

Laborator 4
Grafuri neorientate

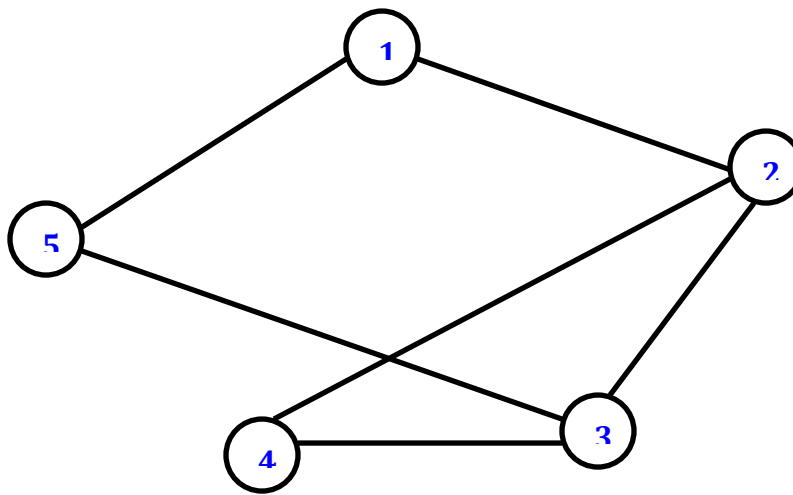
1. Graf partial

Fie $G=(A,B)$ si $G_1=(A_1,B_1)$.

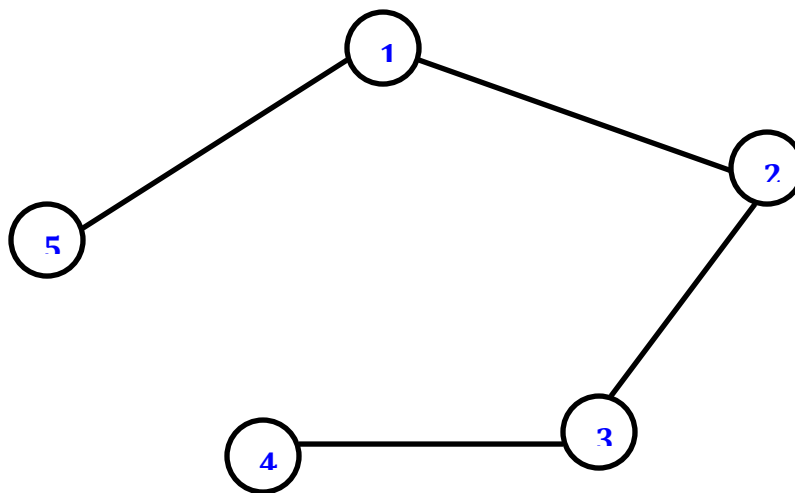
Spunem că G_1 este un graf parțial al lui G dacă $A=A_1$ și B_1 este inclus sau egal cu B .

Un graf parțial se obține dintr-un graf, îndepărtând o parte dintre muchiile sale și păstrând toate nodurile acestuia.

Exemplu 1:



Graful inițial



2. Subgraful unui graf

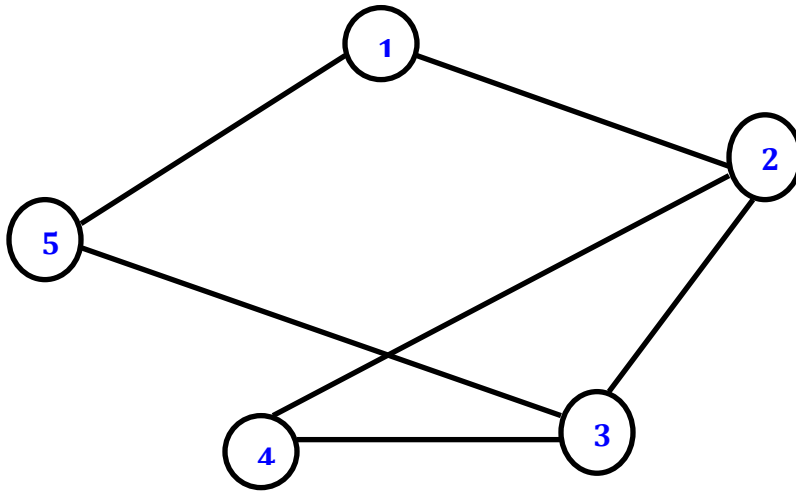
Fie $G=(A,B)$ și $G_1=(A_1,B_1)$;

A_1 inclus sau egal cu A ; B_1 inclus sau egal cu B .

