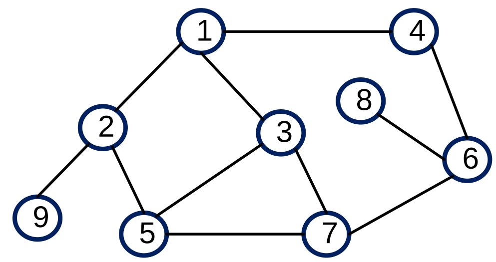
MEMORAREA GRAFURILOR. PARCURGERI. APLICAȚII

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 şi 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.

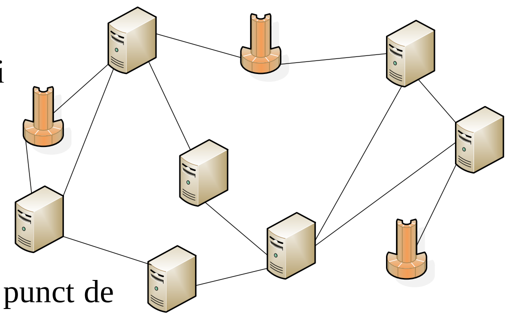
📖 **Exerciţii:**

4. Propuneţi modalităţi de reprezentare şi pentru grafuri orientate şi pentru multigrafuri neorientate/orientate (care admit muchii/arce multiple şi bucle)

5. Precizaţi, în fiecare caz, pentru fiecare modalitate de reprezentare cum se pot determina numărul de muchii ale grafului, gradul fiecărui vârf (gradul interior şi 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 şi 9, şi nodul de start este 1, cel mai apropiat punct de control de 1 este 9, aflat la distanţa 2. Un lanţ 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 **8 9**	1 2 9

3. Se dă o matrice n*m (n,m <= 1000), cu p <= 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 <= 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 conţine un ciclu elementar (nu este aciclic). În caz afirmativ **să se afişeze un astfel de ciclu**.

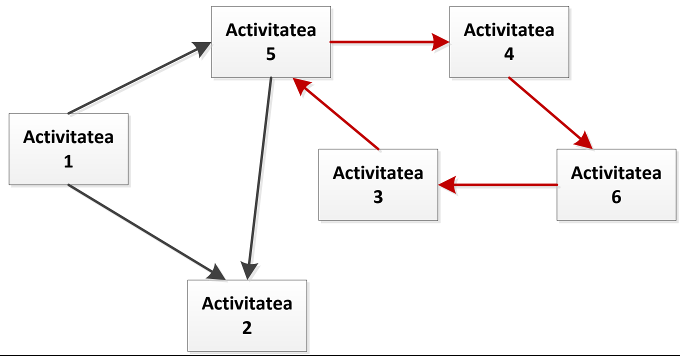
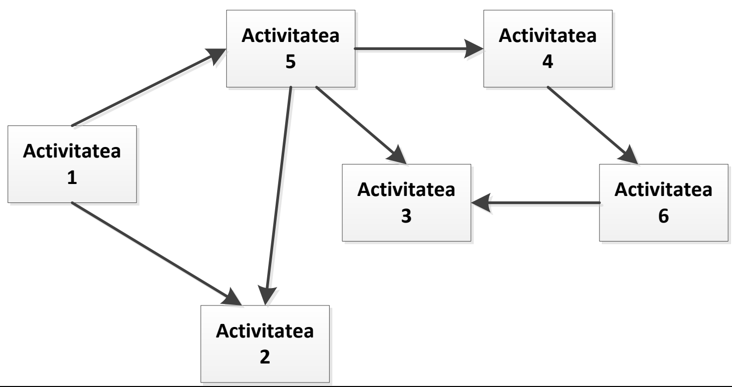
graf.in	graf.out
7 8 1 3 2 4 3 4 3 5 3 6 5 6 6 7 3 7	3 5 6 3 (nu neaparat in aceasta ordine; solutia nu este unica, un alt ciclu este de exemplu 3 6 7 3)

