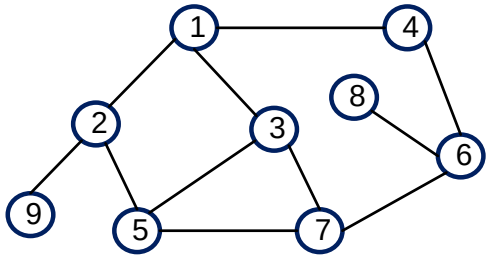
MEMORAREA GRAFURILOR. PARCURGERI. APLICAȚII

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** Propuneţi modalităţi de reprezentare şi pentru grafuri orientate și pentru multigrafuri neorientate/orientate (care admit muchii/arce multiple şi 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 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)

Teme obligatorii (1 punct – 0,2 fiecare problema)

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

1. a) **https://leetcode.com/problems/possible-bipartition/**

b) Modificați funcția de la a) astfel încât să returneze o împărțire a persoanelor în două grupuri conform cerinței, dacă o astfel de împărțire este posibilă, sub forma unui vector cu doua componente: lista persoanelor din primul grup și lista persoanelor din cel de al doilea grup:

```
vector<vector<int>> possibleBipartition(int n, vector<vector<int>>& dislikes)
def possibleBipartition(self, n: int, dislikes: List[List[int]]) -> List[List[int]]:
```

2. https://csacademy.com/contest/archive/task/check-dfs

3. a) **https://leetcode.com/problems/course-schedule-ii/**

b) Modificați funcția de la a) astfel încât să returneze, în cazul în care nu pot fi urmate toate cursurile, un vector de cursuri $[c_1, c_2, \ldots, c_k, c_1]$ care depind circular unele de altele (orice curs c_i trebuie urmat înaintea cursului c_{i-1}, iar c_k înaintea cursului c_1), iar în cazul în care pot fi urmate toate cursurile, un vector vid.

4. **https://www.infoarena.ro/problema/ctc**

5. 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.

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 **8 9**	2 1 3 2 2 1 2 0 0 **(Explicații:** **Punctele de constrol sunt 8 si 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)

Teme suplimentare (1 punct – 0,2 fiecare problema)

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

1. **https://leetcode.com/problems/critical-connections-in-a-network/**
2. **https://www.infoarena.ro/problema/padure**
3. **https://www.infoarena.ro/problema/graf**
4. **https://www.infoarena.ro/problema/lesbulan**
5. **https://leetcode.com/problems/max-area-of-island/**