

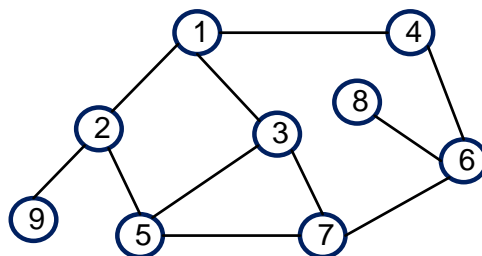
## MEMORAREA GRAFURILOR. PARCURI. APLICAȚII

A. Memorarea unui graf .....	2
B. Determinarea de drumuri minime din parcurgerea BF .....	3
C. Detectarea de cicluri și circuite .....	4
D. Tare-conexitate în grafuri orientate .....	5
E. 2- Conectivitate. Muchii și puncte critice .....	5
F. Sortarea topologică .....	6

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.

### Exerciții:

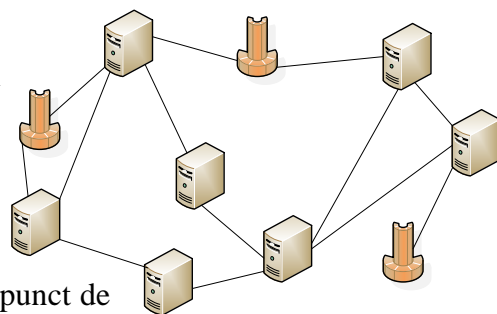
4. Propuneți modalități de reprezentare și pentru grafuri orientate și pentru multigrafuri neorientate/orientate (care admit muchii/arce multiple și bucle)
5. Precizați, în fiecare caz, pentru fiecare modalitate de reprezentare cum se pot determina numărul de muchii ale grafului, gradul fiecărui vârf (gradul interior și exterior al fiecărui vârf în cazul orientat).

## B. Determinarea de drumuri minime din parcurgerea BF

1. <https://infoarena.ro/problema/bfs>

2. a) Se dă o rețea neorientată cu  $n$  noduri și o listă de noduri reprezentând puncte de control pentru rețea. Se citește un nod de la tastatură. Să se determine cel mai apropiat punct de control de acesta și un lanț minim până la acesta  $O(n+m)$

b) Modificați programul de la a) astfel încât să rezolve aceeași cerință dar pentru o rețea orientată (determinarea celui mai apropiat punct de control și un drum minim până la acesta)



**Exemplu:** Dacă în graful considerat ca exemplu punctele de control sunt 8 și 9, și nodul de start este 1, cel mai apropiat punct de control de 1 este 9, aflat la distanța 2. Un lanț minim va fi 1, 2, 9

graf.in - cel de mai sus pe ultima linie se dau punctele de control	graf.out
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>	1 2 9

3. Se dă o matrice  $n*m$  ( $n, m \leq 1000$ ), cu  $p \leq 100$  puncte marcate cu 1 (restul valorilor din matrice vor fi 0). Distanța dintre 2 puncte ale matricei se măsoară în locații străbătute mergând pe orizontală și pe verticală între cele 2 puncte (distanța Manhattan). Se dă o mulțime  $M$  de  $q$  puncte din matrice ( $q \leq 1000000$ ). Să se calculeze cât mai eficient pentru fiecare dintre cele  $q$  puncte date, care este cea mai apropiată locație marcată cu 1 din matrice. (**Licență iunie 2015**)

### Datele de intrare:

Pe prima linie a fișierului „**graf.in**” se afla valorile  $n$  și  $m$  separate printr-un spațiu.

Următoarele  $n$  linii reprezintă matricea cu valori 1 și 0. În final, pe câte o linie a fișierului, se află câte o pereche de numere, reprezentând coordonatele punctelor din  $M$ .

### Date de ieșire:

Fișierul „**graf.out**” va conține  $q = |M|$  linii; pe fiecare linie  $i$  va fi scrisă distanța de la al  $i$ -lea punct din  $M$  la cel mai apropiat element marcat cu 1 în matrice, precum și coordonatele acelui element cu valoarea 1.