3. În cadrul unui proiect trebuie realizate n activități, numerotate 1,…,n. Activitățile nu se pot desfăşura 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

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

D. Tare-conexitate în grafuri orientate

1. **Să se afişeze componentele tare conexe ale unui graf orientat O(n+m)**
   https://www.infoarena.ro/problema/ctc

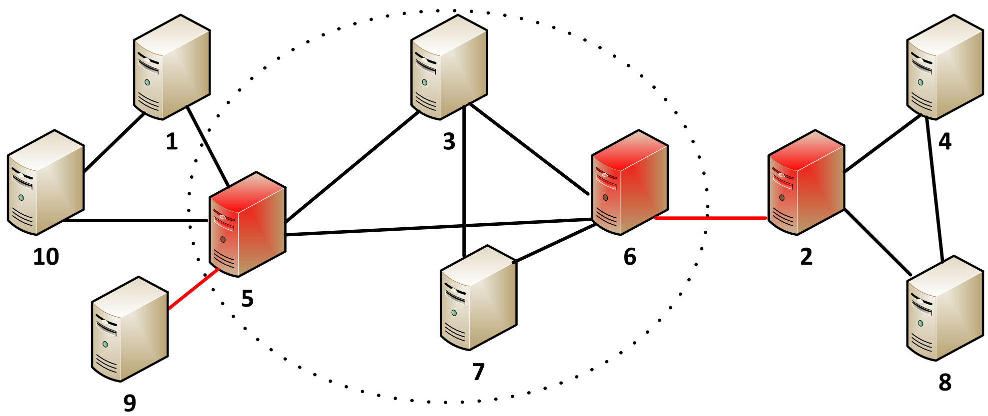
E. 2- Conectivitate. Muchii și puncte critice

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 “retea 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 1 5 1 10 3 5 5 6 5 9 5 10 3 6 3 7 6 7 2 4 2 6 2 8 4 8	Legaturi critice 5 9 2 6 Noduri vulnerabile 2 5 6 Subretea 3 5 6 7 3 5 3 6 3 7 6 7 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 afişeze 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**

Arbori parţiali de cost minim

Fişierul `grafpond.in` **are următoarea structură: numărul de vârfuri n, numărul de muchii m şi lista muchiilor cu costul lor (o muchie fiind dată prin extremităţile sale şi cost). Costul unei muchii este număr întreg.**

grafpond.in
5 7 1 4 1 1 3 5 1 2 10 2 3 2 4 2 6 4 5 12 5 2 11

1. a) Implementaţi algoritmul lui Kruskal pentru determinarea unui arbore parţial de cost minim al unui graf conex ponderat cu n vârfuri şi m muchii. Graful se va citi din fişierul **`grafpond.in`**. **O(m log n)** (+ şi versiunea **O(n^2 + m log n)**)
   https://www.infoarena.ro/problema/apm

   b) Modificaţi programul de la a) astfel încât să determine (dacă există) un arbore parţial de cost cât mai mic care să conţină 3 muchii ale căror extremităţi se citesc de la tastatură. Se vor afişa muchiile arborelui determinat.

2. Implementaţi algoritmul lui Prim pentru determinarea unui arbore parţial de cost minim al unui graf conex ponderat cu n vârfuri şi m muchii. Graful se va citi din fişierul **`grafpond.in`**. **O(m log n)** (+ şi versiunea **O(n^2)**) **https://www.infoarena.ro/problema/apm**

3. **Clustering.** Fişierul `cuvinte.in` conţine cuvinte separate prin spaţiu. Se citeşte de la tastatură un număr natural k. Se consideră distanţa Levenshtein între două cuvinte https://en.wikipedia.org/wiki/Levenshtein_distance.
   Să se împartă cuvintele din fişier în k clase (categorii) nevide astfel încât gradul de separare al claselor să fie maxim (= distanţa minimă între două cuvinte din clase diferite) - v. curs; se vor afişa pe câte o linie cuvintele din fiecare clasă şi pe o altă linie gradul de separare al claselor. **O(n^2 log n) / O(n^2)**

