INFORMATICĂ · CLASA A XI-A · GRAFURI · LECȚIA 12

# Grafuri orientate: arce, grade, drumuri, circuite, tare conexitate

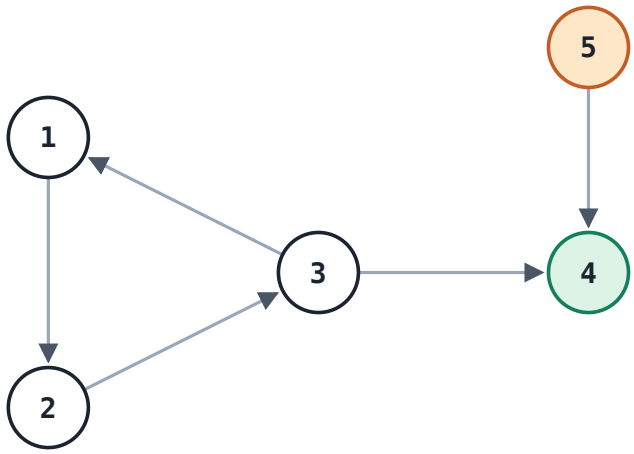
Durată: 60 min  Nivel: mate-info  Înainte: lecțiile xi-11

**CE VEI ȘTI DUPĂ LECȚIE**

- Cunoști terminologia: arc, extremitate inițială/finală, grad interior și exterior, drum, circuit.
- Reprezinți un graf orientat prin matrice de adiacență (nesimetrică) și liste de succesori.
- Aplici formulele pentru numărul de arce și suma gradelor.
- Recunoști componentele tare conexe și construiești matricea drumurilor (Roy-Warshall).

## 1. Când sensul contează

Un drum cu sens unic, o prietenie care nu e reciprocă (te urmăresc, dar nu-i urmărești), un curs care trebuie făcut înaintea altuia: legăturile au **direcție**. Le modelăm cu un **graf orientat**: G = (V, E) cu **arce** (x, y) care merg **doar** de la x la y. Arcul (x, y) este diferit de (y, x).




arcul (x, y) merge DOAR de la x la y · 5: nimic nu intră (sursă) · 4: nimic nu iese (destinație)

*Graf orientat*

Exemplu: n = 5, arcele (1,2), (2,3), (3,1), (3,4), (5,4).

| Termen | Definiție | În exemplu |
|---|---|---|
| **grad exterior** $d^+(x)$ | câte arce **ies** din x | $d^+(3) = 2$ |
| **grad interior** $d^-(x)$ | câte arce **intră** în x | $d^-(4) = 2$ |
| vârf **sursă** | $d^- = 0$ (nimic nu intră) | 5 |
| vârf **destinație** (sink) | $d^+ = 0$ (nimic nu iese) | 4 |

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| vârful | 1 | 2 | 3 | 4 | 5 |
| grad exterior $d^+$ | 1 | 1 | 2 | 0 | 1 |
| grad interior $d^-$ | 1 | 1 | 1 | 2 | 0 |

$d^+$ = câte arce IES din vârf · $d^-$ = câte arce INTRĂ în vârf

*Grade*

**LA BAC**

> - **$\Sigma\, d^+(x) = \Sigma\, d^-(x) = m$**: fiecare arc iese dintr-un vârf și intră în altul. - Numărul maxim de arce (fără bucle, cel mult un arc în fiecare sens între două vârfuri): **n(n − 1)**, dublul celui de la grafuri neorientate. - Numărul de grafuri orientate distincte cu n vârfuri etichetate (fără bucle): **2^(n(n−1))**.

## 2. Matricea de adiacență: nu mai e simetrică

`a[i][j] = 1` dacă există arcul (i, j). Citim arcele cu `a[x][y] = 1;` (**o singură** atribuire).