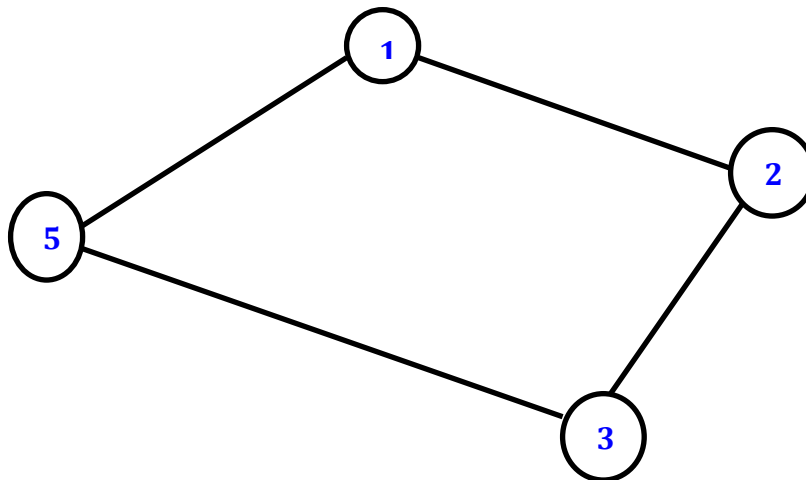
$B_1 = \{(x,y) / \text{oricare } x,y \text{ aparține } A_1 \text{ dacă } (x,y) \text{ aparține de } B \Rightarrow (x,y) \text{ aparține de } B_1\}$

Subgraful se obține din graful inițial selectând o parte din nodurile sale și o parte din nodurile adiacente cu acesta.

Exemplu 2:



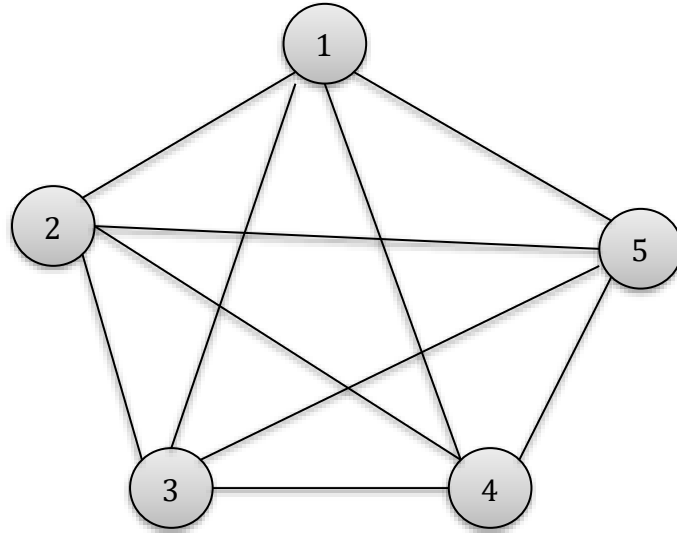
Graful inițial



3. Graf complet

Un graf este complet dacă oricare două vârfuri distince sunt adiacente.

Exemplu 3:



Un graf neorientat cu n noduri are $n(n-1)/2$ muchii.

Există un singur graf complet neorientat cu n noduri.