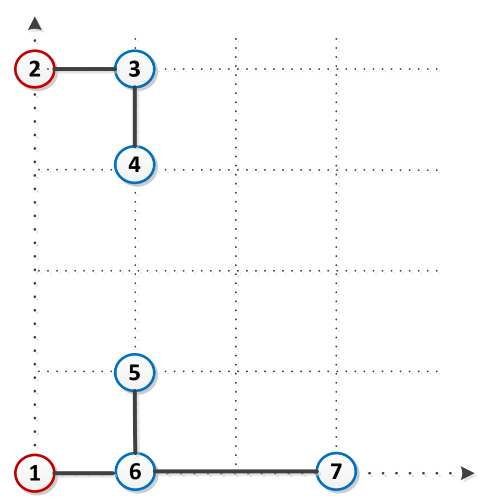
cuvinte.in	Ieșire pentru k=3 (clasele nu sunt unice, dar gradul de separare da)
martian care este sinonim ana case apa arbore partial minim	care este ana case apa arbore martian partial sinonim minim 4

4. **Conectarea cu cost minim a nodurilor la mai multe surse** – varianta a problemei http://www.infoarena.ro/problema/retea2 (doar anumite clădiri pot fi conectate)
   Damia s-a hotărât să își deschidă N centrale electrice, numerotate 1,2,…, N. În orașul ei sunt M blocuri care trebuie să primească curent, numerotate N+1, N+2,…, N+M. Un bloc primește curent electric dacă este conectat la alt bloc care primește curent electric sau dacă este conectat

la o centrală electrică. Se cunosc coordonatele (euclidiene) celor N centrale și **ale** celor M blocuri. Pentru a conecta două dintre aceste puncte trebuie plătit un cost egal cu distanța euclidiană dintre ele. Dintre toate aceste puncte însă doar unele pot fi conectate în mod direct. Ajutați-o pe Damia să determine costul minim pentru a transmite curent electric către toate blocurile.

Date de intrare: În fişierul de intrare `retea2.in` se găsesc trei numere naturale N, M si E. Pe fiecare dintre următoarele M+N linii este câte o pereche de numere întregi reprezentând coordonatele în plan ale unei clădiri (centrală sau bloc). Primele N coordonate vor fi coordonatele asociate centralelor, următoarele M fiind coordonatele asociate blocurilor de locuințe. Pe următoarele E linii se găsesc perechi de indici i,j ∊ {1,…,N+M} reprezentând clădiri care pot fi conectate in mod direct.

Date de ieşire: În fişierul `retea2.out` pe prima line va fi un număr real, reprezentând costul minim pentru a transmite curent electric blocurilor în modul descris în enunț, iar pe următoarele linii perechile de clădiri care trebuie conectate **O(E log (M+N))**

retea2.in	retea2.out	
2 5 9	6	
0 0	1 6	
0 4	5 6	
1 4	6 7	
1 3	2 3	
1 1	3 4	
1 0		
3 0		
1 2		
2 3		
3 4		
4 5		
5 6		
1 6		
3 5		
5 7		
6 7		

5. **Graf dinamic** - Se citesc din fişierul **`grafpond.in`** informații despre un graf ponderat. Folosind unul dintre algoritmii Kruskal sau Prim se determină muchiile unui arbore parțial de cost minim T al grafului **O(mlogn)**.
   Se citeşte de la tastatură o nouă muchie *e* (dată prin extremitățile sale și costul ei) care trebuie adăugată la G.
   a) Să se afişeze costul maxim al unei muchii din ciclul elementar pe care îl închide *e* în *T* (nu în G) și extremitățile acesteia (**O(n)**)
   b) Folosind eventual a), determinați dacă prin adăugarea muchiei *e* noul graf obținut are un arbore parțial de cost minim cu costul mai mic decât G (**O(1)/O(n)**).
   (O problemă similară, dar mai dificilă: https://www.infoarena.ro/problema/online)

grafpond.in	iesire

`5 7`
`1 4 1`
`1 3 5`
`1 2 10`
`2 3 2`
`4 2 6`
`4 5 12`
`5 2 11`
	`Muchiile apcm in G:`
	`1 4`
	`2 3`
	`1 3`
	`2 5`
	`Cost 19`
	`Muchia de cost maxim din ciclul inchis de 3 5`
	`in apcm este 2 5 de cost 11`
	`Dupa adaugarea muchiei apcm are costul 12`

