



$$\text{I) } 1) \sum_{x \in V} d_x^2 = \sum_{xy \in E} (d_x + d_y)$$

$$\begin{array}{ll} d_1^2(1) = 4 & d_1 + d_2 = 3 \\ d_1^2(2) = 1 & d_1 + d_3 = 4 \\ d_1^2(3) = 4 & d_3 + d_4 = 3 \\ d_1^2(4) = 1 & \end{array}$$

$$2) \sum_{v \in V} d_v^2 = \sum_{uv \in E} (d_u + d_v)$$

$$m_a = \frac{a+b}{2} \quad m_g = \sqrt{ab} \geq \sum_{uv} 2\sqrt{d_u d_v} \geq 2 \sum_{uv \in E} \sqrt{d_u d_v}$$

$$m_a \geq m_g \Rightarrow \frac{a+b}{2} \geq 2\sqrt{ab}$$

$$3) \quad n \geq 2 \quad \sum_{uv \in E} \frac{1}{\sqrt{d_u d_v}}$$

$$\sum_{uv \in E} \left( \frac{1}{d_u} + \frac{1}{d_v} \right) \geq 2 \cdot \sqrt{\frac{1}{d_u} \cdot \frac{1}{d_v}} \geq \sum_{uv \in E} \frac{1}{\sqrt{d_u d_v}}$$

"n"

$$\text{II) } 1 \leq d_1 \leq d_2 \dots \leq d_n \leq n-1$$

Havel-Hakimi:

<u>N</u>	1 2 3 4 5
	1 2 2 2 3

(Graf cu gradele)

Algoritm:

→ Sortare desc:

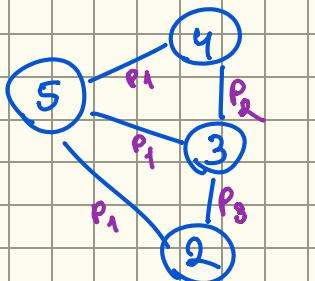
5	4	3	2	1
3	2	2	2	1
0	1	1	1	0

restoresc

4	3	2	1	5
1	1	1	0	0

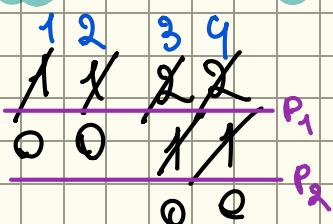
3	2	1	5	4
1	1	1	0	0

... e v

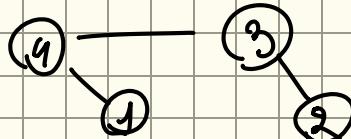


$P_i = \text{Pasul } i.$

# Pentru Arborei.

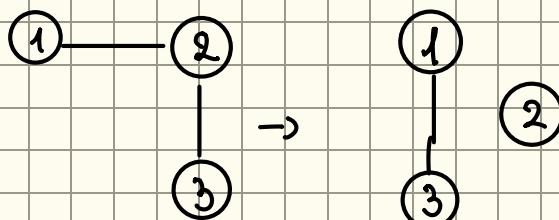


Luăm nod de gr maxim și unim cu cel de gr 0



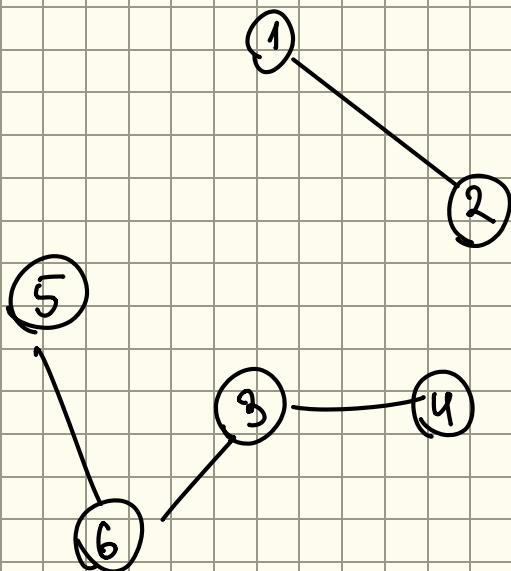
$\Rightarrow$  maxim și minimu.

## Graf complementar.

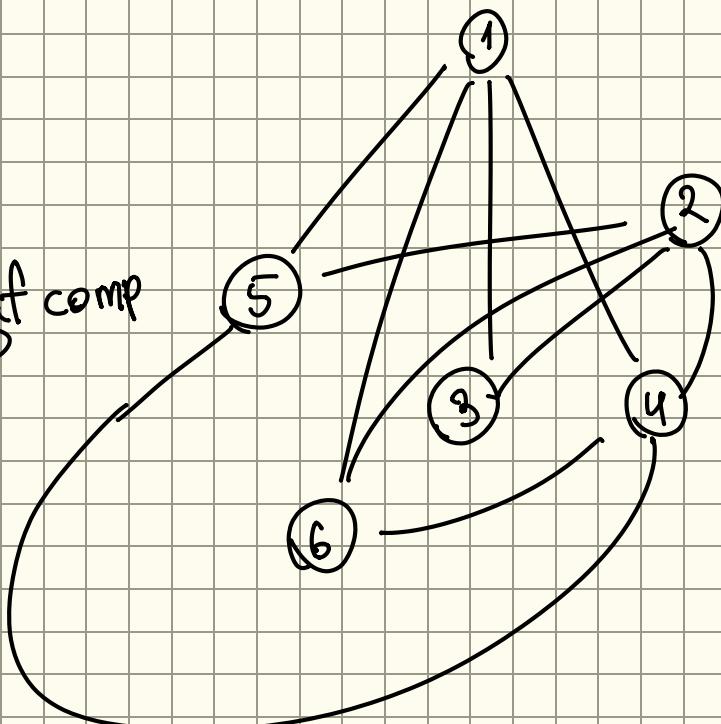


Type Shift

$\uparrow shift$ .



Graf comp  
=>



$$d_G(x) + d_{\bar{G}}(x) = n - 1.$$

# ~Grade~

1) Grup cu  $n \geq 3$  persoane care  $\exists$  rel de prietenie reciprocă

$\Rightarrow \exists 2$  persoane care au același nr de prietenie

$$\underbrace{0 \leq d_i \leq n-1}_{n}$$

$n$  variante pt n gradul unui nod și dacă ar fi distințe

$\Rightarrow$  seav gr lui  $G \subseteq \{0, 1, 2, \dots, n-1\}$

nu se poate  $\rightarrow \exists G$  nf de gr o si unul de  $n-1$

adjacent  
cu  
toate.

2) Algoritm pt a determina un graf  $G$  cu sevența gradelor date.

$\Rightarrow$  (! și construcția). Not  $D_0 = \{d_1, d_2, \dots, d_n\}$

$\rightarrow$  cond necesare:  $d_1 + \dots + d_n = 2$  pară  
 $\bullet 0 \leq d_i \leq n-1$ .

$D_0 = \{1, 1, 3, 3\} \rightarrow$  nu!

$\{1, 1, 3, 3, 5, 5\}$

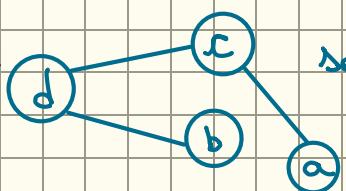
## Algoritm de construcție

- Cu ce vîrf pornim construcția  $\rightarrow$  vf grad maxim

- Ce vecini aleg  $\rightarrow$  pe cei cu gradele cele mai mari dintre cele rămasse.

$$\begin{matrix} a & b & c & d \\ \{1, 1, 2, 2\} \end{matrix}$$

max cu max:

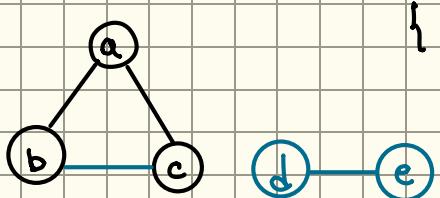


$$D_0' = \{1, 0, 1\}$$

Graful obținut este conex dacă  $d_1 + d_2 + \dots + d_n \geq 2(n-1)$

(în particular arbore  
dacă  $= 2(n-1)$ )

$\{2, 2, 2, 1, 1\}$   
a b c d e



$\{1, 1, 1, 1, 1\}$   
b c d e

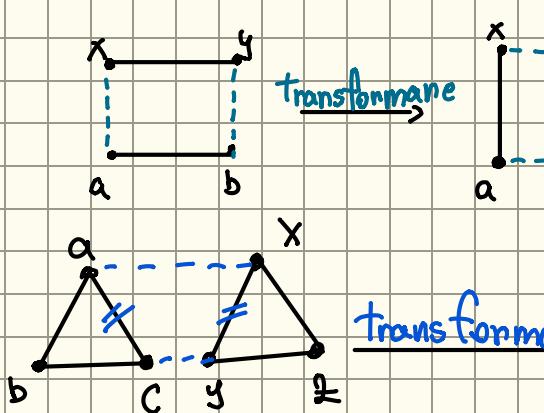
! nu neapărat conex!

