INFORMATICĂ · CLASA A XI-A · GRAFURI · LECȚIA 10

Grafuri neorientate (2): conexitate, graf complet, bipartit, hamiltonian, eulerian

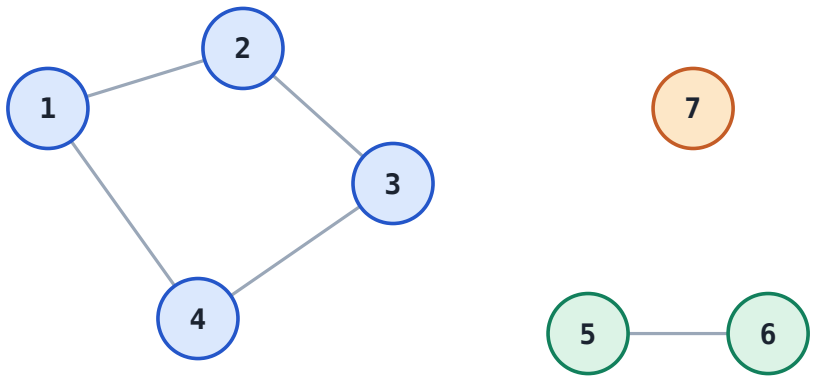
Durată: 60 min Nivel: mate-info Înainte: lecțiile xi-09

CE VEI ȘTI DUPĂ LECȚIE

- Recunoști un graf conex și determini componentele conexe.
- Cunoști graful complet, graful bipartit (și bipartit complet), graful regulat.
- Verifici condițiile pentru graf eulerian și știi ce înseamnă hamiltonian.
- Rezolvi grile numerice: număr minim de muchii pentru conexitate, număr de componente etc.

1. Conexitate: „se poate ajunge de oriunde oriunde?"

- un graf este **conex** dacă între **oricare două** vârfuri există un lanț;
- o **componentă conexă** este „insula" maximală: toate vârfurile la care poți ajunge dintr-un vârf dat;
- un vârf izolat formează singur o componentă.



{1,2,3,4} · {5,6} · {7} (izolat) → minim n − p = 4 muchii, p − 1 = 2 muchii pentru a-l face conex

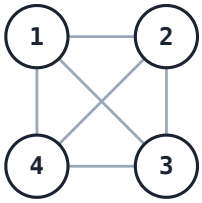
Componente conexe

LA BAC

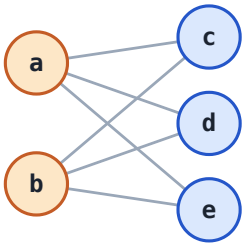
Formule pentru grile (n vârfuri, m muchii, p componente conexe): - conex ⇒ **m ≥ n − 1**; în general **m ≥ n − p**; - ca să faci conex un graf cu p componente trebuie adăugate cel puțin **p − 1** muchii (câte una între insulele vecine); - **neconex** cu număr maxim de muchii: un vârf izolat + graf complet pe celelalte n − 1: **(n − 1)(n − 2)/2**; - cu m muchii, numărul **maxim** de componente este n − m (fiecare muchie „lipește" două componente).

De ce m ≥ n − 1 pentru un graf conex: pornești de la n vârfuri izolate (n componente). Fiecare muchie scade numărul de componente cu cel mult 1, iar tu vrei să ajungi la 1: ai nevoie de cel puțin n − 1 muchii.

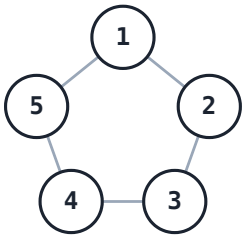
2. Grafuri speciale



K_4 complet: 6 muchii



$K_{2,3}$ bipartit complet: $2 \cdot 3 = 6$ muchii



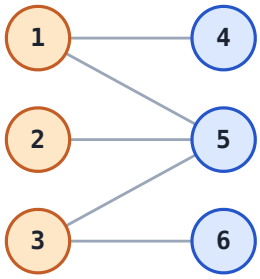
ciclu C_5: regulat (grad 2), nu e bipartit