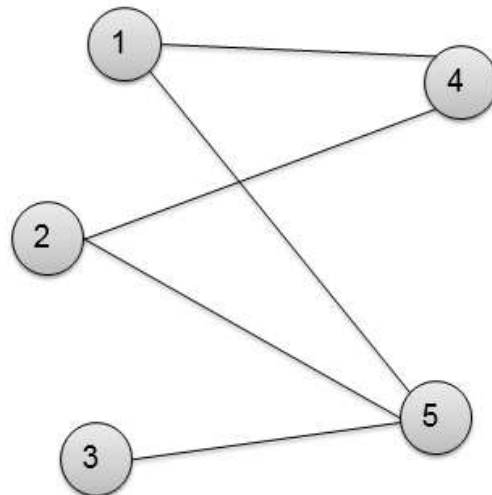
Există mai multe grafuri orientate complete cu n noduri.

4. Grafuri bipartite

Fie $G=(A,B)$ neorientat.

G este bipartit dacă există două mulțimi, A_1 și A_2 astfel încât $A_1 \cap A_2 = \emptyset$ și $A_1 \cup A_2 = A$, iar oricare muchie (x,y) aparținând lui B are un capăt în mulțimea A_1 și celălalt în A_2 .

Exemplu 4:



Un graf bipartit este bipartit complet dacă fiecare nod din mulțimea A_1 este adiacent cu toate nodurile din A_2 și reciproc.

5. Grafuri conexe

Un graf este conex dacă este format dintr-un singur nod sau dacă între oricare două noduri ale sale există cel puțin un lanț.