## Algoritmul Havel-Hakimi

- $P_0 \rightarrow$  Verificarea condițiilor necesare.
- Eliminăm di din  $S_0$ .
- Cât timp există  $n > 0$  în  $S_0$  alegem di maxim eliminăm di din  $S_0$  alegem cele mai mici di elemente rămasă în  $S_0$  și unim i cu nodurile corespunzătoare
- Scadem cu i aceste elemente
- dacă obținem  $S_0$  NU  $\Rightarrow$  (stop)

Ar trebui să democă că dc  $\exists G$  cu  $S(G) = S_0$ ,  $\exists G^*$  cu secu gradelelor  $S_0$  în care vf de grad maxim este adjacent cu vârfuri cu gradele cele rămasă. (construcția nu e particulară)

OBS:



- graf cu aceeași secu de grade.
- graf conex cu aceeași secu de grade.

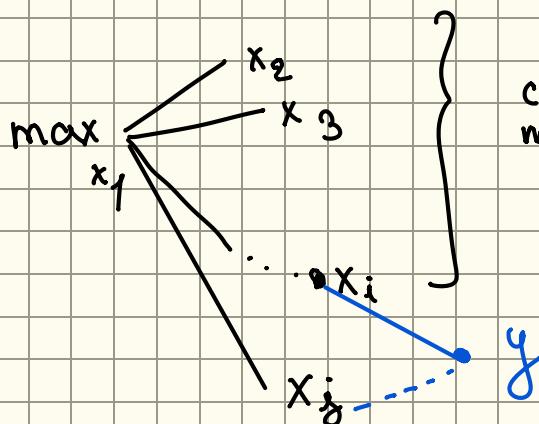
Dem: Pp cā există  $s(G) = s_0$

Construim  $G^*$  cu  $s(G^*) = s_0$  în care max  $\leftarrow \{ \max \}$

Folosin Transf pe pătrat(tipt)

Pt că  $d_1 \geq d_2 \geq \dots \geq d_n$  și în  $G$  vîlă  $x_1$  nu este adiacent cu cel puțin  $1$  dintre  $x_1, x_2, x_3, \dots, x_{d_1+1}$ .

Fie  $x_i$  primul care care  $x_i$  nu este adiacent.



$\exists x_j$  cu  $j > d_1 + 1$  (de grad mai mic) ca care  $x_1$  e adiacent ( $!x_1$  are grad  $d_1$ )

Decoarece  $d_i \geq d_j \Rightarrow \exists y$  ca  $x_i, y \in \mathbb{F}_i$  și  $x_{ij}, y \in E$

Folosind transformarea t pt pătratul P obținem un graf (!) cu aceeași secu de grade în care  $x_1$  este adiacent cu  $x_i$ .

Succesiv, fol transformarea obținem  $G^*$  cu  $s(G^*) = s_0$ . Se căre  $x_1$  - adiacent  $\underbrace{x_2, x_3, \dots, x_{d_1+1}}_{vîlă de gr max}$

OBS: De fapt alegerea lui  $x_1$  ( $d_1$  max) nu intervine în corectitudine.

(imp. sunt doar vecinii de gr. maxim).

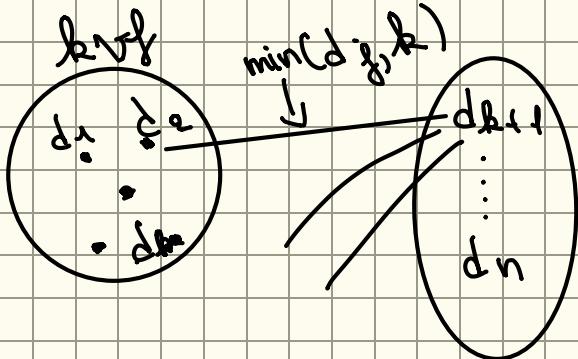
Cond. necesară și suficientă: Teorema ERDOS - GALLIAN

$\lambda_0 = \{d_1 \geq d_2 \geq \dots \geq d_n\}$  cu suma pară,  $0 \leq d_i \leq n-1$

este secvența gradelor unui grad  $\Leftarrow$

$$\forall k \quad d_1 + d_2 + \dots + d_k \leq k(k-1) + \sum_{j=k+1}^n \min(k, d_j)$$

(2 · nn max de muchii)



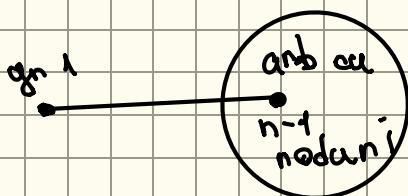
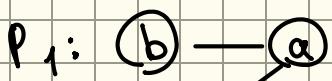
3. Const unui arbore cu secu gr dată.

$$\Delta_0 = \{d_1, \dots, d_n\} \quad d_1 + \dots + d_n = 2(n-1) \quad 0 < d_0 \leq n-1$$

$\exists$  T arbore cu  $s(T) = \Delta_0$  (+ construcție)

$$\Delta_0 = \{ \begin{matrix} a & b & c & d & e & f & g & h & i \\ 4 & 2 & 2 & 3 & , & 1 & & & \end{matrix} \}$$

Aleg: Cons un vif Terminal (de gr min / trb să fie 1)  
+ unim cu un vif de gr max. (suficient  $> 1$ ; pt  $n > 3$ )



După acest pas, secu. verifică în cont. că suma gradelor din secvență verifică în cont că suma  $2(n-1)$  unde  $n \leq n-1$ , deoarece suma scade cu 2 după fiecare pas.

$$\text{Dc } d_1 + d_2 + \dots + d_n = 2(n-1)$$

$$d_1 > \dots > d_n$$

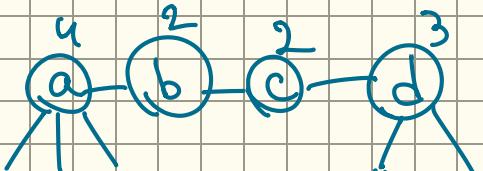
$$\Rightarrow d_n = 1. \quad (\text{Altfel } d_1 + \dots + d_n \geq 2n)$$

$$2(n-1)$$

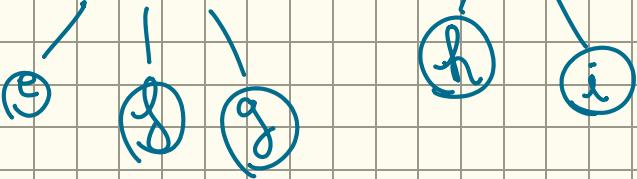
$\Rightarrow$  fol const inductiv un arbore.

Varianta mai simplă  $\rightarrow$  Omidă: const corpul (un lant care unește vif de gr 1)

$$\{ 4, 2, 2, 3, 1 \}$$



$\leftarrow$  exact câte picioare  
trb deoarece  $d_1 + \dots + d_n = 2(n-1)$

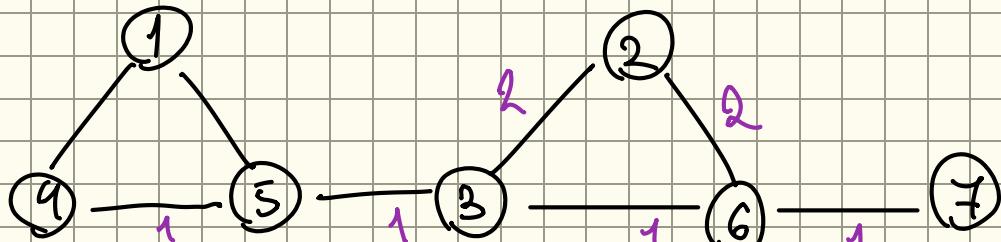


$\begin{matrix} 1 & 1 \\ 2 & 2 & 1 \end{matrix}$        $\leftarrow$       2

16 oct 2015 Seminar 2

### Componente biconexe. Muchii critice, puncte critice

$G$  neorientat



muchii vulnerabile (critice):  $(6,7); (3,5)$

muchie este în critică (puncte "bridge") dacă prin el ei obținem mai multe componente conexe.

a muchie conex: în  $G$  unde dacă eliminăm 2 puncte pt ca  $G$  să nu mai fie conex.