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | 0 | 1 | 0 | **0** | 0 |
| 2 | 0 | 0 | 1 | **0** | 0 |
| 3 | **1** | **0** | **0** | **1** | **0** |
| 4 | 0 | 0 | 0 | **0** | 0 |
| 5 | 0 | 0 | 0 | **1** | 0 |

a[i][j] = 1 dacă există arcul (i, j) · NU e simetrică · linia 3 = $d^+(3) = 2$ · coloana 4 = $d^-(4) = 2$

*Matricea*

- linia i: arcele care **ies** din i, deci **$d^+(i)$ = suma liniei i**;
- coloana j: arcele care **intră** în j, deci **$d^-(j)$ = suma coloanei j**;
- numărul de arce = numărul de 1 din matrice (nu se mai împarte la 2).

Listele de adiacență: `L[x].push_back(y);` **o singură dată** (doar vecinii în care se poate ajunge direct).

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

int a[101][101], n, m;

int main()
{
    cin >> n >> m;
    for (int k = 1; k <= m; k++)
    {
        int x, y;
        cin >> x >> y;
        a[x][y] = 1;                    // doar un sens!
    }
    for (int i = 1; i <= n; i++)        // gradele exterioare: sume pe linii
    {
        int s = 0;
        for (int j = 1; j <= n; j++) s += a[i][j];
        cout << s << ' ';
    }
    cout << '\n';
    for (int j = 1; j <= n; j++)        // gradele interioare: sume pe coloane
    {
        int s = 0;
        for (int i = 1; i <= n; i++) s += a[i][j];
        cout << s << ' ';
    }
    return 0;
}
```

**DE CE AȘA?**

**De ce linie pentru $d^+$ și coloană pentru $d^-$?** `a[i][j]` descrie arcul de la i la j. Toate arcele care **pleacă** din i sunt pe linia i; toate cele care **ajung** în j sunt pe coloana j. Dacă încurci cele două, răspunsul la „sursă” și „destinație” se inversează.

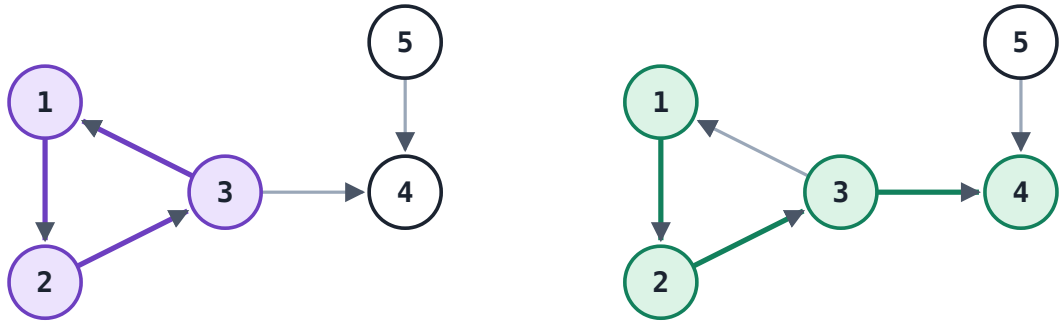
Trace pentru gradele vârfului 3 (arcele (3,1) și (3,4) ies, (2,3) intră):

| i | linia i | $d^+(i)$ | coloana i | $d^-(i)$ |
|---|---|---|---|---|
| 1 | 0 1 0 0 0 | 1 | 0 0 1 0 0 | 1 |
| 2 | 0 0 1 0 0 | 1 | 1 0 0 0 0 | 1 |
| 3 | 1 0 0 1 0 | 2 | 0 1 0 0 0 | 1 |
| 4 | 0 0 0 0 0 | 0 | 0 0 1 0 1 | 2 |
| 5 | 0 0 0 1 0 | 1 | 0 0 0 0 0 | 0 |

## 3. Drumuri și circuite

- **drum**: șir de vârfuri $x_1, x_2, \ldots, x_k$ cu arcele $(x_1,x_2)$, $(x_2,x_3)$, ... toate existente; lungimea = numărul de arce;

- **circuit**: drum care se întoarce unde a plecat ($x_1 = x_k$), fără să repete arce; **circuit elementar** = vârfuri distincte, în afară de primul = ultimul.




circuit: 1 → 2 → 3 → 1 drum: 1 → 2 → 3 → 4 (nu se poate înapoi!)

*Drum și circuit*

> **ATENȚIE — GREȘELI FRECVENTE**
>
> Spre deosebire de lanț (neorientat), la **drum** nu poți merge pe un arc „înapoi”: din 4 nu se poate ajunge la 3, deși se poate de la 3 la 4.

### Ce vârfuri pot fi atinse dintr-un vârf?

Același DFS ca la grafuri neorientate, dar cu o singură atribuire la citire (și deci urmăm arcele doar în sensul lor):

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

int a[101][101], viz[101], n, m;

void dfs(int x)
{
    viz[x] = 1;
    cout << x << ' ';
    for (int y = 1; y <= n; y++)
        if (a[x][y] && !viz[y]) dfs(y);
}

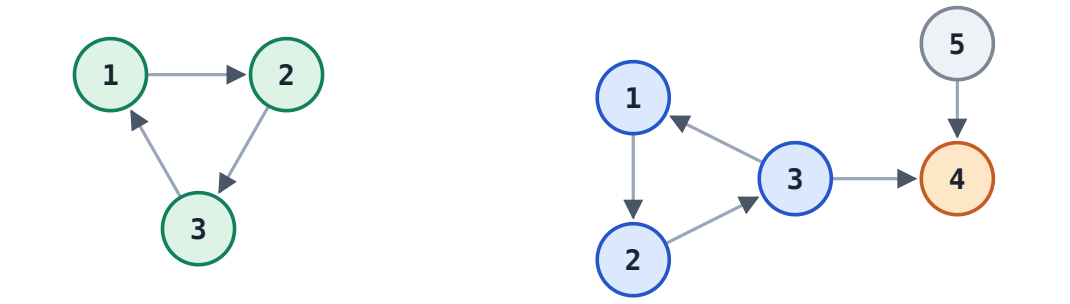
int main()
{
    cin >> n >> m;
    for (int k = 1; k <= m; k++)
    {
        int x, y;
        cin >> x >> y;
        a[x][y] = 1;
    }
    int start;
    cin >> start;
    dfs(start);
    return 0;
}
```