Grafuri speciale

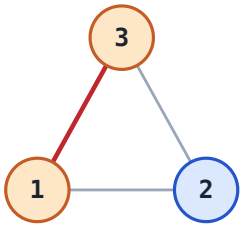
Graf	Definiție	Număr de muchii
complet K_n	oricare două vârfuri sunt adiacente	$n(n-1)/2$
regulat de grad r	toate vârfurile au gradul r	$n \cdot r / 2$
bipartit	V se împarte în A și B; muchiile leagă doar A cu B	—
bipartit complet $K_{p,q}$	toate muchiile posibile între A (p vârfuri) și B (q vârfuri)	$p \cdot q$

Bipartit: colorarea cu două culori

Pornești cu un vârf colorat portocaliu; vecinii lui trebuie să fie albaștri; vecinii lor portocalii, și tot așa. Dacă ajungi la două vârfuri vecine de aceeași culoare, graful **nu** e bipartit.



bipartit: A = {1,2,3} · B = {4,5,6}



triunghi: apare muchia 1-3 între două vârfuri portocalii → nu e bipartit

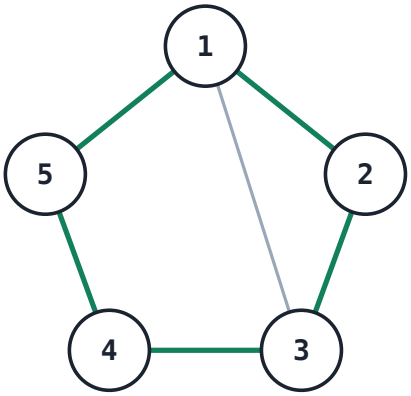
Bipartit

SFAT

Un graf este bipartit dacă și numai dacă **nu are cicluri de lungime impară**. Orice triunghi (ciclu de 3) îl face nebipartit; un ciclu de 4 sau de 6 nu.

3. Graf hamiltonian: toate vârfurile

- **ciclu hamiltonian**: ciclu elementar care trece **prin toate vârfurile**, o singură dată fiecare;
- graf **hamiltonian** = are un ciclu hamiltonian.



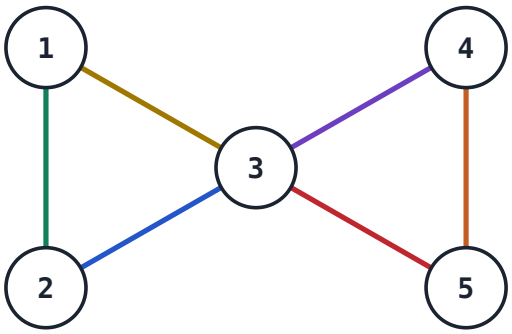
1-2-3-4-5-1; muchia 1-3 nu e folosită (nu e obligatoriu să treci prin toate muchiile)

Hamilton

Nu există o condiție simplă „dacă și numai dacă". O condiție **suficientă** (Dirac): dacă n ≥ 3 și fiecare vârf are gradul ≥ n/2, graful este hamiltonian. Graful complet K_n (n ≥ 3) este hamiltonian, iar numărul ciclurilor hamiltoniene distincte (fără să contăm punctul de start și sensul) este **(n − 1)! / 2**.

4. Graf eulerian: toate muchiile

- **ciclu eulerian**: ciclu (nu neapărat elementar) care trece **prin toate muchiile**, o singură dată fiecare;
- un graf (fără vârfuri izolate) este **eulerian** ⇔ este **conex** și toate vârfurile au **grad par**;
- are doar un **lanț** eulerian (nu și ciclu) ⇔ este conex și are exact **2** vârfuri de grad impar (lanțul începe într-unul și se termină în celălalt).



traseu: 1 → 2 → 3 → 4 → 5 → 3 → 1 (fiecare muchie o singură dată)

Euler

De ce gradele pare: de fiecare dată când „intri” într-un vârf, trebuie să și „ieși” pe o altă muchie, deci muchiile fiecărui vârf se consumă în perechi. Dacă un vârf ar avea grad impar, la un moment dat ai rămâne blocat în el.

> **DE ȘTIUT**
>
> Memorare: **H**amilton = **v**ârfuri (hotelurile pe care le vizitezi); **E**uler = **m**uchii (străzile pe care le parcurgi, ca poștașul). Euler a rezolvat în 1736 problema podurilor din Königsberg: toate cele 4 „insule” aveau grad impar, deci traseul dorit nu exista.