Pb: Determinați toate m.c. ale unui graf.

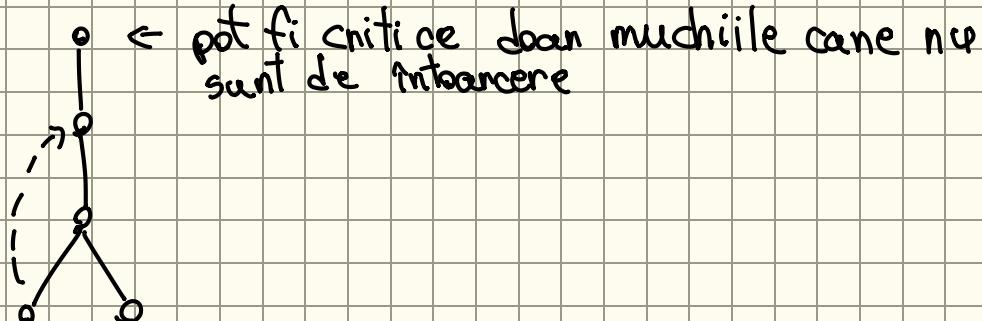
Det. Dacă un graf este 2-conex.

(V1) Testăm pe rând fiecare muchie

(V2) Cicluri:

Obs.: o muchie  $\in$  ciclu  $\Rightarrow$  nu este critică.

DFS:



id: pt un nod este suf. să considerăm doar în de întoarcere „cel mai sus”

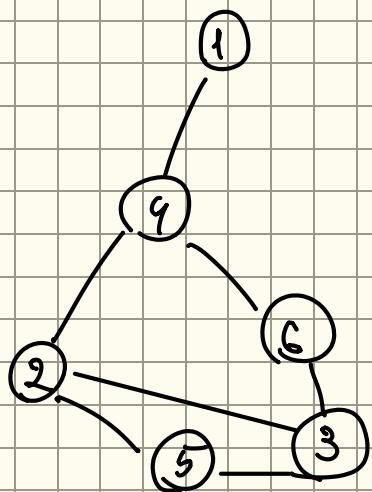
(la un nivel căt mai mic.)

Mai Util

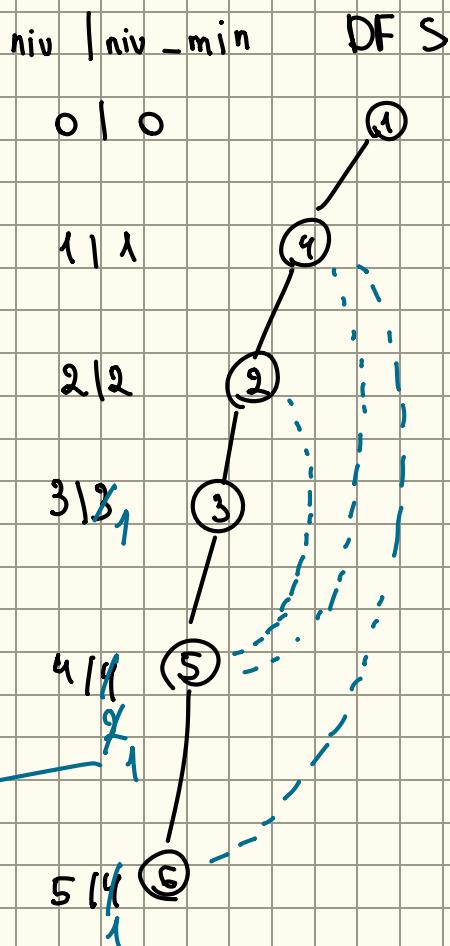
pt fiecare

niv - multij cît de sus este prim în ciclă

„cât de sus se poate ajunge din i  
sau dintr-un deschidere al lui prim  
m. de întoarcere”



în dr. lui  
5 rămâne doar  
m. de înt. (5,4)  
către niv 1



DFS(i)

```

viz[i] = 1
niv_min[i] = niv[i]
for j ∈ E (j in l. a<i,j)
  if(viz[j] == 0)
    niv[j] = niv[i] + 1
    DFS(j)
    // testă j este m.c
  
```

if  $niv\_min[c_j] > niv[i]$   
 $i, j$  este m.c.  
 $niv\_min[c_i^j] = \min(niv\_min[i], niv\_min[j])$

else

if  $niv[j] < niv[i] - 1$  //  $j$  nu este tată lui  $i$ .  
 $niv\_min[i] = \min(niv\_min[i], niv[j])$  ← întoarcere directă

23 Oct 2025 lab 2

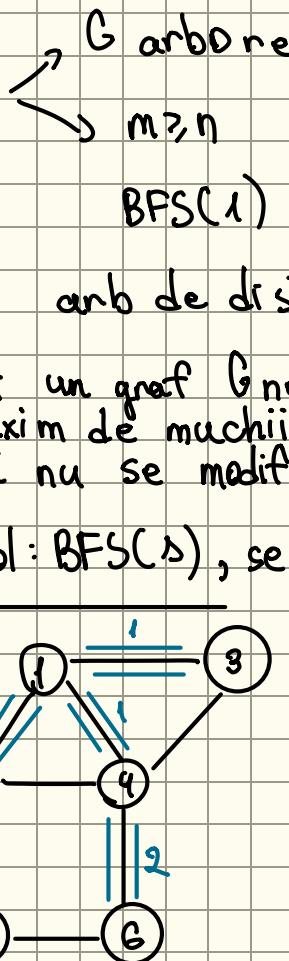
30 Oct 2025 seminar 3

Ex: 1) Se dă un graf conex  $\rightarrow m, n$  muchii

a) Determinați cel puțin 2 arbori parțiali ai lui  $G$ .

- DFS(1)
- o muchie care nu aparține arb DFS

b) În ce caz pt  $G$  conex arboarele BFS(1) = arborele DFS(1)

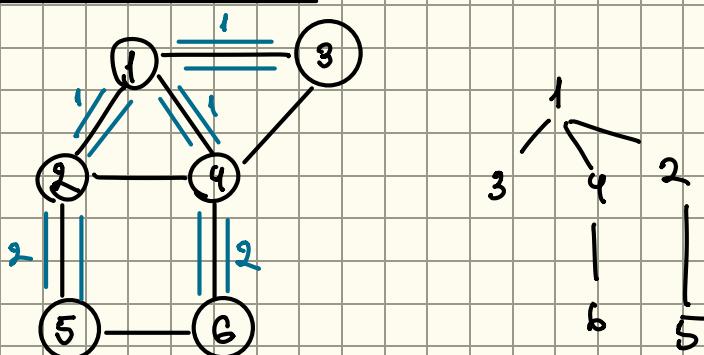
  
 G arbore ar.  $BFS(1) = ar. DFS(1) = G$   
 m, n

$$BFS(1) = DFS(1)$$

arb de dist 1      DFS

2) Dat un graf  $G$  neorientat conex, să se eliminate din  $G$  un nr maxim de muchii astfel încât distanțele de la un nod la celelalte să nu se modifice. (t să se afișeze)

Sol:  $BFS(s)$ , se elimină toate muchiile  $\notin$  arb  $BFS(s)$

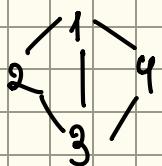
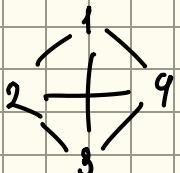


3)  $G_1, G_2$   $V_1 = V_2 = \{1, \dots, n\}$  conexe.