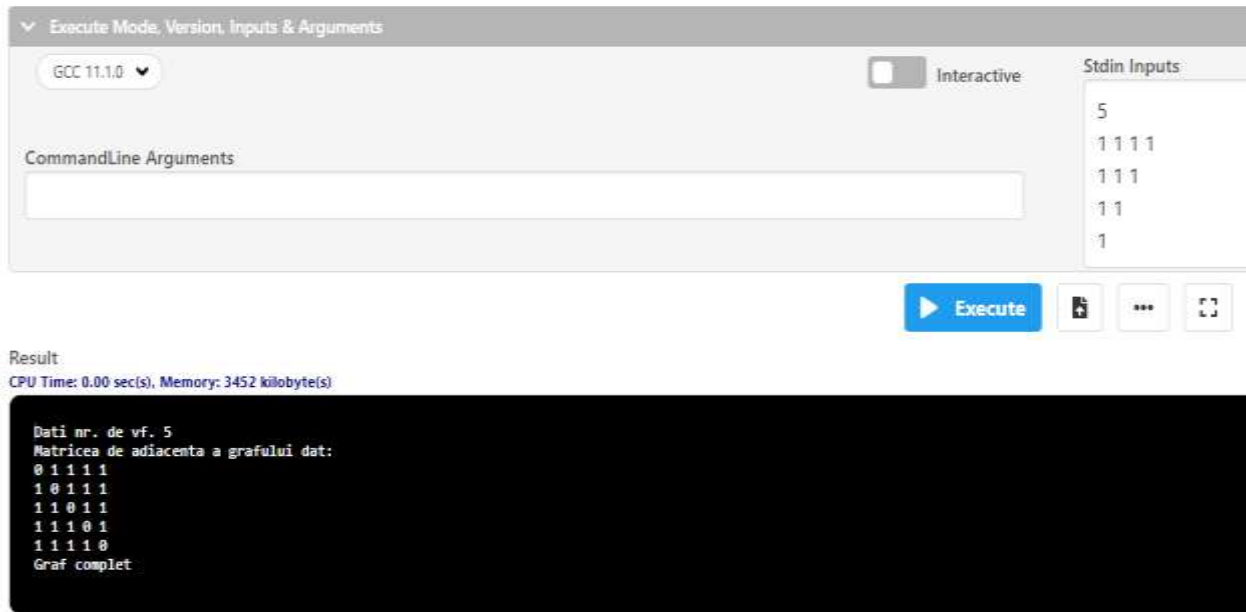
Probleme rezolvate

1) Sa se verifice daca un graf neorientat este complet.

```
#include<iostream>
using namespace std;

int n,i,j,m,a[10][10];
int main()
{
cout<<"Dati nr. de vf. ";
cin>>n;
cout<<n;
for(i = 1; i <= n - 1; i++)
    for(j = i + 1; j <= n; j++)
    {
        cin>>a[i][j];
        a[j][i] = a[i][j];
    }
cout<<"\nMatricea de adiacenta a grafului dat: \n";
for(i = 1; i <= n; i++)
{
    for(j = 1; j <= n; j++)        cout<<a[i][j]<<" ";
    cout<<"\n";
}
// determinarea nr. de muchii
m = 0;
for(i = 1; i <= n; i++)
    for(j = 1; j <= n; j++)
        if(a[i][j] == 1) m++;
m = m / 2;
if(m == n * (n - 1) / 2)        cout<<"Graf complet";
else        cout<<"Graful nu este complet";
    return 0;
}
```

Executia programului folosind compilatorul online <https://www.jdoodle.com/online-compiler-c++/>:



2) Sa se verifice daca o succesiune de varfuri date este ciclul pentru un graf. In caz afirmativ, sa se stabileasca daca este ciclul elementar sau nu.

```
#include<iostream>
using namespace std;

int n,i,j,m,a[10][10],x[50];
int main()
{
cout<<"\nDati nr. de vf. ";
cin>>n;cout<<n;
//cout<<"Dati elementele matricei de adiacenta"<<endl;
for(i=1;i<=n-1;i++)
    for(j=i+1;j<=n;j++)
    {
        cin>>a[i][j];
        a[j][i]=a[i][j];
    }
// citirea succesiunii de vf.
cout<<"\nDati nr. de muchii ";
cin>>m;cout<<m;
for(i = 1; i <= m + 1; i++)      cin>>x[i];

// verificare daca este lant: vf. consecutive sa fie muchii in matrice
int lant=1;
for(i=1;i<=n-1;i++)
    if(a[x[i]][x[i+1]]==0) lant=0;
// primul si ultimul vf. sunt egale
```

```

int ok=1;
if(lant==1)
    if(x[1]!=x[m]) ok=0;

// muchii distincte
int ciclu=1;
if(ok==1)
    for(i=1;i<=m-2;i++)
        for(j=i+1;j<=m;j++)
            if(x[i]==x[j] && x[i+1]==x[j+1] || x[i]==x[j+1] && x[i+1]==x[j]) ciclu=0;
// verificare daca este ciclu elementar: vf. sa fie distincte doua cate doua, mai putin primul si
// ultimul
cout<<'\\n';
for(i = 1; i <= m + 1; i++)    cout<<x[i]<<" ";
if(ciclu==1 && x[1]==x[m]) cout<<" este Ciclu elementar";
else    cout<<" este Ciclu neelementar";
    return 0;
}
    
```

Executia programului pentru urmatoarele date:

Dati nr. de vf. 5

Dati elementele matricei de adiacenta

```

a[1][1]=0
a[1][2]=1
a[1][3]=0
a[1][4]=1
a[1][5]=0
a[2][2]=0
a[2][3]=1
a[2][4]=1
a[2][5]=1
a[3][3]=0
a[3][4]=1
a[3][5]=1
a[4][4]=0
a[4][5]=1
    
```

Dati nr. de muchii 6

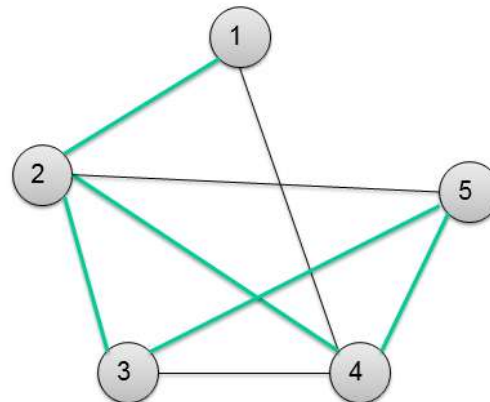
```

x[1]=1
x[2]=2
x[3]=4
x[4]=5
x[5]=3
x[6]=2
x[7]=1
    
```