De la tastatura `3 5 4`	

6. **Second best minimum spanning tree** – Implementați un algoritm eficient pentru determinarea **primilor doi** arbori parțiali cu cele mai mici costuri, pentru un graf conex ponderat, în ipoteza că **muchiile au costuri distincte** (**O(n^2)**). Graful se va citi din fişierul `grafpond.in`.

Determinarea următorului arbore cu costul cel mai mic după arborele parțial de cost minim poate fi utilă spre exemplu ca soluție secundară în probleme care se reduc la determinarea unui apcm (conectare, revizie rețea etc)

grafpond.in	iesire

`6 7`
`1 2 13`
`1 3 14`
`2 3 16`
`1 4 11`
`1 5 12`
`4 5 15`
`5 6 10`
	`Primul`
	`Cost 60`
	`Muchii`
	`1 2`
	`1 3`
	`1 4`
	`1 5`
	`5 6`
	`Al doilea`
	`Cost 62`
	`Muchii`
	`1 2`
	`2 3`
	`1 4`
	`1 5`
	`5 6`

Drumuri minime

Ca și în laboratorul trecut, fișierul `grafpond.in` **are următoarea structură: numărul de vârfuri n, numărul de muchii/arce m şi lista muchiilor/arcelor cu costul lor (o muchie fiind dată prin extremităţile sale şi cost).**

grafpond.in
5 7 1 4 1 1 3 5 1 2 10 2 3 2 4 2 6 4 5 12 5 2 11

Justificaţi complexitatea+corectitudinea algoritmilor propuşi.

1. **Drum critic (Critical Path Method).** Se citesc din fișierul `activitati.in` următoarele informații despre activitățile care trebuie să se desfășoare în cadrul unui proiect:
   - n – numărul de activități (activitățile sunt numerotate 1,…, n)
   - d_1, d_2, …., d_n durata fiecărei activități
   - m – număr natural
   - m perechi (i, j) cu semnificația: activitatea i trebuie să se încheie înainte să înceapă j

   Activitățile se pot desfășura și în paralel.

   a) Să se determine timpul minim de finalizare a proiectului, știind că acesta începe la ora 0 (echivalent – să se determine durata proiectului) și o succesiune (critică) de activități care determină durata proiectului (un drum critic – v. curs) **O(m + n).**

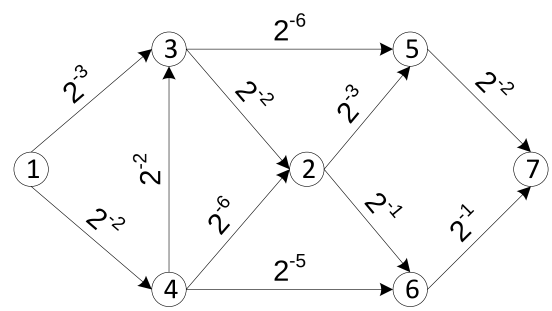
   b) Să se afișeze pentru fiecare activitate un interval posibil de desfășurare (!știind că activitățile se pot desfășura în paralel) **O(m + n).**

activitati.in	**iesire**
6 **7 4 30 12 2 5** **6** **1 2** **2 3** **3 6** **4 3** **2 6** **3 5**	Timp minim 47 Activitati critice: 4 3 6 1: 0 7 2: 7 11 3: 12 42 4: 0 12 5: 42 44 6: 42 47

Robert Sedgewick and Kevin Wayne, Algorithms, 4th Edition, Addison-Wesley, 2011.