Dacă  $BFS(1, G_1) = BFS(1, G_2)$ ; vecinii în ordine crescătoare  
 $DFS(1, G_1) = DFS(1, G_2)$

$$\Rightarrow G_1 = G_2$$

exp:



$\text{arb DFS}(1, G_1) = \text{arb BFS}(1)$

$\text{BFS}(1) = 1 2 3 4$

$\text{DFS}(1) = 1 2 3 4$

$\text{BFS}(1) = 1 2 3 4$

$\text{DFS}(1) = 1 2 3 4$

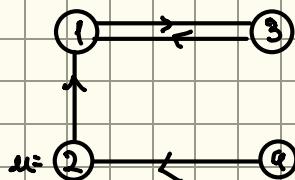
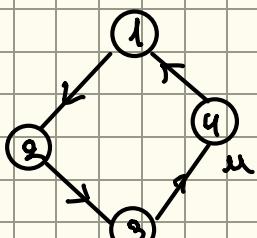
Teme: Dar dacă au și aceeași nr de muchii.

#### 4) Graf orientat

for  $u \in V = \{1, 2, \dots, n\}$   
if  $(v_i \neq u \wedge v_i \neq 0)$

$\text{DFS}(u) \rightarrow G$  așa că  $\text{DFS}(u)$  se vizitează doar  $u$   
deși  $d^-(u), d^+(u) \neq 0$

(vecinii  $\rightarrow$  în ordine crescătoare)



$\text{DFS}(1) \rightarrow 1 \leftarrow 2 \leftarrow 4 \leftarrow 3$   
 $\text{DFS}(2) \quad \quad \quad \text{DFS}(4)$   
 $\downarrow \quad \quad \quad \downarrow$   
 $5$

5) Se dă un graf neorientat  $G$  cu  $m$  muchii,  $n$  noduri.

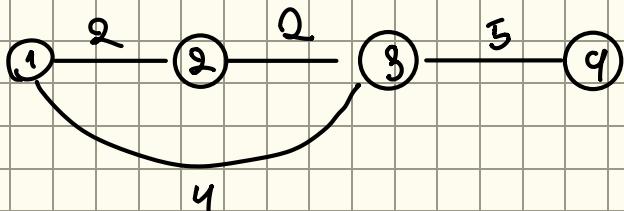
Complex pt un alg care det dacă  $G$  e aciclic.

$m \geq n-1+1 = n$  are cicluri,

$m < n \rightarrow O(n)$

APCM:

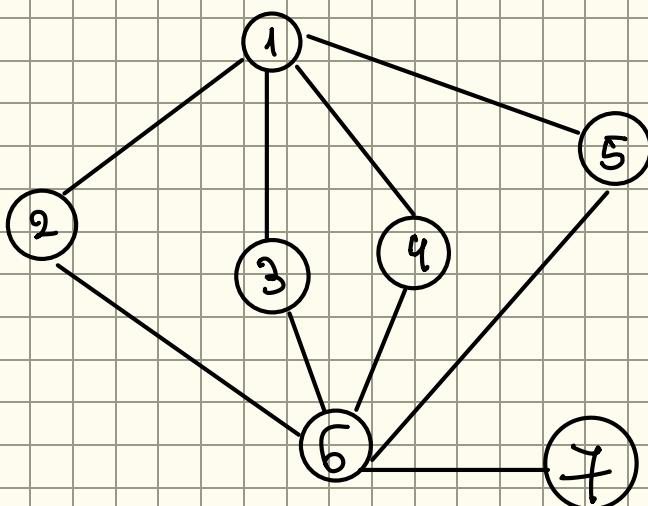
6) a) Dati ex de un graf cu exact doi arbori cu 2



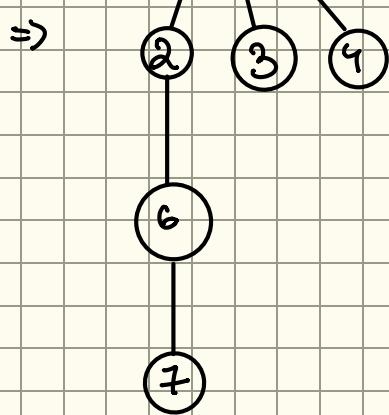
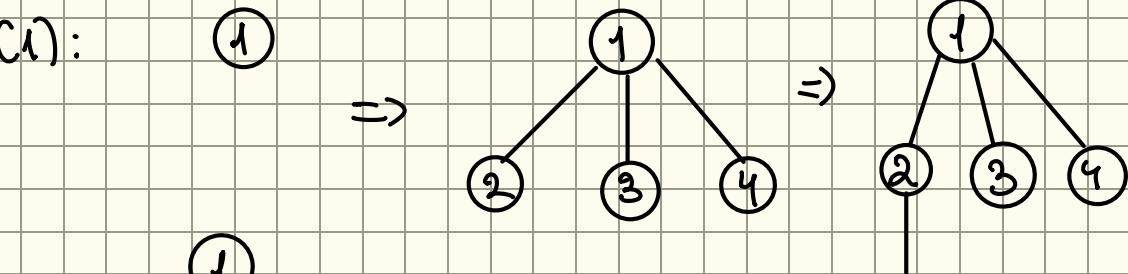
Temă : b)  $M$  distințe  $\Rightarrow$  apcm (unic?)

TEMĂ 1 - SEMINAR 22 noiembrie

1. a)

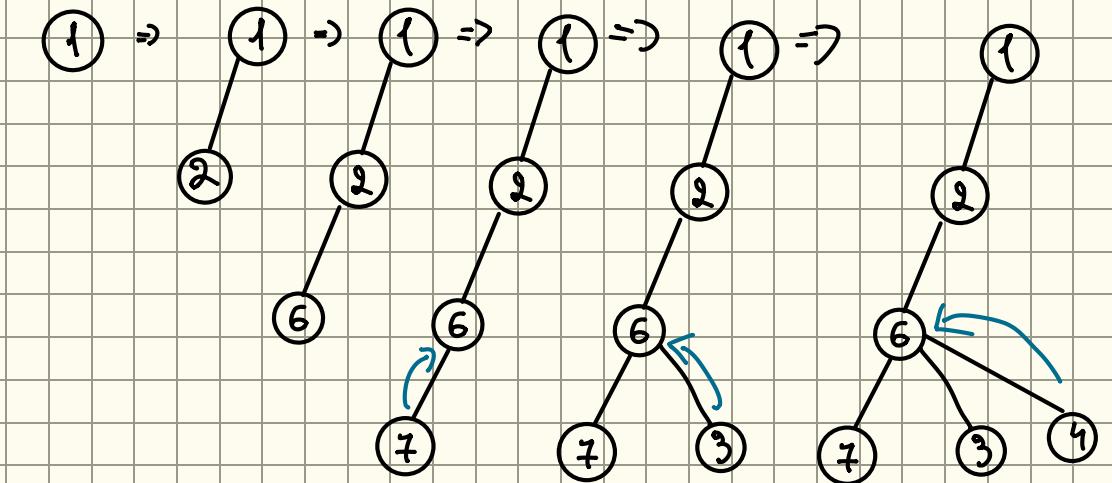


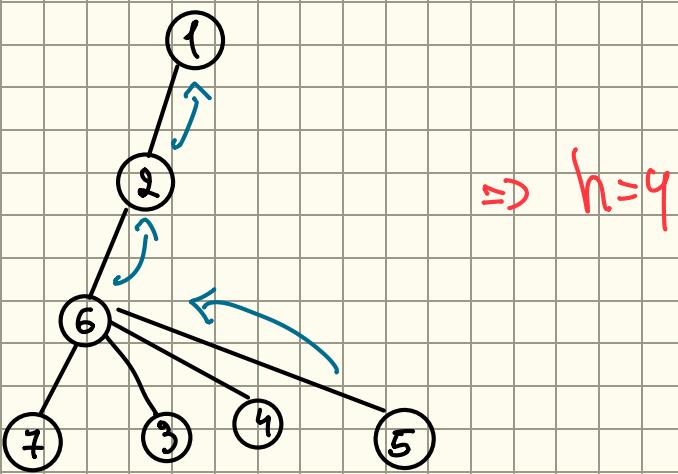
BFSC(1):



$$\Rightarrow h = 4$$

DFS(1):