Ciclu neelementar

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

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



Executia programului folosind compilatorul online <https://www.jdoodle.com/online-compiler-c++/>:



3) Se citesc 2 grafuri neorientate, unul cu n noduri si m muchii, iar celalalt cu k varfuri si l muchii, ambele date prin vectorul muchiilor. Sa se determine daca al doilea graf este subgraf al primului.

```
#include<iostream>
using namespace std;

int a[100][100], b[100][100], n,m,k,l, i,j;

int subgraf()
{
    for(int i=1;i<=k;i++)
        for(int j=1;j<=l;j++)
            if(a[i][j]!=b[i][j]) return 0;
    return 1;
}

int main(void)
{
    int x,y,i;
    cin>>n>>m;
    cout<<"\nGraful G are "<<n<<" varfuri si "<<m<<" muchii";
    for(i=1;i<=m;i++)
    {
        cin>>x>>y;
        a[x][y]=1;
        a[y][x]=1;
    }
    cout<<"\n";
    for(i=1;i<=n;i++)
```

```

    {
        for(j=1;j<=n;j++) cout<<a[i][j]<<" ";
        cout<<'\n';
    }
    cin>>k>>l;
    cout<<"\nGraful H are "<<k<<" varfuri si "<<l<<" muchii";
    for(i=1;i<=l;i++)
    {
        cin>>x>>y;
        b[x][y]=1;
        b[y][x]=1;
    }
    cout<<'\n';
    for(i=1;i<=n;i++)
    {
        for(j=1;j<=n;j++) cout<<b[i][j]<<" ";
        cout<<'\n';
    }
    if(subgraf()) cout<<"\nGraful H este subgraf al grafului G";
    else cout<<"\nGraful H NU este subgraf al grafului G";
    return 0;
}

```

Executia programului folosind compilatorul online <https://www.jdoodle.com/online-compiler-c++/>:



- 4) Memorarea grafurilor neorientate folosind cele trei modalități de reprezentare:
- a) Matrice de adiacență
 - b) Vector de muchii
 - c) Liste de vecini

```
#include<iostream>
using namespace std;

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

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;
    }
}

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]<<" ";
    }
}
```

```
        cout<<endl;
    }
}

int main()
{
    citire();
    afisare();
    return 0;
}
```

Executia programulu folosind compilatorul online <https://www.jdoodle.com/online-compiler-c++/>:

Execute Mode, Version, Inputs & Arguments

GCC 11.1.0 Interactive Stdin Inputs

5
7
1
2
1
3
1
5
2
3
2
4
3
4
4
5

CommandLine Arguments

Execute

Result
CPU Time: 0.00 sec(s), Memory: 3372 kilobyte(s)

```
matricea de adiacenta:
0 1 1 0 1
1 0 1 1 0
1 1 0 1 0
0 1 1 0 1
1 0 0 1 0
lista muchilor:
1 2
1 3
1 5
2 3
2 4
3 4
4 5
lista vecinilor:
1: 1,2
2: 1,3
3: 1,5
4: 2,3
5: 2,4
```

PROIECTAREA ALGORITMILOR

Laborator 4

5) Se citește o valoare n reprezentând numărul de noduri pentru un graf neorientat și de pe următoarele n linii și n coloane matricea de adiacență corespunzătoare grafului.