2. Se citesc din fișierul `grafpond.in` informații despre un graf **neorientat** ponderat și de la tastatură un număr k, o listă de k puncte de control ale grafului și un vârf s. Determinați cel mai apropiat punct de control de vârful s și afișați un lanț minim până la acesta, folosind algoritmul lui **Dijkstra** (problema B.2. din laboratorul 1 pentru cazul ponderat) - **O(m log(n)).**

3. (**v. Seminar – Aplicaţie Dijkstra**) Pentru fiecare arc al unei reţele de comunicaţie acestui graf se cunoaşte o pondere pozitivă subunitară reprezentând probabilitatea ca legătura corespunzătoare să nu se defecteze (de forma $1/2^p = 2^{-p}$). Aceste probabilităţi sunt independente, deci **siguranţa unui drum** este egală cu produsul probabilităţilor asociate arcelor care îl compun. Arătaţi că problema determinării unui drum de siguranţă maximă de la un vârf de start s la un vârf destinaţie t (accesibil din s) se poate reduce la o problemă de determinare a unui drum minim între s şi t (pentru un graf cu ponderile modificate). Pornind de la acest fapt, implementaţi un algoritm bazat pe algoritmul lui **Dijkstra** pentru determinarea unui drum de siguranţă maximă între două vârfuri s şi t citite de la tastatură pentru o reţea orientată dată în fişierul retea.in prin următoarele informaţii:
   - n, m – numărul de vârfuri, respectiv arce
   - m linii conţinând triplete de numere naturale i j p cu semnificaţia: (i,j) este arc în reţea cu probabilitatea să nu se defecteze egală cu 2^{-p} **O(m log(n)).**



4. **Drumuri minime din surse multiple** http://www.infoarena.ro/problema/catun **O(m log(n)).**

5. **Bellman Ford** Se dă un graf orientat ponderat (în fișierul `grafpond.in`) și un vârf s citit de la tastatură. Dacă graful nu conține circuite negative accesibile din s afișați câte un drum minim de la s la fiecare dintre celelalte vârfuri accesibile din s, altfel afișați un astfel de circuit (folosind algoritmul Bellman Ford) **O(nm)**

6. **Floyd-Warhsall**

   a) Dat un graf orientat ponderat (în fisierul `grafpond.in`), afișați matricea distanțelor dacă graful nu conține circuite de cost negativ și un circuit cu cost negativ în caz contrar. **O(n^3)**

   b) Fie G un graf neorientat ponderat. Pentru două vârfuri u şi v ale lui G, notăm cu d(u, v) **distanța** de la vârful u la vârful v.
   Pentru un vârf v, **excentricitatea** lui v este cea mai mare distanță de la acest vârf la celelalte vârfuri:

   $$e(v) = \max\{d(v, u) \mid u \in V\}$$

   Excentricitatea minimă a vârfurilor se numește **raza** grafului:

   $$r(G) = \min\{e(v) \mid v \in V\}$$

   Mulțimea vârfurilor cu excentricitatea minimă (egală cu r(G)) se numeşte **centrul** grafului:

$$c(G) = \{ v \in V \mid e(v) = r(G)\}$$

Excentricitatea maximă a vârfurilor se numeşte **diametrul** grafului; altfel spus, diametrul este cea mai mare distanță dintre două vârfuri:

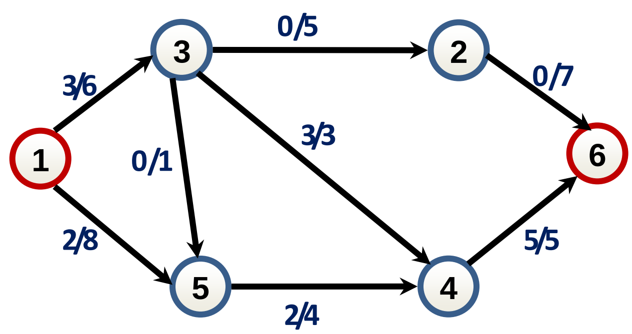
$$diam(G) = \max\{e(v) \mid v \in V \} = \max\{d(u, v) \mid v, u \in V \}$$