**DE CE AȘA?**

Singura diferență față de graful neorientat este `a[x][y] = 1` fără `a[y][x]`. Din 1 putem ajunge la 2, 3 și 4; din 5 doar la 4 (iar 4 nu mai are arce care ies).

## 4. Conexitate tare

Un graf orientat este **tare conex** dacă pentru **orice** două vârfuri x, y există drum de la x la y **și** de la y la x. O **componentă tare conexă** este o mulțime maximală de vârfuri cu această proprietate.




*Tare conex*

**LA BAC**

> - Un graf tare conex cu n vârfuri are cel puțin **n arce** (un circuit care trece prin toate vârfurile este varianta minimă). - Un circuit este întotdeauna în interiorul unei componente tare conexe. - Graful condensat (fiecare componentă tare conexă devine un singur vârf) **nu are circuite**.

**Test practic:** graful este tare conex ⇔ din vârful 1 se poate ajunge la toate (în G) **și** din toate se poate ajunge la 1, adică din 1 se atinge tot în graful cu arcele **inversate** (matricea transpusă).

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

int a[101][101], t[101][101], v1[101], v2[101], n, m;

void dfs(int x, int g[][101], int viz[])
{
    viz[x] = 1;
    for (int y = 1; y <= n; y++)
        if (g[x][y] && !viz[y]) dfs(y, g, viz);
}