- Identificați mulțimea X
- Identificați mulțimea U
- Calculați gradele nodurilor impare
- Verificați dacă graful are vârfuri izolate, dacă da afișați nodul, dacă nu afișați un mesaj

```
#include<iostream>
using namespace std;
int a[20][20];
int n,i,j,k;
int main()
{
    cin>>n;
    //a)
    for(i = 1; i <= n; i++)
        for(j = 1; j <= n; j++)
            cin>>a[i][j];
    cout<<"\nX = ";
    for(i = 1; i <= n; i++) cout<<i<<" ";
    cout<<endl;

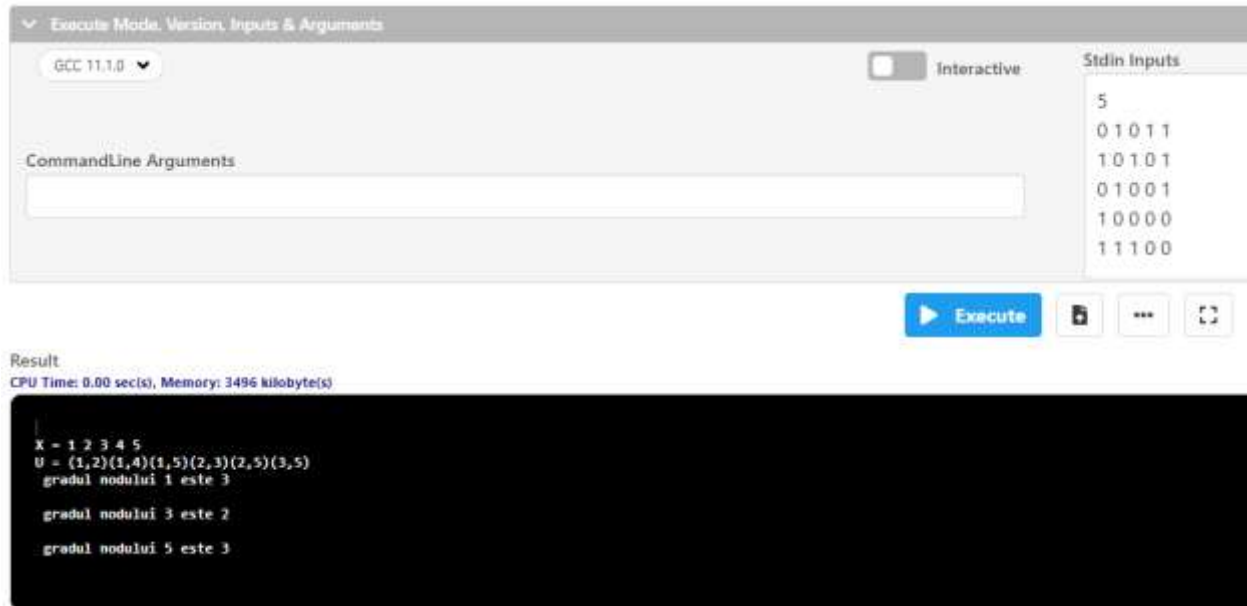
    //b)
    cout<<"U = ";
    for(i = 1; i <= n; i++)
        for(j = 1; j <= n; j++)
            if( a[i][j] == 1 && i < j ) cout<<"("<<i<<","<<j<<")";

    //c)
    for(i = 1; i <= n; i++)
    {
        k = 0;
        for(j = 1; j <= n; j++)
            if(i % 2 == 1 && a[i][j] == 1) k++;
        if(i % 2 == 1) cout<<"\n gradul nodului "<<i<<" este "<<k<<endl;
    }

    //d)
    for(i=1;i<=n;i++)
    {
        for(j=1;j<=n;j++)
            if(a[i][j] == 1) return 0;
    }

    return 0;
}
```

Executia programului folosind compilatorul online <https://www.jdoodle.com/online-compiler-c++/>:



- 6) Se citește n numărul de vârfuri și m numărul de muchii(n, m) de pe următoarele m randuri perechi de varfuri reprezentand muchiile grafului.
Se cere:
- a) Să se construiască matricea de adiacență și sa se afișeze
 - b) Să se calculeze gradul fiecărui nod și păstrați acest grad într-un vector
 - c) Să se verifice dacă graful are varfuri izolate

```
#include<iostream>
using namespace std;

int a[20][20],v[40];
int j,i,n,m,h,r=0,x,y,ok=0;

int main()
{
    cin>>n>>m;

    // a)
    for(i=1;i<=m;i++)
    {
        cin>>x>>y;
        a[x][y]=a[y][x]=1;
    }
    for(i=1;i<=n;i++)
    {
```

```

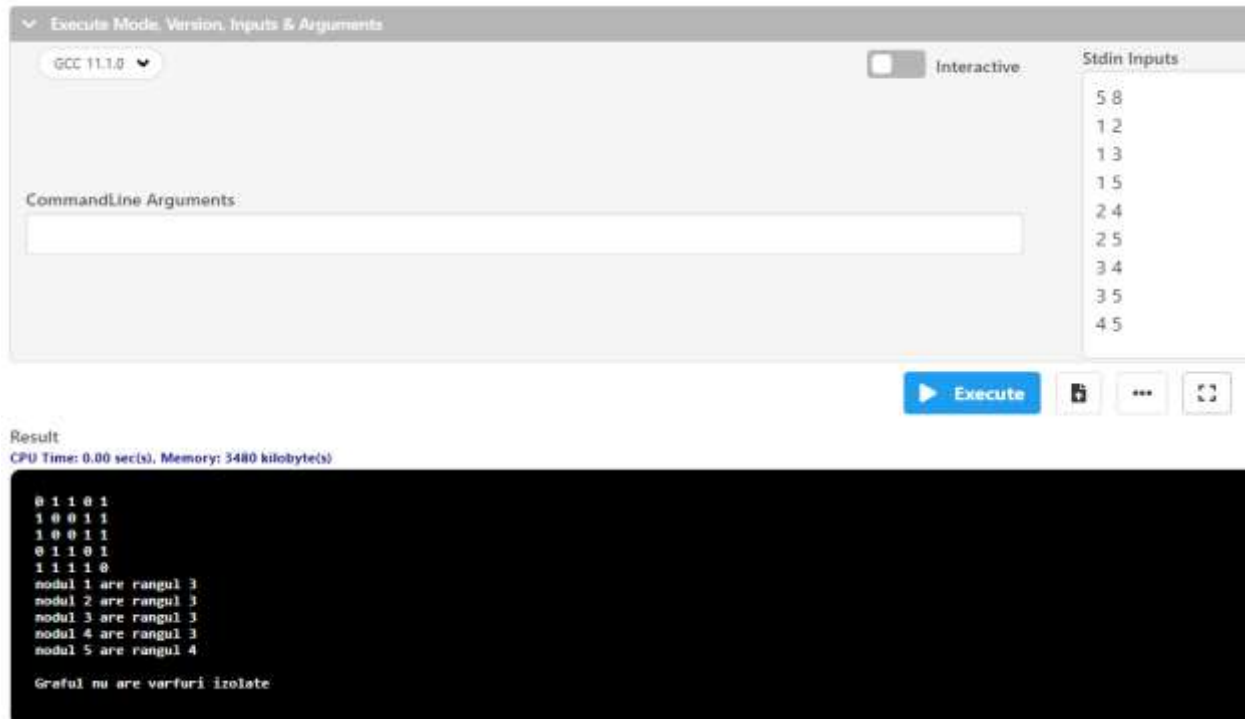
    for(j=1;j<=n;j++) cout<<a[i][j]<<" ";
    cout<<endl;
}

//b)
for(i=1;i<=n;i++)
{
    h=0;
    for(j=1;j<=n;j++)
        if(a[i][j]==1) h++;
    r++;
    v[r]=h;
}
for(i=1;i<=n;i++)
    cout<<"nodul "<<i<<" are rangul "<<v[i]<<endl;

// c)
for(i=1;i<=n;i++)
    if(v[i]==0) ok=0;
    else ok=1;
if(ok==0) cout<<"\nGraful are varfuri izolate"<<" ";
else cout<<"\nGraful nu are varfuri izolate";
return 0;
}

```

Executia programului folosind compilatorul online <https://www.jdoodle.com/online-compiler-c++/>:



7) Din fisierul GRAF.TXT, cititi de pe primul rand nr de noduri si nr de muchii, si de pe urmatoarele m randuri perechiile de noduri reprezentand muchiile. Formati matricea de adiacenta, enumerati nodurile terminale, nodurile intermediare

```
#include<iostream>
using namespace std;

int a[20][20];
int i,j,n,m,x,y,k;
int main()
{
    cin>>n>>m;
    for(i=1;i<=n;i++)
        for(j=1;j<=n;j++)
        {
            cin>>x>>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<<endl;
    }
    for(i=1;i<=n;i++)
    {
        k=0;
        for(j=1;j<=n;j++)
            if(a[i][j]==1) k++;
        if(k==1) cout<<"\nNod terminal = "<<i;
        if(k>1) cout<<"\nNod intermediar = "<<i;
    }
    return 0;
}
```