5. În program: componentele conexe și testele de mai sus

Cea mai simplă metodă: pornim o parcurgere din primul vârf nevizitat, o numim „insulă nouă” și marcăm tot ce găsim (în lecția următoare discutăm parcurgerile în detaliu). Cu matricea de adiacență:

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

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

void dfs(int x)
{
    viz[x] = 1;                              // marcăm vârful
    for (int y = 1; y <= n; y++)
        if (a[x][y] == 1 && !viz[y])         // vecin încă nevăzut
            dfs(y);
}

int main()
{
    cin >> n >> m;
    for (int k = 1; k <= m; k++)
    {
        int x, y;
        cin >> x >> y;
        a[x][y] = a[y][x] = 1;
    }
    int comp = 0;
    for (int i = 1; i <= n; i++)
        if (!viz[i])
        {
            comp++;                          // vârf nou: insulă nouă
            dfs(i);                          // o marcăm toată
        }
    cout << comp;
    return 0;
}
```

DE CE AȘA?

De ce numărul de apeluri din `main` este numărul de componente? `dfs(i)` marchează toate vârfurile insulei lui i. Deci un vârf pe care `main` îl găsește nevăzut aparține cu siguranță unei insule neatinse până acum. Graful este conex ⇔ `comp == 1`.

Test: pentru `5 4` cu muchiile [1,2], [1,3], [2,3], [3,4] se găsesc 2 componente ({1,2,3,4} și {5}); pentru `4 0` (fără muchii), 4 componente.

Eulerian sau nu?

Combinăm cele două reguli: număr de componente (doar cele cu muchii) și gradele impare.

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

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

void dfs(int x)
{
    viz[x] = 1;
    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] = a[y][x] = 1;
        grad[x]++;
        grad[y]++;
    }
    dfs(1);
    bool conex = true;
    for (int i = 1; i <= n; i++)
        if (!viz[i]) conex = false;

    int impare = 0;
    for (int i = 1; i <= n; i++)
        if (grad[i] % 2 == 1) impare++;

    if (conex && impare == 0) cout << "ciclu eulerian";
    else if (conex && impare == 2) cout << "lant eulerian";
    else cout << "nu";
    return 0;
}
```

DE CE AȘA?

De ce verificăm și conexitatea? Două triunghiuri separate au toate gradele pare, dar nu poți desena ambele fără să ridici creionul. Condiția completă este „conex ȘI grade pare”.

Bipartit?

Colorăm vârfurile cu 1 și 2; vecinii primesc culoarea opusă; un conflict înseamnă „nu”.

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

int a[101][101], col[101], n, m;
bool ok = true;

void coloreaza(int x, int c)
{
    col[x] = c;
    for (int y = 1; y <= n; y++)
        if (a[x][y])
        {
            if (col[y] == 0) coloreaza(y, 3 - c);     // 1 <-> 2
            else if (col[y] == c) ok = false;          // vecini de aceeași culoare
        }
}