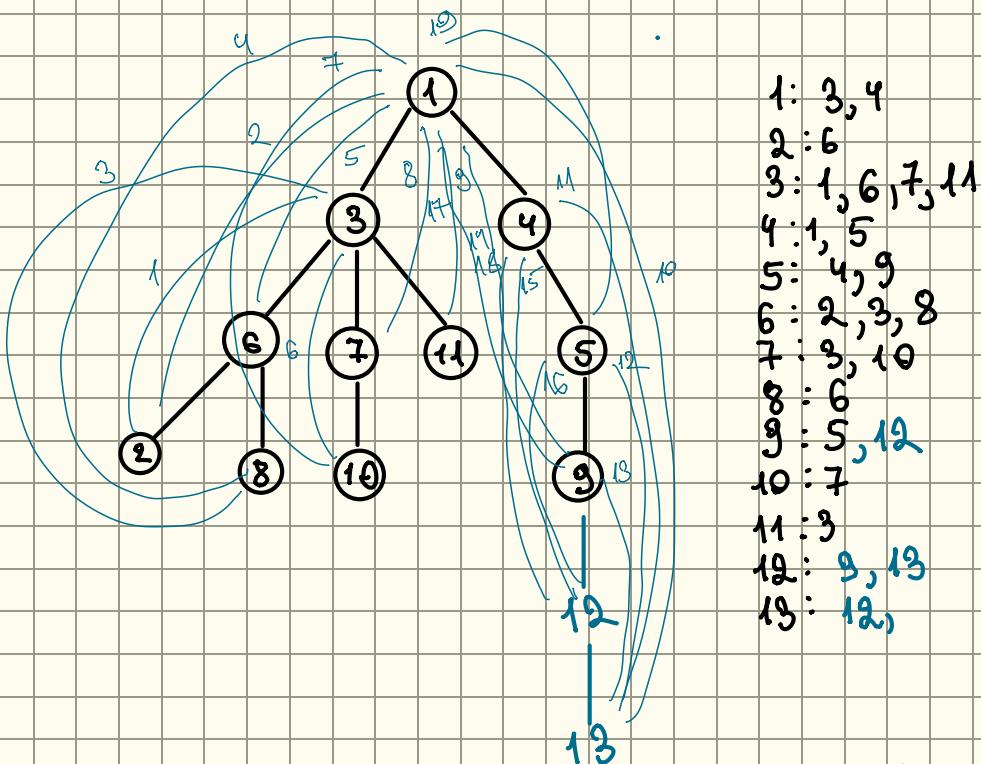


$$\Rightarrow h=4$$

b)  $G = (V; E)$ ;  $V = \{1, \dots, 19\}$ ; 12, 13 - lipses c

Ca arborele DFS să rămână la fel ar trebui să adăugăm din adâncime toate muchiile de întoarcere. Dacă folosim 12 și 13 pentru a mări adâncimea arborelui pe cat mai multe ramuri vom avea un nn mai mare

Adăugăm 12 și 13 în continuarea lui g.



În total sunt 19 muchii de întoarcere. Însamânnd cele 12 muchii din arbore  $\Rightarrow$  31 muchii max în graful original.

2.  $S_0(n) = \{1, 2, 3, \dots, n-1, n, n, n, n+2, \dots, 2n+1\}$  este secvența gradelor unui grafoorientat.

$1, 2, 3, \dots, n-1 \Rightarrow n-1$  noduri

$n, n, n \Rightarrow 3$  noduri

$n+2, \dots, 2n+1 \Rightarrow n$  noduri

2n+2 noduri (+)

Aplicăm Havel - Hakimi

$$S_0(n) = \{ \cancel{2n+1}, \underbrace{2n, 2n-1, \dots, n+2}_{n-1} \underbrace{n, n, n}_{3} \underbrace{n-1, \dots, 3, 2, 1}_{n-1} \} = 2n+1$$

$$S_0(n) = \{ 2n-1, 2n-2, \dots, n+1, n-1, n-1, n-2, 2, 1, \cancel{0} \}$$

$$= \{ 2(n-1)+1, 2(n-1), \dots, (n-1)+2, n-1, n-1, n-1, (n-1)-1, \cancel{0}, 1 \}$$

$$= S_0(n-1)$$

$P_n$ :  $S_0(n)$  - secvență grafică.

I Verificare:  $P_1: S_0(1) = \{ 1, 1, 1, 3 \}$

Aplicăm HH  $S_0(1) = \{ 3, 1, 1, 1 \}$

$S_0(1) = \{ 0, 0, 0, 0 \} \Rightarrow$  fără grafi.

II  $P_k: S_0(k)$

$P_k \rightarrow P_{k-1}$ .

$P_0, P_{k-1}$  adev.

$$S_0(k) = \{ 1, 2, 3, \dots, k-1, k, k, k, k+2, \dots, 2k+1 \}$$

$$S_0(k) = \{ \cancel{2k+1}, \underbrace{\dots, k+2}_{2k-k-1+1}, \underbrace{k, k, k}_{3}, \underbrace{k-1, \dots, 3, 2, 1}_{k-1} \} = 2k+1$$

(avem destul să scădem)

$$S_0(k) = \{ 2k-1, 2k-2, \dots, k+1, k-1, k-1, k-1, k-2, \dots, 2, 1, \cancel{0} \}$$

$$= \{ 2(k-1)+1, 2(k-1), (k-1)+1, k-1, k-1, k-1, (k-1)-1, \dots, 3, 2, 1 \}$$

$$= S_0(k-1).$$

$\Rightarrow$  Conform inducției matematice,  $\forall S_0(k)$ , după un pas de Havel - Hakimi este egal cu  $S_0(k-1)$ .

$$\left. \begin{aligned} & HH(S_0(k)) = S_0(k-1) \\ & S_0(1) \text{ este grafic} \end{aligned} \right\} \Rightarrow (S_0(n))_{n \geq 1} \text{ grafic } \forall n.$$

Unde  $HH(\lambda) =$  sortare +  
Scăzut primul din toate restul căte 1.

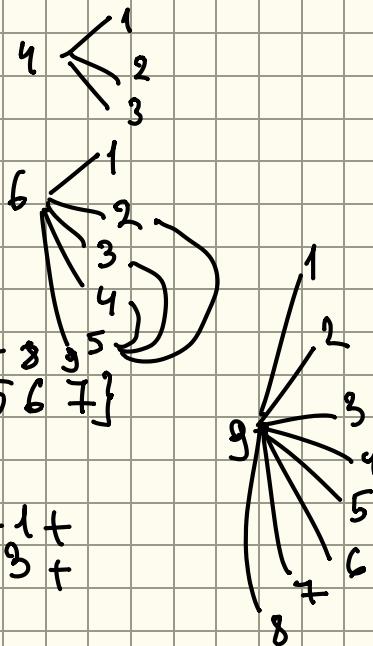
FAMILIE DE GRAFURI:  $2n+3$  noduri

$$S(n) = \{1, 2, 3, \dots, n-1, n, n, n, n+2, \dots, 2n+1\}$$

$$S_0(1) = \begin{matrix} 1 & 2 & 3 & 9 \\ 1 & 1 & 1 & 3 \end{matrix}$$

$$S_0(2) = \begin{matrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 1 & 2 & 2 & 2 & 4 & 5 \end{matrix}$$

$$S_0(3) = \begin{matrix} 1 & 2 & 3 & 9 & 5 & 6 & 7 & 3 \\ 1 & 2 & 3 & 3 & 3 & 4 & 5 & 6 \end{matrix}$$



$$A = \{1, 2, 3, \dots, n-1\}$$

$$\frac{n-1}{3} +$$

$$B = \{n, n, n, n\}$$

$$C = \{n+2, n+3, \dots, 2n+1\}$$

Fie  $C'$  initial un graf complet

$$\Rightarrow C' = \{n-1, n-1, \dots, n-1\}$$

$$C = \{n+2, n+3, \dots, 2n+1\}$$

(-)

$$C - C' = \left\{ \frac{n+3}{1}, \frac{n+4}{2}, \dots, \frac{2n+2}{n+1} \right\} = 2n+3 - n-3+1 = n+1$$

$$A = \{1, 2, \dots, n-1\}$$

Din  $A$   $n-1: n+4, n+5, \dots, 2n+2$  ( $2n+2 - n-4+1 = n-1$ )

$n-2: n+5, \dots, 2n+2$  ( $2n+2 - n-5+1 = n-2$ )

$$\Rightarrow C - C' - A = \{3, 3, \dots, 3\}$$

Avem legam  $\forall x \in B$  cu fiecare din  $C - C' - A$

$$\Rightarrow C - C' - A - B = \{00\dots 0\}$$

Deci familia  $\rightarrow$  Componentă completă cu elementele

$$C = \{n+2, n+3, \dots, 2n+1\} \text{ cu } i = \overline{n+3, 2n+2}$$

$\rightarrow$  Din componentă  $A = \{1, 2, \dots, n-1\}$

$i = \overline{1, n-1}$  legăm descrescător

fiecare  $x \in A$  cu toate din  $C$

iar fiecare grad din  $C$  mai are  
nevoie de 3.