### Exemplu

graf.in	graf.out
5 4 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 1 1 1 2 1 4 4 1 5 3	2 [1, 3] 1 [1, 3] 1 [1, 3] 2 [3, 2] 2 [4, 4]

4. <http://www.infoarena.ro/problema/rj>

5. Se dă un graf neorientat și două noduri s și t. Să se afișeze toate lanțurile minime de la s la t.

6. <http://www.infoarena.ro/problema/graf>

### C. Detectarea de cicluri și circuite

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.**

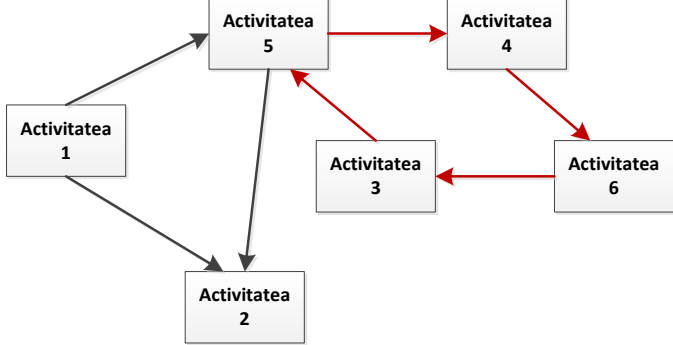
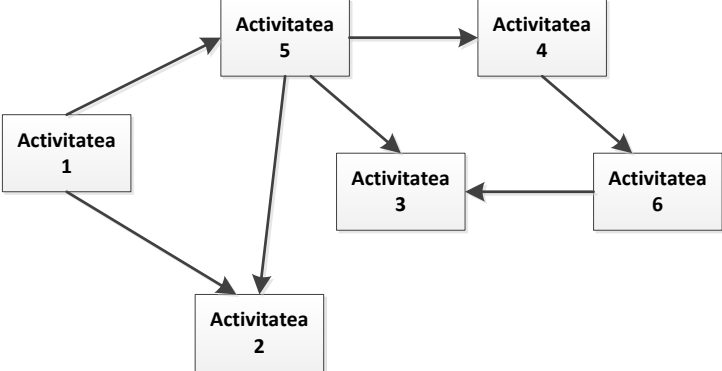
graf.in	graf.out
7 8 1 3 2 4 3 4 3 5 3 6 5 6 6 7 3 7	3 5 6 3 (nu neaparat in aceasta ordine; solutia nu este unica, un alt ciclu este de exemplu 3 6 7 3)

3. În cadrul unui proiect trebuie realizate  $n$  activități, numerotate  $1, \dots, n$ . Activitățile nu se pot desfășura în orice ordine, ci sunt activități care nu pot începe decât după terminarea altora. Date  $m$  perechi de activități  $(a, b)$  cu semnificația că activitatea trebuie să se desfășoare înainte de activitatea  $b$ , să se testeze dacă proiectul este realizabil, adică nu există dependențe circulare între activitățile sale. În cazul în care proiectul nu se poate realiza să se afișeze o listă de activități între care există dependențe circulare.

**Datele de intrare:** Pe prima linie a fișierului „graf.in” se afla valorile  $n$  și  $m$  separate printr-un spațiu. Pe fiecare dintre următoarele  $m$  linii sunt două numere  $a$  și  $b$  cu semnificația din enunț

**Date de ieșire:** Fișierul ”graf.out” va conține o listă de activități între care există dependențe circulare, separate prin spațiu, dacă astfel de activități există sau mesajul REALIZABIL altfel.

