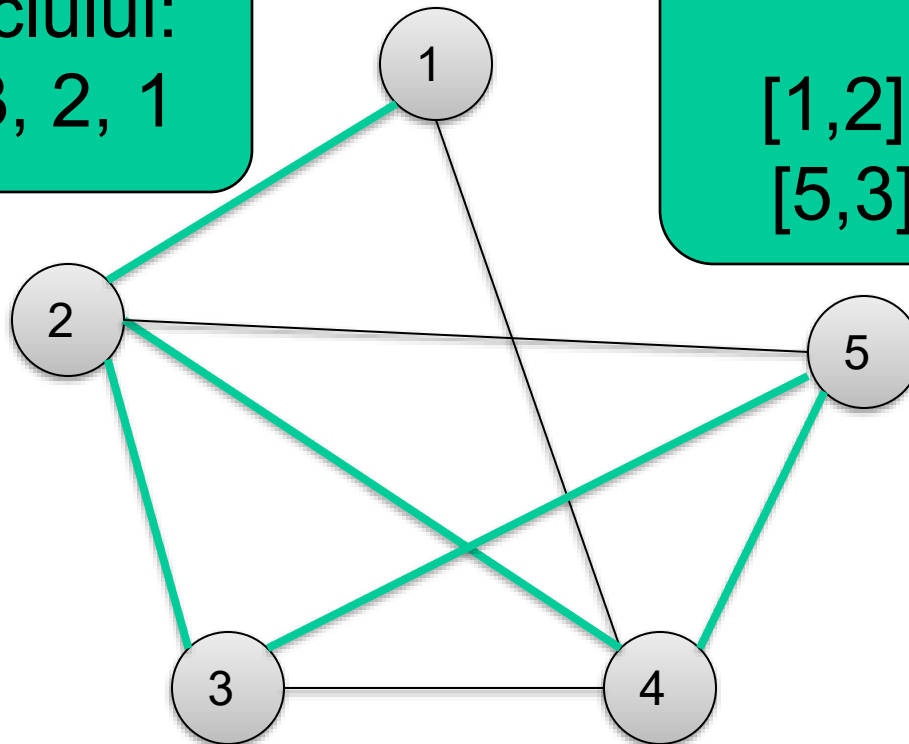
Observație:

Dacă vârfurile într-un ciclu cu excepția primului și ultimului sunt distincte două câte două, atunci ciclul se numește **elementar**, altfel **ne-elementar**.

6.1. Definiții

Vârfurile ciclului:
1, 2, 4, 5, 3, 2, 1

Ciclu:
[1,2], [2,4], [4,5],
[5,3], [3,2], [2,1]



Conținutul cursului

6.1. Definiții

6.2. Memorarea(reprezentarea) grafurilor

6.3. Parcurgerea grafurilor

6.2. Memorarea(reprezentarea) grafurilor

- Fie $G=(X,U)$ un graf neorientat.
- Deoarece între mulțimea X cu n elemente și mulțimea $\{1,2,\dots,n\}$ există o legătură directă, atunci există mai multe modalități de a reprezenta graful G , pentru care se folosesc diverse structuri de date.
- *Reprezentările vor fi utilizate în algoritmi care prelucrează grafuri, deci și în programele care implementează pe calculator aceiași algoritmi.*

6.2. Memorarea(reprezentarea) grafurilor

Selectarea uneia sau a alteia dintre structurile de date ce vor fi prezentate este în funcție de problema și de algoritmul ales pentru rezolvarea ei:

- 1) Se precizează numărul n de vârfuri și **matricea de adiacență** A a grafului, care este o matrice pătratică de ordinul n .
- 2) Se precizează numărul n de vârfuri și, pentru fiecare vârf i , **lista L_i a vecinilor săi**, adică lista vârfurilor j pentru care $[i,j] \in U$.

6.2. Memorarea(reprezentarea) grafurilor

3) Se dau numărul n de vârfuri, numărul m de muchii, precum și două tablouri unidimensionale e_1 și e_2 cu câte m componente fiecare, conținând extremitățile muchiilor grafului, adică

$$U = \{ (e_1[1], e_2[1]), (e_1[2], e_2[2]), \dots, (e_1[m], e_2[m]) \}$$

O variantă de implementare mai naturală ar fi aceea de a defini un tip de dată *muchie* utilizând tipul **struct**, care să conțină cele două extremități ale unei muchii, și apoi de a defini un tablou cu n componente de acest tip:

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

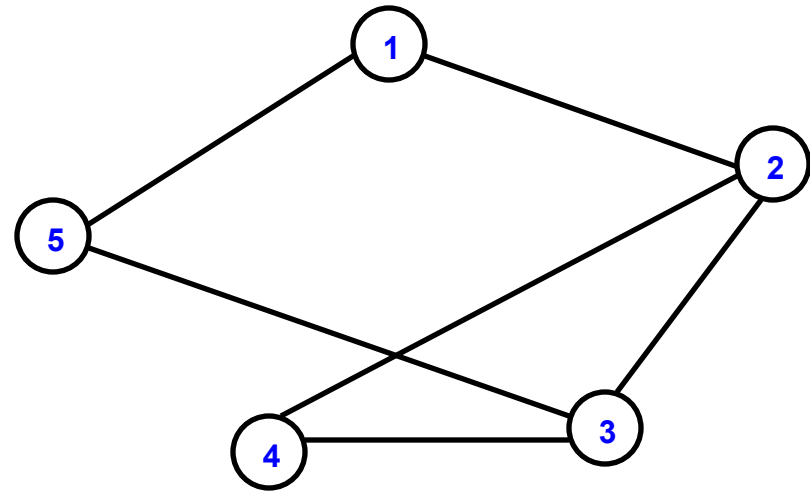
.....

```
struct muchie u[20];
```

- Referirea la extremitățile muchiei i se face prin:
 - $u[i].x$
 - respectiv $u[i].y$
- Acest mod de reprezentare permite înglobarea naturală în tipul de date *muchie* și a altor informații asociate muchiilor și este utilizată în problemele în care muchiile se prelucrează succesiv, eventual după o modificare a ordinii lor în tabloul u .

1) Reprezentarea grafurilor folosind matricea de adiacență:

Fie graful urmator:



Graful are $n=5$ vârfuri

Matricea de adiacenta corespunzatoare grafului este:

$$A = \begin{pmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \end{pmatrix}$$

Citirea unui graf memorat cu ajutorul matricei de adiacență

Varianta 1:

Citirea numărului de varfuri, **citirea regiunii de deasupra diagonalei principale** urmată de **simetrizare**.

```
#include<iostream>
using namespace std;
int main(void)
{
    int a[20][20], i, j, n;
    cin>>n;
    for(i=1; i<=n; i++) a[i][i] = 0;

    for(i=1; i<=n-1; i++)
        for(j=i+1; j<=n; j++)
        {
            cin>>a[i][j];
            a[j][i] = a[i][j];
        }
    for(i=1; i<=n; i++)
        {
            for(j=1; j<=n; j++)
                cout<<a[i][j]<<" ";
            cout<<"\n";
        }
    return 0;
}
```

Varianta 2:

Se citesc n - nr de varfuri, m - nr de muchii;

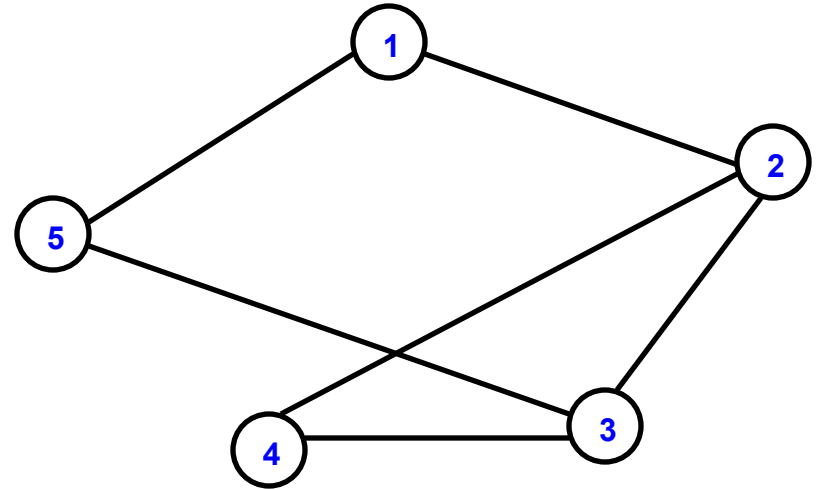
- initializarea matricei de adiacenta cu zero
- citirea extremitatii fiecarei muchii
- completarea matricei de adiacenta

```
#include<iostream>
using namespace std;
int a[20][20], i, j, n, m, x, y;
int main(void)
{
    cin>>n>>m;
    for(i=1; i<=n; i++)
    for(j=1; j<=n; j++) a[i][j] = 0;
    for(i=1; i<=m; i++)
    {
        cin>>x; cin>>y;
        a[x][y] = 1;
        a[y][x] = 1;
    }
}
```

```
for(i=1; i<=n; i++)
{
    for(j=1; j<=n; j++)
        cout<<a[i][j]<<" ";
    cout<<"\n";
}
return 0;
}
```