int main()
{
    cin >> n >> m;
    for (int k = 1; k <= m; k++)
    {
        int x, y;
        cin >> x >> y;
        a[x][y] = 1;
        t[y][x] = 1;                        // arcul inversat
    }
    dfs(1, a, v1);                          // ce atinge 1
    dfs(1, t, v2);                          // cine ajunge în 1
    bool tare = true;
    for (int i = 1; i <= n; i++)
        if (!v1[i] || !v2[i]) tare = false;
    cout << (tare ? "DA" : "NU");
    return 0;
}
```

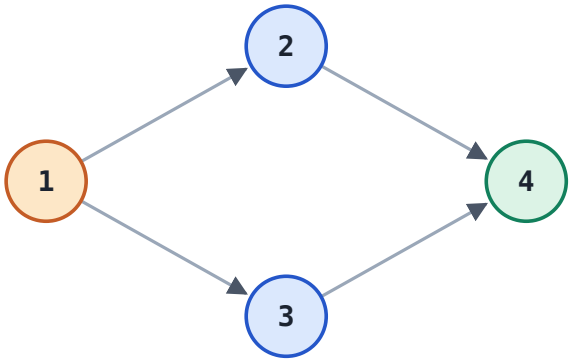
**DE CE AȘA?**

**De ce e suficient să verificăm doar vârful 1?** Dacă din 1 ajungem la oricine și oricine ajunge la 1, atunci pentru orice x, y: x → 1 → y este un drum de la x la y. **De ce transpusa?** Un drum de la i la 1 în G este un drum de la 1 la i în graful cu arcele inversate.

## 5. Grafuri speciale orientate

| Graf | Definiție | Număr de arce |
|---|---|---|
| **complet** | pentru orice x ≠ y ambele arce (x,y) și (y,x) | n(n − 1) |

| Graf | Definiție | Număr de arce |
|---|---|---|
| **turneu** | pentru orice x ≠ y **exact unul** dintre (x,y) și (y,x) | n(n − 1)/2 |
| **fără circuite (DAG)** | nu există circuite | cel mult n(n − 1)/2 |




orice arc merge de la un vârf mai din față la unul mai din spate în ordinea aleasă

*DAG*

Un graf fără circuite are o **ordine topologică**: vârfurile pot fi scrise în șir astfel încât **toate arcele merg spre dreapta**. Exemplu: sarcini cu dependențe (să te îmbraci înainte de a ieși).

## 6. Greșeli frecvente

| Greșeala | Efect | Corect |
|---|---|---|
| `a[x][y] = a[y][x] = 1` la graf orientat | transformi arcele în muchii | doar `a[x][y] = 1` |
| m = suma matricei / 2 | jumătate din arce | m = suma matricei |
| $d^+$ pe coloană, $d^-$ pe linie | surse și destinații inversate | linie = ies, coloană = intră |
| „lanț" pentru drum | ignori sensul arcelor | drum = respecți sensul |
| tare conex = „conex dacă ignor sensul" | graful (1→2) are legătură, dar nu e tare conex | drumuri în **ambele** sensuri |

## 7. La BAC

**LA BAC**

Întrebări tipice: „câte arce are graful dat prin matrice" (numeri 1-urile), „care vârf are gradul exterior / interior maxim", „câte circuite elementare", „câte componente tare conexe", „număr minim de arce de adăugat pentru a deveni tare conex", „număr maxim de arce într-un graf fără circuite" (n(n−1)/2: arcele merg doar „înainte" într-o ordine topologică). Începe prin a nota $d^+$ și $d^-$ pentru fiecare vârf.

## 8. Exersează pe hârtie

**1.** Graful orientat cu n = 4 și arcele (1,2), (1,3), (2,3), (3,4), (4,2). Scrie $d^+$ și $d^-$ pentru fiecare vârf.

**2.** Câte arce poate avea cel mult un graf orientat cu 8 vârfuri? Dar un graf orientat fără circuite?

**3.** Scrie matricea de adiacență a grafului de la exercițiul 1.

**4.** Are graful de la exercițiul 1 circuite? Care? Este tare conex?

**5.** Un graf orientat cu 6 vârfuri are 3 componente tare conexe, fiecare cu 2 vârfuri. Care este numărul minim de arce?

**6.** (BAC) Câte grafuri turneu (fără circuite) cu n = 4 vârfuri etichetate 1..4 există, dacă se cere ca vârful 1 să aibă gradul exterior 3?

**7.** Un graf orientat are 7 vârfuri și toate au gradul interior 2. Câte arce are?

**8.** Pentru DAG-ul din figura secțiunii 5 (arcele 1→2, 1→3, 2→4, 3→4), scrie toate ordinile topologice posibile.

**9.** Care este numărul minim de arce într-un graf orientat tare conex cu 100 de vârfuri?

**10.** Scrie un program care afișează sursele ($d^- = 0$) și destinațiile ($d^+ = 0$) ale unui graf citit prin arce.

## 9. Mai departe (pentru cine vrea mai mult)

**Sortare topologică (algoritmul lui Kahn).** Cât timp există un vârf cu gradul interior 0: îl scoți, îl adaugi în rezultat și scazi gradul interior al vecinilor lui. Dacă toate vârfurile au fost scoase, graful nu avea circuite.

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

int a[101][101], gint[101], coada[101], n, m;

int main()
{
    cin >> n >> m;
    for (int k = 1; k <= m; k++)
    {
        int x, y;
        cin >> x >> y;
        a[x][y] = 1;
        gint[y]++;                              // gradul interior
    }
    int prim = 0, ultim = 0;
    for (int i = 1; i <= n; i++)
        if (gint[i] == 0) coada[ultim++] = i;
    while (prim < ultim)
    {
        int x = coada[prim++];
        cout << x << ' ';
        for (int y = 1; y <= n; y++)
            if (a[x][y] && --gint[y] == 0)      // y nu mai așteaptă pe nimeni
                coada[ultim++] = y;
    }
    return 0;
}
```