$\rightarrow$  legăm fiecare  $x \in B$  cu el  $C$ .

$$\begin{aligned} \text{explicatie: } C &= \{n+3, n+4, \dots, 2n+2\} & = 2n+2-n-3+1=n \\ &= \{1, 2, \dots, n-1\} & = n-1-1+1=n-1. \end{aligned}$$

(legăm fiecare  $A[i]$  cu  $C[i+1] \dots C[2n+2]$ )

$$\Rightarrow C = \{n-1, n, n+1, \dots, 2n-2\} \text{ n-ei}$$

$$B = \{n, n, n\}$$

Legăm  $B$  cu toate  $C$

$$C = \{n+2, \dots, 2n+1\}.$$

3.  $G$ -graf conex neonorientat

$P$  și  $Q$  două lățuni elementare de lungime max.

Anătați că  $P$  și  $Q$  au cel puțin un nod comun.

$G$ -conex  $\Rightarrow$   $\exists$  un drum către oricare 2 noduri

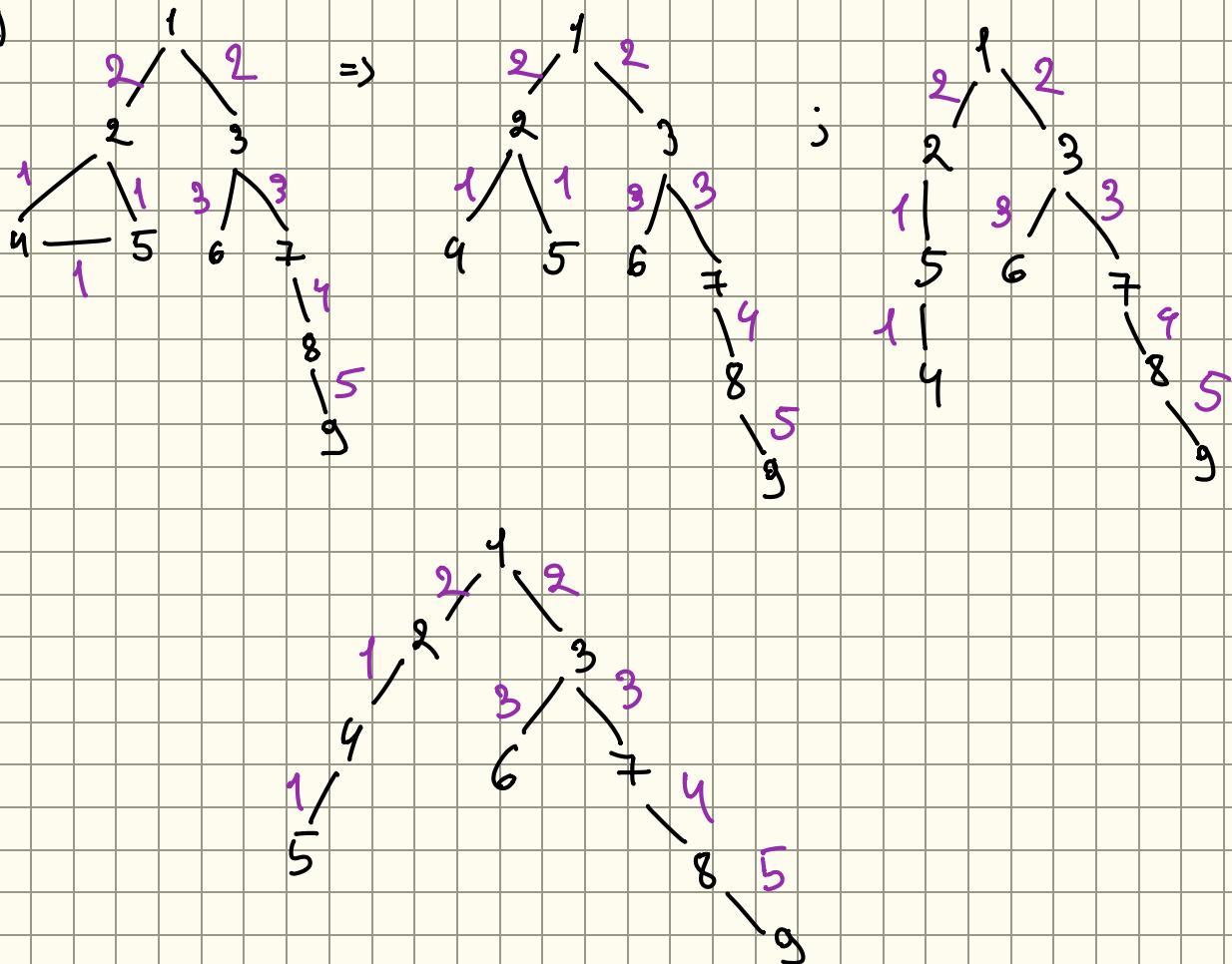
Presupunem că  $P$  și  $Q$  nu au niciun nod în comun.

Dar  $G$ -conex  $\Rightarrow$   $\exists$  un drum către oricare 2 noduri

$\Rightarrow$   $\exists$  un lanț  $O$  cu minim 2 noduri care leagă prima nod din  $P$  cu ultimul din  $Q$

$\Rightarrow$  Există lantul  $P - Q - Q$  cu nr mai mare de noduri decât  $P$  sau  $Q$ , (alt maxim)  $\Rightarrow$  CONTRADICTION

9. a)



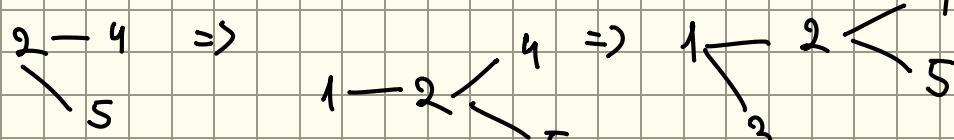
b) KRUSKAL: 1-2 (2)  
1-3 (2)  
2-4 (1)  
2-5 (1)  
6-5 (1)  
3-6 (3)  
3-7 (3)  
7-8 (4)  
8-9 (5)

le sortam

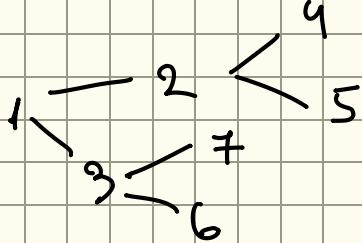
2-4 (1) V  
2-5 (1) V  
4-5 (1) X  
1-2 (2) V  
1-3 (2) V  
3-6 (3) V  
3-7 (3) V  
7-8 (4) V  
8-9 (5)

Adaugăm grafului căte o muchie fără se facem cicluri.

2-9  $\Rightarrow$  2-4  $\Rightarrow$



$\Rightarrow$  1-2-2-9-5  $\Rightarrow$



$\Rightarrow$

$$\begin{array}{c} 1 \quad 2 \\ | \quad | \\ 3 \quad 7-8 \\ | \quad | \\ 6 \end{array} \quad \Rightarrow$$

$$\begin{array}{c} 1 \quad 2 \\ | \quad | \\ 3 \quad 7-8-9 \\ | \quad | \\ 6 \end{array}$$

PRIM: I Start = 1

II     $1-2$

$1-2$  sau  $1-3$

III     $1-2$   
 $2-3$

$1-3; 2-4; 2-5$

IV     $1-2-4$   
 $2-3$

$2-4; 2-5; 3-6; 3-7$

V     $1-2-4$   
 $2-3$   
 $2-5$

$2-5; 4-5; 3-6; 3-7$

VI     $1-2-4$   
 $2-3$   
 $2-5$

$3-6; 3-7$

VII     $1-2-4$   
 $2-3$   
 $2-5$

$3-7$

VIII     $1-2-4$   
 $2-3$   
 $2-5$

$7-8$

$7-4$

IX     $1-2-4$   
 $2-3$   
 $2-5$

$8-9$

$7-8-5-9$

c)  $G$ -conex, neorientat ponderat cu ponderi pozitive

fie vecinii lui  $S: V = \{v_1, v_2, \dots, v_n\}$  fie  $M \subseteq V$  cu prop. că  $v \in M$  are costul minim din  $V$  (adică nodurile cu care care are  $S$  cost min; și sau mai multe)

Dijkstra va alege fix aceste muchii de cost minim

Algoritmul de APCM va alege nodurile de cost min care iau din  $S$

Cea mai ieftină muchie care este dintr-un nod va apărea în toate APCM

5.  $G = (V, E)$  neorientat ponderat

$S \subseteq V$

Presupunem că există mai multe APCM 1, APCM 2 ... DIFERITE

Fie  $S \subseteq V \Rightarrow$  tăietura  $(S, V-S)$  cu o unică muchie light notată „ $m$ ”

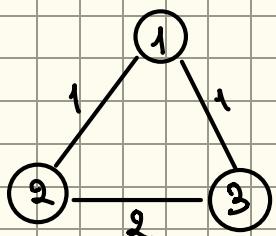
$\Rightarrow m \in \text{APCM} 1, \text{APCM} 2 \dots$  și este unică, căci este ponderea minimă unică, iar ca alta nu ar mai fi minim  $\Rightarrow$  nu este apcm.