2) Reprezentarea grafurilor folosind listele de vecini ale fiecarui varf:

Fie graful urmator:



Graful are $n=5$ vârfuri

Listele de vecini corespunzatoare grafului sunt:

Nodul 1: are vecini pe 2,5

Nodul 2: are vecini pe 1,3,4

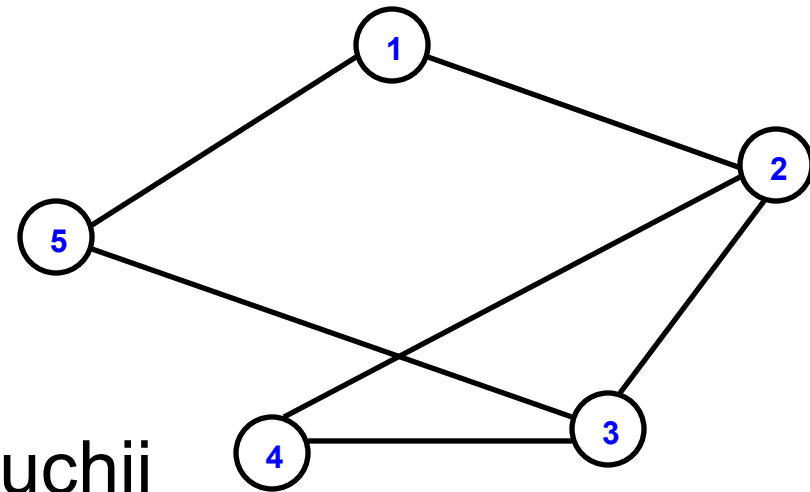
Nodul 3: are vecini pe 2,4,5

Nodul 4: are vecini pe 2,3

Nodul 5: are vecini pe 1,3

3) *Reprezentarea grafurilor folosind doua tablouri unidimensionale in care se retin extremitatile fiecarei muchii din graf:*

Fie graful urmator:



Graful are $n=5$ vârfuri si $m=6$ muchii

Cele tablouri contin elementele:

$$e_1 = \{1, 1, 2, 2, 3, 3\}$$

$$e_2 = \{2, 5, 3, 4, 4, 5\}$$

Atunci graful este format din multimea de muchii:

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

Conținutul cursului

6.1. Definiții

6.2. Memorarea(reprezentarea) grafurilor

6.3. Parcurgerea grafurilor

Probleme cu grafuri neorientate

1. Memorarea grafurilor neorientate folosind cele trei modalitati de reprezentare:

- a) Matrice de adiacenta
- b) Vector de muchii
- c) Liste de vecini

Probleme cu grafuri neorientate

Solutie

```
#include <iostream.h>
```

```
struct muchie{  
    int i,j;  
};
```

```
int A[50][50];           // matrice de adiacenta  
muchie M[1000];         // vector muchiilor  
int V1[50], V2[50];     // liste de vecini  
int n,m;                 // n – nr. de noduri, m – nr. de muchii
```

Probleme cu grafuri neorientate

Solutie (continuare)

```
void citire()
{
    int x,y,i;
    cin>>n>>m;
    for(i=1;i<=m;i++)
    {
        cin>>x>>y;
        M[i].i=x; M[i].j=y;
        A[x][y]=A[y][x]=1;
        V1[i]=x;
        V2[i]=y;
    }
}
```

Probleme cu grafuri neorientate

Solutie (continuare)

```
void afisare()
{
    int i,j;
    cout<<"matricea de adiacenta:\n";
    for(i=1;i<=n;i++)
    {
        for(j=1;j<=n;j++)
            cout<<A[i][j]<<" ";
        cout<<endl;
    }
}
```

```
    cout<<"lista muchiilor:\n";
    for(i=1;i<=m;i++)
        cout<<M[i].i<<" "<<M[i].j<<endl;
    cout<<"lista vecinilor:\n";
    for(i=1;i<=n;i++)
    {
        cout<<i<<": ";
        cout<<V1[i]<<","<<V2[i]<<" ";
        fcout<<endl;
    }
}
```

Probleme cu grafuri neorientate

Solutie (continuare)

```
int main()
{
    citire();
    afisare();
}
```

Probleme cu grafuri neorientate

2. Se citește dintr-un fișier de pe primul rând o valoare n reprezentând numărul de noduri pentru un graf neorientat și de pe următoarele n linii și coloane matricea de adiacență corespunzătoare grafului.

- a) Identificați mulțimea X
- b) Identificați mulțimea U
- c) Calculați gradele nodurilor impare
- d) Verificați dacă graful are varfuri izolate, dacă da afișați nodul, dacă nu afișați un mesaj

Întrebări?