### Exemplu

<b>graf.in</b> 6 7 1 2 1 5 5 2 5 4 3 5 4 6 6 3		<b>graf.out</b> 5 4 6 3 (nu neaparat in aceasta ordine)
<b>graf.in</b> 6 7 1 2 1 5 5 2 5 4 5 3 4 6 6 3		<b>graf.out</b> REALIZABIL

## D. Tare-conexitate în grafuri orientate

1. Să se afișeze componentele tare conexes ale unui graf orientat  $O(n+m)$

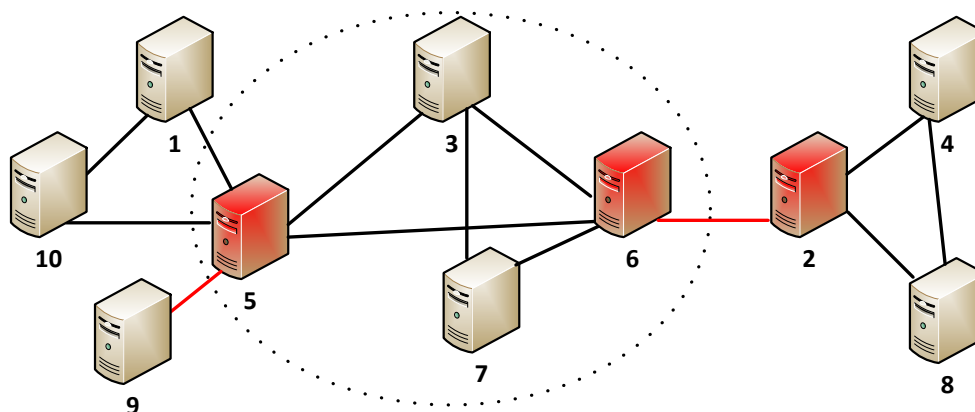
<https://www.infoarena.ro/problema/ctc>

## E. 2- Conectivitate. Muchii și puncte critice

1. Se dă o rețea cu  $n > 2$  noduri numerotate 1, 2, ..., n prin numărul de noduri n și perechile de noduri între care există legături directe prin care se pot trimite mesaje. Printr-o legătură directă pot comunica în ambele sensuri. Între oricare două noduri ale rețelei se pot trimite mesaje direct sau prin puncte intermediare (rețeaua este conexă).
  - a) O legătură directă în rețea este considerată critică dacă după defectarea sa există cel puțin două noduri ale rețelei între care nu se mai pot trimite mesaje. Să se determine toate legăturile critice ale rețelei. Dacă nu există astfel de legături se va afișa mesajul “rețea 2 muchie-conexa”  $O(n+m)$

b) Un nod al rețelei este considerat vulnerabil dacă după defectarea sa există cel puțin două noduri ale rețelei între care nu se mai pot trimite mesaje. Să se determine toate nodurile vulnerabile ale rețelei. Dacă nu există astfel de puncte se va afișa mesajul “retea biconexa”  $O(n+m)$

c) Să se determine un număr maxim de noduri care determină o subrețea conexă (formată păstrând doar legăturile directe între nodurile selectate) fără noduri vulnerabile. Se vor afișa nodurile acestei subrețele și legăturile directe dintre ele (în orice ordine).  $O(n+m)$



graf.in	graf.out
10 13	Legaturi critice
1 5	5 9
1 10	2 6
3 5	Noduri vulnerabile
5 6	2
5 9	5
5 10	6
3 6	Subretea
3 7	3 5 6 7
6 7	3 5
2 4	3 6
2 6	3 7
2 8	6 7
4 8	5 6

**Observație** – O rețea cu legături critice și noduri vulnerabile prezintă probleme de securitate, deoarece un atac extern asupra unei astfel de legături sau într-un astfel de nod întrerupe comunicarea între noduri ale rețelei.

2. <https://infoarena.ro/problema/biconex>

## F. Sortarea topologică

1. **Sortarea topologică** – Pentru **problema C2**, să se afișeze în plus, dacă proiectul se poate realiza, o ordine în care se pot efectua activitățile (astfel încât să se respecte dependențele dintre ele).  $O(n+m)$  <https://www.infoarena.ro/problema/sortaret>