Executia programului folosind compilatorul online <https://www.jdoodle.com/online-compiler-c++/>:

Execute Mode, Version, Inputs & Arguments

GCC 11.1.0 Interactive

Stdin Inputs

- 5 8
- 1 2
- 1 3
- 1 5
- 2 4
- 2 5
- 3 4
- 3 5
- 4 5

CommandLine Arguments

Execute   

Result

CPU Time: 0.00 sec(s), Memory: 3492 kilobyte(s)

```
0 1 1 0 1
1 0 0 1 1
1 0 0 1 1
0 1 1 0 1
1 1 1 1 0

Nod intermediar = 1
Nod intermediar = 2
Nod intermediar = 3
Nod intermediar = 4
Nod intermediar = 5
```

Probleme propuse spre rezolvare

1. Să se ruleze programele prezentate mai sus, urmărind apelurile și valorile parametrilor de apel.
2. Se citește un graf de la tastatură : numărul de noduri, numărul de muchii și muchiile.
 - a) să se afișeze matricea de adiacențe
 - b) să se determine gradul unui nod citit
 - c) să se afișeze pentru un nod citit nodurile adiacente
 - d) să se afișeze nodurile incidente cu cea de a x muchie din matrice
 - e) să se afișeze pentru fiecare nod gradul
 - f) să se afișeze nodul (nodurile) având cei mai mulți vecini
 - g) să se afișeze nodurile izolate
3. Să se determine dacă o matrice citită de la tastatură poate fi matricea unui graf neorientat. În caz afirmativ se va determina câte muchii are graful.
4. Să se facă un program de numărare a gradelor fiecărui vârf dintr-un graf. Să se determine apoi:
 - a) dacă există vârfuri izolate și vârfuri terminale;
 - b) toate vârfurile de grad maxim.
5. Să se determine toate grafulile parțiale ale unui graf.
6. Două grafuri $G=(X,U)$ și $H=(Y,V)$ se numesc izomorfe dacă există o bijecție $f: X \rightarrow Y$ astfel încât $[x,y] \in U$ dacă și numai dacă $[f(x),f(y)] \in V$. Deci două grafuri izomorfe au același număr de vârfuri și se obțin unul din celălalt printr-o renumerotare a vârfurilor. Să se facă un program care să verifice dacă două grafuri sunt izomorfe.
7. Fiind dat un graf $G = (X, U)$, să se determine un lanț simplu de lungime maximă.
8. Fiind dat un graf $G = (X, U)$, să se facă un program care să verifice dacă graful este bipartit complet.
9. Să se facă un program de generare a tuturor ciclurilor unui graf. Observație: puteți adapta programul realizat pentru generarea tuturor ciclurilor elementare.
10. Fie G un graf neorientat, cu n vârfuri și m muchii, reprezentat prin matricea de adiacență. Să se realizeze programe, în C/C++, care:
 - a) afișează gradele tuturor vârfurilor;
 - b) afișează vârfurile de grad par;
 - c) afișează vârfurile izolate;
 - d) afișează 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) afișează gradul unui vârf dat;
- k) afișează vecinii unui nod dat, vf;
- l) verifică dacă un vârf dat este terminal, izolat sau interior;
- m) afișează gradul cel mai mare și toate vârfurile care au acest grad
- n) afișează frecvența vârfurilor:
 - izolate: n1
 - terminale: n2
 - interioare: n3
- o) fiind dat șirul g_1, \dots, g_n , să se verifice dacă poate reprezenta șirul gradelor vârfurilor în această ordine;
- p) fiind dat șirul g_1, \dots, g_n , să se verifice dacă poate reprezenta șirul gradelor vârfurilor (nu neapărat în această ordine).