Dacă parcurgem toate tăieturile  $(S, V-S)$  și alegem aceasta muchie ajungem să adăugăm aceeași muchie în toate APCM.

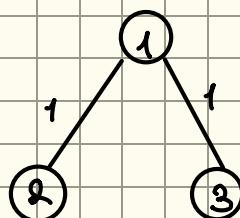
$\Rightarrow \text{APCM} 1 = \text{APCM} 2 = \dots \Rightarrow$  CONTRADIȚIE  $\Rightarrow$  un APCM unic pînă prop din ipoteză.

IMPLICAȚIA INVERSĂ:

Dacă apcm este unic  $\Rightarrow$  pt orice tăietură  $(S, V-S)$  există o singură muchie light.



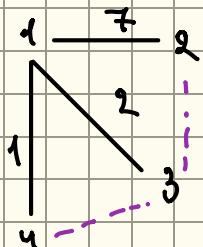
APCM  $\Rightarrow$



este UNIC.

Fie traiectoria  $(\{1\}; \{2, 3\})$ . Muchiile care o traversează sunt  $1-2(1)$ ;  $1-3(1)$ . Ambele sunt light, deci nu există o unică muchie light pentru orice traiectorie  $(S, V-S)$  dacă APCM este unic.

6.



Calculăm un APCM cu Kruskal / PRIM din inputul dat.

Dacă muchie nouă adăugată într-un APCM va crea un ciclu, aşa că putem calcula maximul din acest ciclu, iar ponderea noii muchii va fi acest maxim minus o unitate, pentru a fi aleasă în locul celei maxime la calculul unui viitor APCM, dar maximizând costul APCM.

Pentru a calcula acest maxim, vom face DFS din prima extremitate a muchiei, până la două, iar această parcurgere smușează unicul ciclu realizat prin adăugarea unei muchii în APCM.

Complexitate: I CITIRE și STOCARE

Timp:  $O(M)$   
Memorie:  $O(M)$

## II KRUSKAL

a) Sortare Muchii: Quick Sort.

Timp:  $O(M \log M)$   
Memorie:  $O(1)$

b) Reuniune pt APCM.

Timp:  $O(N)$   
Memorie:  $O(N)$

----- (+)

Timp:  $O(M \log M)$   
Memorie:  $O(M+N)$

### III DFS-uri pe APGM

Viz. assign( $n+1, 0$ )  $\Rightarrow$  Timp:  $O(n)$   
Mem:  $O(1)$

Un DFS pe APGM: Timp:  $O(n)$   
Mem:  $O(1)$

$Q$  DFS-uri: Timp:  $O(Q \cdot (N+n)) = O(Q \cdot N)$   
MEM:  $O(N)$

$\Rightarrow$  Timp total:  $O(M \log N + Q \cdot N)$

$$\begin{array}{l} M \in (1, 10^5) \\ Q \in (1, 10^3) \end{array} \Rightarrow O(M \log N)$$

Memoriile totale:  $O(M+N)$

PSEUDOCOD:

```

    citire N, M, Q
    pentru i <= 1, M, +1 execută
        citire x, y, cost
        adaugă în lista(x, y, cost)
    sort(lista, > cost)
  
```

reuniune

```

    pentru i <= 1, Q, +1 execută
        citire a, b
        viz. assign(n+1, 0);
        max = DFS(A, B, 0)
        afis max - 1
  
```

CORECTITUDINE: De unde stim dacă muchia adăugată apare în toate APGM dacă Kruskal returnează unul singur.

În drumul dintr-un APGM între două noduri  $u, v$ , muchia maximă apare în toate APGM, căci dacă ar fi diferență, nu ar mai fi toate APGM-uri valide.  $\Rightarrow$  Nu contează din ce APGM luăm muchia maximă  $\Rightarrow$  Il putem folosi pe cel dat de Kruskal

1) Reverse - Delete  $\rightarrow$  tPM

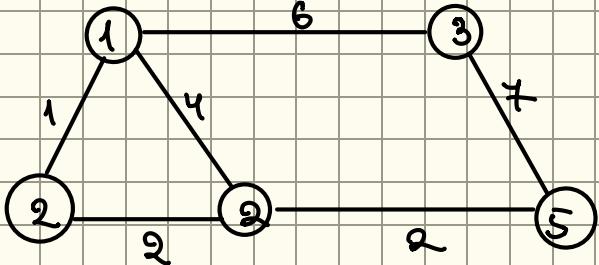
$V_1$   
 $V_2$

$T \subset G$

cât timp  $T$  mai conține cicluri  
alege  $C$  un "ciclu" elem din  $T$   
care e o mulțime de cost  
maxim în  $C$

$T \leftarrow T - e$

scrie  $T$



$$C = [1, 4, 2, 6] \rightarrow \text{elim } (1, 2)$$

$$C = [1, 4, 2, 5, 3, 1] \rightarrow \text{elim } (2, 5)$$

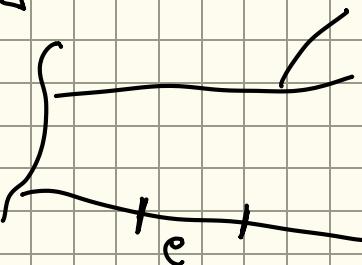
CORECTITUDINE: Fie  $e$  o mulțime de cost maxim din  $C$  astfel încât  $\exists$  APCM care nu conține  $e$   
(la fiecare pas  $\exists$  APCM  $\subseteq T$ )