Dacă `ultim < n` la final, graful are circuit (vârfurile din circuit nu ajung niciodată cu gradul interior 0).

**Matricea drumurilor (Roy-Warshall).** `d[i][j] = 1` dacă există drum de la i la j; se calculează cu trei `for`-uri, ca la Floyd (lecția XI.19).

**Rețele sociale.** „Cine urmărește pe cine" este un graf orientat; „prieteni" (reciproc) înseamnă că ambele arce există. PageRank, algoritmul de la început al Google, lucrează pe un astfel de graf cu miliarde de vârfuri.

## Rezumat

- Arc (x, y): doar de la x la y. $d^+$ = ies (suma liniei), $d^-$ = intră (suma coloanei); $\Sigma d^+ = \Sigma d^- = m$.
- Matrice nesimetrică, citire cu `a[x][y] = 1`; arce = numărul de 1; maxim $n(n-1)$ arce.
- Drum / circuit respectă sensul; tare conex = drum în ambele sensuri între oricare două vârfuri ($\geq n$ arce).
- Fără circuite ⇒ ordine topologică, cel mult $n(n-1)/2$ arce.

## Probleme de lucru (pbinfo)

Începe cu acestea, în ordine: sunt alese să urmeze pașii din lecție.

- [ ] #573 **grade2** — ușoară
  https://www.pbinfo.ro/probleme/573/grade2
- [ ] #586 **gradintnul** — ușoară
  https://www.pbinfo.ro/probleme/586/gradintnul
- [ ] #580 **roy-warshall** — ușoară
  https://www.pbinfo.ro/probleme/580/roy-warshall

## Temă (pbinfo)

Trimite soluțiile pe pbinfo și notează punctajul obținut. Bifează ce ai terminat.

- [ ] #575 **cunoscuti** — ușoară
  https://www.pbinfo.ro/probleme/575/cunoscuti
- [ ] #4153 **grade3** — ușoară
  https://www.pbinfo.ro/probleme/4153/grade3
- [ ] #3423 **ctcmax** — ușoară
  https://www.pbinfo.ro/probleme/3423/ctcmax
- [ ] #576 **afiscicluri** — ușoară
  https://www.pbinfo.ro/probleme/576/afiscicluri
- [ ] #583 **tare-conexitate** — medie
  https://www.pbinfo.ro/probleme/583/tare-conexitate
- [ ] #581 **drumuri** — medie
  https://www.pbinfo.ro/probleme/581/drumuri

## Pentru cei care vor mai mult

Probleme mai grele sau de concurs, pentru cine vrea să meargă dincolo de BAC.