Se citesc din fișierul `grafpond.in` informații despre un graf **neorientat** ponderat G. Să se determine, folosind algoritmul **Floyd-Warhsall**, raza, diametrul, centrul grafului și un lanț diametral (un lanț minim P între două vârfuri u și v cu ponderea w(P)=d(u,v)=diam(G)). **O(n**^{3})

Fluxuri în reţele de transport

1. **Flux maxim**. Se consideră o reţea de transport (care verifică ipotezele din curs) şi un flux în această reţea. Se citesc din fişierul `retea.in` următoarele informaţii despre această reţea: numărul de vârfuri n (numerotate 1…n), două vârfuri s şi t reprezentând sursa şi destinaţia, numărul de arce m şi pe câte o linie informaţii despre fiecare arc: extremitatea iniţială, extremitatea finală, capacitatea arcului şi fluxul deja trimis pe arc.
   a) Să se verifice dacă fluxul dat este corect (respectă constrângerile de mărginire şi conservare) şi să se afişeze un mesaj corespunzător.
   b) Să se determine un flux maxim în reţea pornind de la acest flux, prin revizuiri succesive ale fluxului pe s-t lanţuri nesaturate de lungime minimă (Algoritmul Ford - Fulkerson va porni de la fluxul dat, nu de la fluxul vid). Se vor afişa
      - Valoarea fluxului obţinut **şi fluxul pe fiecare arc**
      - Capacitatea minimă a unei tăieturi în reţea şi arcele directe ale unei tăieturi minime **O(mL), L= capacitatea minimă a unei tăieturi / O(nm^2)**
   c) https://www.infoarena.ro/problema/maxflow

retea.in	**iesire**
6	DA
1 6	10
8	1 3 6
1 3 6 3	1 5 4
1 5 8 2	3 2 5
3 2 5 0	3 4 1
3 4 3 3	5 4 4
5 4 4 2	2 6 5
2 6 7 0	4 6 5
4 6 5 5	3 5 0
3 5 1 0	10
	1 3
	5 4



2. **Cuplaj maxim în graf bipartit**. Se citesc din fişierul `graf.in` următoarele informaţii despre un graf neorientat **<u>bipartit conex</u>**: numărul de vârfuri n>2, numărul de muchii m şi lista muchiilor (o muchie fiind dată prin extremităţile sale). Să se determine un cuplaj de cardinal maxim în acest graf reducând problema la o problemă de flux maxim şi folosind apoi algoritmul Ford-Fulkerson. Se vor afişa muchiile cuplajului maxim obţinut (vârfurile sunt numerotate 1..n, dar **nu este neapărat ca vârfurile de aceeaşi culoare să fie numerotate consecutiv**) **O(nm)**
   Dacă graful dat la intrare <u>nu</u> este bipartit, se va afişa un mesaj corespunzător și un ciclu impar al grafului

graf.in	iesire (nu este unica solutia)
8 9 1 2 1 3 2 4 3 4 2 5 3 5 3 7 6 7 7 8	1 2 3 4 6 7

3. **Construcția unui graf orientat cu secvențele de grade de intrare și ieșire date**. Se citesc din fișierul **`secvente.in`**: un număr natural n>2, o secvență s_1 de n numere naturale și o secvență s_2 de n numere naturale. Să se construiască, dacă se poate, un graf cu secvența gradelor interne s_1 și cu secvența gradelor externe s_2 (reducând problema la o problemă de flux maxim). În caz afirmativ se vor afișa arcele grafului, altfel se va afișa mesajul NU. **O(mn²)** (unde m = suma numerelor din s_1 = numărul de arce ale lui G)

secvente.in	iesire (nu este unica solutia)
3 2 1 1 1 1 2	1 3 2 1 3 1 3 2

4. (Suplimentar) Implementați o altă aplicație discutată la curs/seminar care se reduce la problema determinării unui flux maxim sau a unei tăieturi minime în rețea