int main()
{
    cin >> n >> m;
    for (int k = 1; k <= m; k++)
    {
        int x, y;
        cin >> x >> y;
        a[x][y] = a[y][x] = 1;
    }
    for (int i = 1; i <= n; i++)
        if (col[i] == 0) coloreaza(i, 1);
    cout << (ok ? "DA" : "NU");
    return 0;
}
```

DE CE AȘA?

De ce `3 - c`? Dacă c este 1, rezultatul este 2; dacă c este 2, rezultatul este 1: comută între cele două culori fără `if`. **De ce `for` în `main`?** Graful poate avea mai multe componente; fiecare se colorează independent.

Intrare / ieșire

Intrare	Ieșire

`5 4`
`1 2`
`1 3`
`2 3`
`3 4`
	`2` (componente conexe)

Intrare	Ieșire

`3 3`
`1 2`
`2 3`
`1 3`
	`ciclu eulerian` (triunghi: gradele 2, 2, 2)

`3 2`
`1 2`
`2 3`
	`lant eulerian` (gradele 1, 2, 1)

`4 3`
`1 2`
`1 3`
`1 4`
	`nu` (gradele 3, 1, 1, 1: patru vârfuri impare)

6. Greșeli frecvente

Greșeala	Efect	Corect
eulerian = „toate gradele pare”	uiți graful neconex	conex **și** grade pare
hamiltonian ⇔ eulerian	sunt probleme diferite	vârfuri (H) / muchii (E)
(n−1)(n−2)/2 pentru oricâte componente	formula e pentru 2 componente (una izolată)	recalculezi pe cazul dat
bipartit ⇔ fără triunghiuri	un ciclu de 5 îl strică și el	fără orice ciclu **impar**
p − 1 muchii ⇒ conex mereu	doar dacă le pui între componente diferite	fiecare muchie leagă două insule

7. La BAC

> **LA BAC**
>
> Întrebări tipice: „numărul **minim** de muchii pentru n noduri conex” (n − 1), „numărul **maxim** de muchii dacă graful nu e conex” ((n − 1)(n − 2)/2), „câte muchii trebuie adăugate/eliminate ca să devină conex/arbore”, „care e numărul maxim de componente”, „este eulerian/hamiltonian”. Desenează un caz mic (n = 4 sau 5) și numără: formulele vin din asta.

8. Exersează pe hârtie

1. Un graf are 10 vârfuri și 3 componente conexe. Care este numărul minim de muchii pe care le poate avea?

2. Care este numărul maxim de muchii ale unui graf neconex cu 8 vârfuri?

3. Un graf are 20 de vârfuri și 10 muchii. Câte componente conexe poate avea cel mult?

4. Câte muchii trebuie eliminate din graful complet K_6 pentru a obține un graf parțial conex fără cicluri (arbore)?

5. Gradele unui graf conex sunt 2, 2, 4, 2, 2, 2. Este eulerian? Justifică.

6. Gradele unui graf conex sunt 3, 3, 2, 2, 2. Are lanț eulerian? Dar ciclu eulerian?

7. Un graf bipartit complet $K_{4,5}$ are câte muchii? Câte vârfuri?

8. Care dintre grafurile K_n cu $n \geq 3$ sunt euleriene? (Gradele sunt $n - 1$.)

9. (BAC) Un graf cu 12 noduri are 3 componente conexe, fiecare fiind arbore. Câte muchii are?

10. Dă un exemplu de graf hamiltonian care nu este eulerian și unul eulerian care nu este hamiltonian.

9. Mai departe (pentru cine vrea mai mult)

Drumuri hamiltoniene: dificultate. Nu se cunoaște un algoritm rapid (polinomial) pentru „există un ciclu hamiltonian?"; backtracking-ul ia timp exponențial. Pentru ciclul eulerian, în schimb, testul de grade pare și conexitate este instantaneu.

Algoritmul lui Hierholzer. Construiește efectiv ciclul eulerian: parcurgi muchii nevizitate cu DFS și adaugi vârful în rezultat la ieșirea din recursivitate; inversezi la final. Se rezolvă în $O(n + m)$.

Rețele de zbor. Problema „toate orașele, o singură dată" (comisul voiajor) este varianta cu costuri a ciclului hamiltonian și e una dintre cele mai studiate probleme din informatică.

Numărul de cicluri hamiltoniene în K_n. $(n - 1)!/2$: pentru $n = 5$, 12; pentru $n = 10$, 181 440.

Rezumat

- Conex: lanț între oricare două vârfuri; $m \geq n - 1$; p componente cer $p - 1$ muchii de adăugat.
- Neconex maxim: $(n - 1)(n - 2)/2$ muchii. Complet: $n(n - 1)/2$. Bipartit complet $K_{p,q}$: $p \cdot q$.
- Bipartit ⇔ fără cicluri impare (colorare cu 2 culori).
- Eulerian ⇔ conex + toate gradele pare; lanț eulerian ⇔ conex + exact 2 grade impare. Hamiltonian = ciclu prin toate vârfurile.

Probleme de lucru (pbinfo)

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

- [ ] #431 **graf-complet** — medie
  https://www.pbinfo.ro/probleme/431/graf-complet
- [ ] #437 **conex** — ușoară
  https://www.pbinfo.ro/probleme/437/conex
