

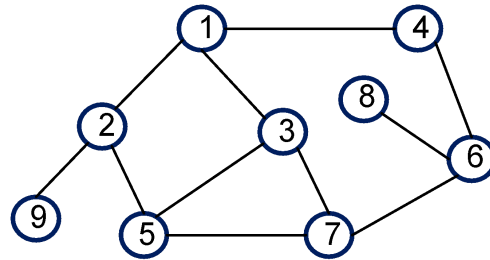
## **MEMORAREA GRAFURILOR. PARCURI. APLICAȚII**

<b>A. Memorarea unui graf</b>	<b>2</b>
<b>B. Parcurgerea în lățime BF</b>	<b>2</b>
<b>C. Parcurgerea în adâncime DF</b>	<b>2</b>
<b>D. Probleme</b>	<b>3</b>

Datele de intrare se vor citi din fișierul *graf.in*.

**Dacă nu se precizează în enunț, un graf este dat prin următoarele informații: numărul de vârfuri  $n$ , numărul de muchii  $m$  și lista muchiilor (o muchie fiind dată prin extremitățile sale).**

graf.in	
9	11
1	2
1	3
1	4
2	5
2	9
3	5
3	7
5	7
6	7
6	8
4	6



## A. Memorarea unui graf

1. Scrieți un subprogram pentru construirea în memorie a matricei de adiacență a unui graf (neorientat/orientat în funcție de un parametru trimis subprogramului) citit din fișierul *graf.in* cu structura precizată mai sus și un subprogram pentru afișarea matricei de adiacență
2. Scrieți un subprogram pentru construirea în memorie a listelor de adiacență pentru un graf (neorientat/orientat în funcție de un parametru trimis subprogramului) citit din fișierul *graf.in* cu structura precizată mai sus și un subprogram pentru afișarea listelor de adiacență
3. Implementați algoritmi de trecere de la o modalitate de reprezentare la alta.
4. **Exercițiu** Propuneți modalități de reprezentare și pentru grafuri orientate și pentru multigrafuri neorientate/orientate (care admit muchii/arce multiple și bucle)

## B. Parcurgerea în lățime BF

1. a) <https://infoarena.ro/problema/bfs>  
b) Se citește în plus (față de a)) de la tastatură două vârfuri  $s$  și  $x$ . Să se afișeze un drum minim (cu număr minim de arce) de la  $s$  la  $x$

## C. Parcurgerea în adâncime DF

1. <https://infoarena.ro/problema/dfs> + afișarea arcelor de întoarcere, traversare, avansare (pe categorii)
2. Dat un graf neorientat (nu neapărat conex), să se verifice dacă graful conține un ciclu elementar (nu este aciclic). În caz afirmativ **să se afișeze un astfel de ciclu.**

graf.in	graf.out
7 8 1 3 2 4 3 4 3 5 3 6	3 5 6 3 (nu neaparat in aceasta ordine; solutia nu este unica, un alt ciclu este de exemplu 3 6 7 3)

5 6	
6 7	
3 7	

## D. Probleme

**Implementați algoritmi eficienți pentru rezolvarea următoarelor probleme. Se vor adăuga comentarii în cod cu ideea algoritmului și complexitatea sa.**

1. **Graf bipartit.** <https://cses.fi/problemset/task/1668>

2. a) **Sortare topologică.** <https://cses.fi/problemset/task/1679>

b) Modificați programul de la a) astfel încât în cazul în care nu pot fi urmate toate cursurile (deci se afișează IMPOSSIBLE), să afișeze și o listă de cursuri  $[c_1, c_2, \dots, c_k, c_1]$  care depind circular unele de altele (orice curs  $c_i$  trebuie urmat înaintea cursului  $c_{i-1}$ , iar  $c_k$  înaintea cursului  $c_1$ ).

3. **Fill** <https://cses.fi/problemset/task/1192>

4. **Componente tare conexe.** <https://cses.fi/problemset/task/1683>

5. **Puncte critice (de articulație).** <https://www.spoj.com/problems/SUBMERGE/>

6. **Muchii critice.** [https://www.spoj.com/problems/EC\\_P/](https://www.spoj.com/problems/EC_P/)

7. **Distanțe și drumuri minime.** Se dă o rețea neorientată cu  $n$  noduri și o listă de noduri reprezentând puncte de control pentru rețea în fișierul **graf.in**, în formatul precizat la începutul laboratorului; în plus, pe ultima linie din fișier se află punctele de control separate prin spațiu. Să se determine pentru fiecare nod din rețea distanța până la cel mai apropiat punct de control de acesta. În fișierul **graf.out** se vor afișa pentru fiecare nod de la 1 la  $n$  aceste distanțe separate prin spațiu.

<b>graf.in</b> - pe ultima linie se dau punctele de control	<b>graf.out</b>
9 11 1 2 1 3 1 4 2 5 2 9 3 5 3 7 5 7 6 7 6 8 4 6 <b>8 9</b>	2 1 3 2 2 1 2 0 0  (Explicații: Punctele de control sunt 8 și 9 - cel mai apropiat punct de control de nodul 1 este 9, aflat la distanță 2 - cel mai apropiat punct de control de nodul 2 este 9, aflat la distanță 1 etc)

8. **BFS 0-1.** <https://www.infoarena.ro/problema/padure>