- [ ] #584 **anunt** — medie
  https://www.pbinfo.ro/probleme/584/anunt

## Grile din subiectele de BAC (pbinfo)

Exerciții de tip grilă din variantele de Bacalaureat, rezolvate online pe pbinfo, cu verificare automată a răspunsului.

- [ ] #133 **Exercițiul #133**
  proprietăți ale unui graf orientat cu n vârfuri
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=133

- [ ] #142 **Exercițiul #142**
  arce de adăugat pentru a obține o proprietate
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=142
- [ ] #213 **Exercițiul #213**
  numărul de circuite pentru un graf dat prin matrice
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=213
- [ ] #515 **Exercițiul #515**
  graf fără circuite: număr maxim de arce
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=515
- [ ] #644 **Exercițiul #644**
  graf fără circuite cu 8 vârfuri: arce maxime
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=644
- [ ] #656 **Exercițiul #656**
  componente tare conexe și arce
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=656
- [ ] #1066 **Exercițiul #1066**
  graf orientat cu 5 vârfuri dat prin desen: grade, drumuri
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=1066
- [ ] #1122 **Exercițiul #1122**
  arce minime pentru un graf tare conex
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=1122
- [ ] #1152 **Exercițiul #1152**
  graf cu 10 vârfuri și 12 arce: valoarea maximă a unui grad
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=1152
- [ ] #1795 **Exercițiul #1795**
  graf cu 4 vârfuri dat prin matrice de adiacență
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=1795

## Citește și exersează în plus

- Teorie grafuri orientate https://www.pbinfo.ro/articole/509/grafuri-orientate
- Graf tare conex. Componente tare conexe https://www.pbinfo.ro/articole/6036/tare-conexitate
- Matricea drumurilor. Algoritmul Roy-Warshall https://www.pbinfo.ro/articole/5888/matricea-drumurilor
- Graf orientat — definiții — terminologia completă, cu exemple (arc, drum, circuit, conexitate tare)
  https://ro.wikipedia.org/wiki/Graf_orientat
- VisuAlgo — Topological Sort — DFS/BFS pe grafuri orientate și ordinea topologică (engleză)
  https://visualgo.net/en/dfsbfs
- CSES — Course Schedule — sortare topologică: ordinea în care parcurgi cursurile (engleză)
  https://cses.fi/problemset/task/1679

# Răspunsuri

## Exersează pe hârtie

**1.** $d^+$: 1 → 2, 2 → 1, 3 → 1, 4 → 1. $d^-$: 1 → 0, 2 → 2 (de la 1 și 4), 3 → 2 (de la 1 și 2), 4 → 1. Suma $d^+ = 5$ = suma $d^- = m$.

**2.** Maxim $8 \cdot 7 =$ **56** arce. Fără circuite: cel mult $8 \cdot 7 / 2 =$ **28**.

**3.**

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

**4.** Da: **2 → 3 → 4 → 2** (circuit de lungime 3). Nu este tare conex: din 2 nu se poate ajunge la 1 (niciun arc nu intră în 1).

**5.** Fiecare componentă tare conexă cu 2 vârfuri are nevoie de cel puțin 2 arce (x → y și y → x): $3 \cdot 2 =$ **6** arce. (Între componente nu se cere nimic.)

**6.** Turneu fără circuite = ordinea topologică totală (o permutare a vârfurilor); vârful 1 cu $d^+ = 3$ înseamnă că 1 este primul, iar restul pot fi ordonate în 3! = **6** moduri.

**7.** Suma gradelor interioare = $7 \cdot 2 =$ **14** arce.

**8.** 1 este mereu primul și 4 mereu ultimul; 2 și 3 se pot alege în orice ordine: **1 2 3 4** și **1 3 2 4**.

**9. 100** arce (un circuit care trece prin toate vârfurile: 1 → 2 → … → 100 → 1).

**10.** Calculezi `gint[]` și `gext[]` la citire (la `x y`: `gext[x]++`, `gint[y]++`), apoi afișezi vârfurile cu `gint[i] == 0` (surse) și cele cu `gext[i] == 0` (destinații).