- [ ] #438 **componenteconexe** — ușoară
  https://www.pbinfo.ro/probleme/438/componenteconexe

Temă (pbinfo)

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

- [ ] #4069 **grafcomplet** — ușoară
  https://www.pbinfo.ro/probleme/4069/grafcomplet
- [ ] #471 **bipartit** — ușoară
  https://www.pbinfo.ro/probleme/471/bipartit
- [ ] #473 **bipartitcomplet** — ușoară
  https://www.pbinfo.ro/probleme/473/bipartitcomplet
- [ ] #475 **lant** — ușoară
  https://www.pbinfo.ro/probleme/475/lant
- [ ] #478 **ciclu** — ușoară
  https://www.pbinfo.ro/probleme/478/ciclu
- [ ] #441 **componenteconexe1** — medie
  https://www.pbinfo.ro/probleme/441/componenteconexe1
- [ ] #545 **euler** — medie
  https://www.pbinfo.ro/probleme/545/euler

Pentru cei care vor mai mult

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

- [ ] #548 **hamilton** — medie
  https://www.pbinfo.ro/probleme/548/hamilton

Grile din subiectele de BAC (pbinfo)

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

- [ ] #73 **Exercițiul #73**
  număr maxim de componente conexe pentru n noduri și m muchii
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=73
- [ ] #79 **Exercițiul #79**
  număr maxim de vârfuri izolate
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=79
- [ ] #218 **Exercițiul #218**
  muchii maxime cu un număr dat de componente
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=218
- [ ] #261 **Exercițiul #261**
  muchii maxime într-un graf neconex
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=261

- [ ] #288 **Exercițiul #288**
  muchii de adăugat pentru ca graful să devină conex
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=288

- [ ] #322 **Exercițiul #322**
  graf conex: număr minim de noduri cu grad 1
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=322

- [ ] #544 **Exercițiul #544**
  graf bipartit complet: număr de muchii
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=544

- [ ] #1049 **Exercițiul #1049**
  graf eulerian: muchii minime în funcție de grade
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=1049

- [ ] #1129 **Exercițiul #1129**
  număr maxim / minim de componente conexe
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=1129

- [ ] #1131 **Exercițiul #1131**
  conex: n și numărul de valori nenule din matrice
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=1131

Citește și exersează în plus

- Teorie grafuri neorientate https://www.pbinfo.ro/articole/810/grafuri-neorientate
- Graf hamiltonian https://www.pbinfo.ro/articole/5790/graf-hamiltonian
- Graf eulerian https://www.pbinfo.ro/articole/5792/graf-eulerian
- Cele șapte poduri din Königsberg — povestea din care s-a născut teoria grafurilor (Euler, 1736)
  https://ro.wikipedia.org/wiki/Problema_podurilor_din_Königsberg
- VisuAlgo — Graph Traversal — vezi cum se descoperă componentele conexe (engleză)
  https://visualgo.net/en/dfsbfs
- Bipartite graphs — explicație — caracterizarea prin cicluri impare, cu exemple (engleză)
  https://en.wikipedia.org/wiki/Bipartite_graph

Răspunsuri

Exersează pe hârtie

1. Cel puțin n − p = 10 − 3 = **7** muchii (de exemplu trei arbori).

2. Un vârf izolat + K_7: 7·6/2 = **21** muchii.

3. n − m = 20 − 10 = **10** componente (fiecare muchie unește două vârfuri izolate; 10 perechi acoperă toate cele 20 de vârfuri).

4. K_6 are 15 muchii, arborele are 5: se elimină 15 − 5 = **10**.

5. Gradele sunt toate pare (2, 2, 4, 2, 2, 2) și graful e conex, deci **da, este eulerian**.

6. Exact două grade impare (3 și 3): **are lanț eulerian**, dar **nu are ciclu eulerian**.

7. 4 · 5 = **20** muchii și 4 + 5 = **9** vârfuri.

8. Gradul n − 1 trebuie să fie par, deci n impar: **n = 3, 5, 7, ...**

9. Un arbore cu k vârfuri are k − 1 muchii. Suma: 12 − 3 = **9** muchii.

10. Hamiltonian, nu eulerian: K_4 (grade 3, impare). Eulerian, nu hamiltonian: două triunghiuri lipite într-un vârf (fundă): gradele 2, 2, 4, 2, 2 (toate pare), dar vârful comun 3 ar trebui vizitat de două ori într-un ciclu prin toate vârfurile.