Dem Fie  $T$  APCM

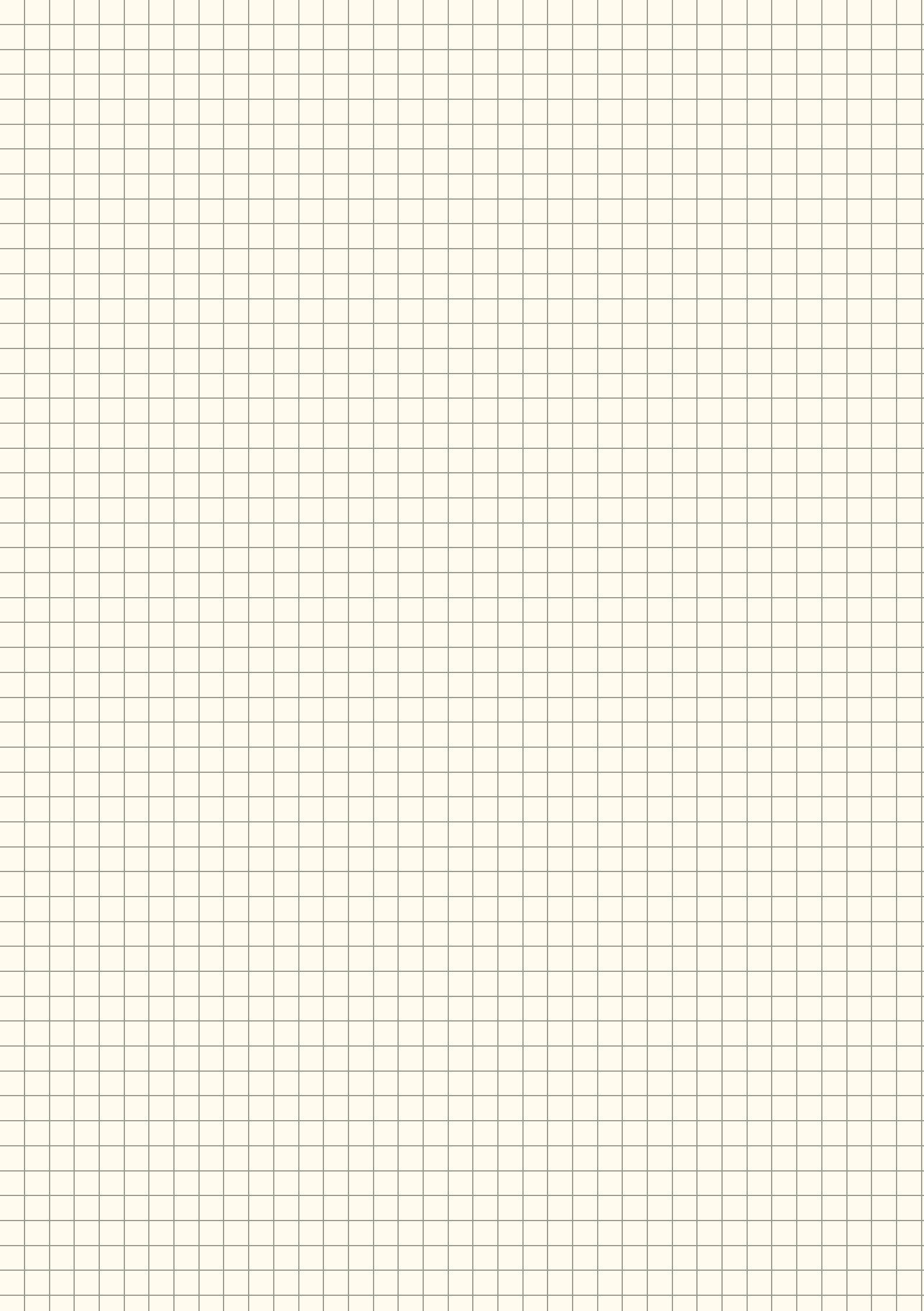
$e \notin T \rightarrow \exists$

$e \in T$

Const  $T' = T - e$



..



Numărul de dr min  
modul alg D.  $nr[u] = nr$  de dr de la s la u de cost din

$$nr[u] = 0 \quad \forall u$$

$$nr[D] = 1$$

for  $uv \in E$   
 if  $d[v] = -d[u] + w(uv)$   
 :  $nr[r] = nr[v] + nr[u]$  } mai multe dr de cost  $d[v]$   
 else  
 if  $(d[v] > d[u] + w(uv))$   
 $d[v] = d[u] + w(uv)$   
 $nr[r] = nr[u]$

b) a) D-max de dist

Cum verificăm dacă un vf k se află pe un dr min de la i la j.

$$d[i][k] + d[k][j] = d[i][j]$$

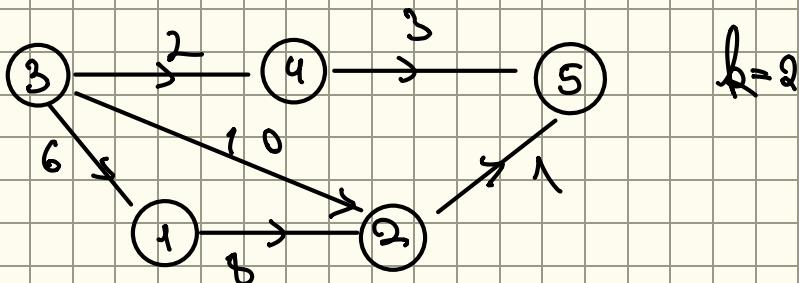
b) n orașe  $1, 2, \dots, n$

drumuri unidirectionale care au asociată o taxă.

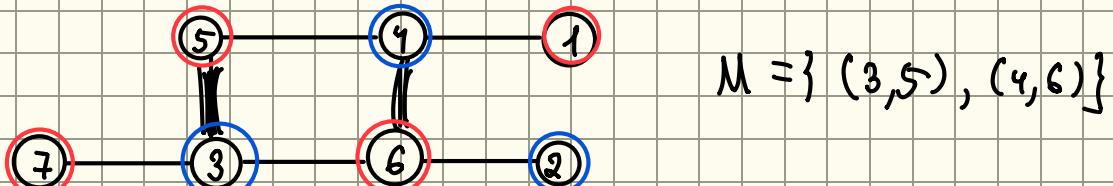
→ Ce alg. folositi pt a determina un traseu de la i la j.  
 pt care trb plătită taxa min.

→ Din cauza unei epidemii, orașele de la 1-k au devenit centre de testare. Si eliminate taxele de deplasare dintre ele.

Cum ați fol. alg pe care îl cun. pt care perechi  $i, j$  determină un traseu cu taxa minimă de la i la j care conține cel puțin un centru de testare.



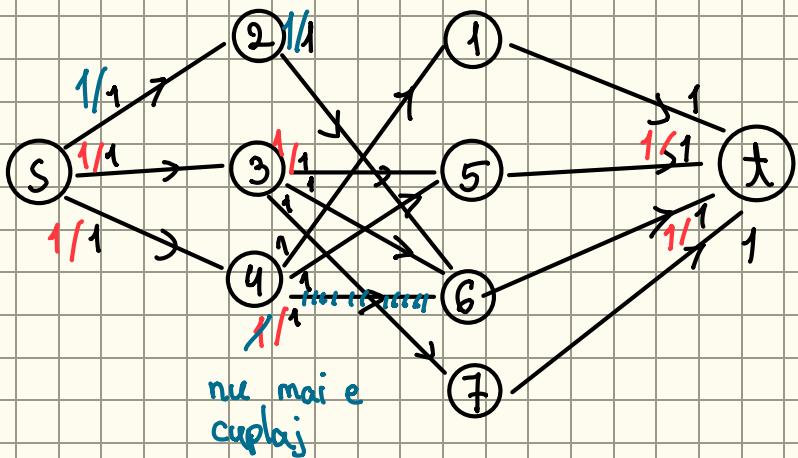
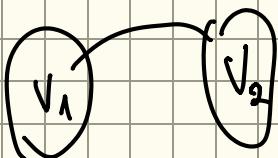
2)  $G, M$  cuplă și



a) Construiți rețeaua asociată cu fluxul coresp. cuplajului  $M$ .

b) Constr. fluxul max în rețea și cuplajul max coresp.

a)  $G$  bipartit



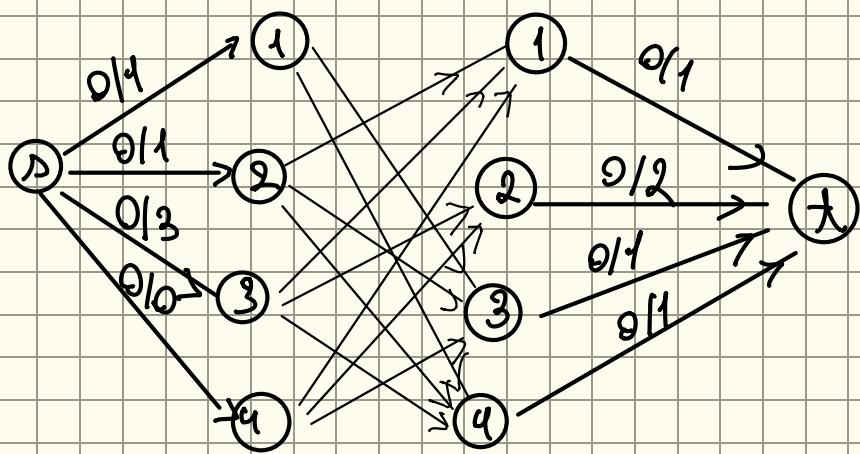
cuplaj max  $M^* = \{(2,6); (4,1); (3,5)\}$

3)  $\Delta_0^t = \{1, 1, 3, 0\}$  - ieșire

$\Delta_0^- = \{1, 2, 1, 1\}$  - intrare

Folosin alg. FF. verifică dacă  $\exists$   $G$  graf orientat cu secvențele gradelelor date care nu conține arcul  $(1,2)$

Rețea:



anțele  $i, j$  care pot fi în  $G$

$\exists G \Leftarrow$

$\text{val } f = \text{nr anțe} = \text{suma gr în } \Leftarrow \text{Toate anțele care}\}$   
 $\text{ies din } s \text{ sunt pline.}$

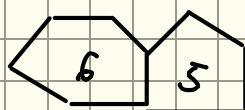
- 4)  $M = (V, E, F)$  conex cu toate vf de gr 3 cu fețe doar de grad 5 sau 6  
 Arătăți că  $M$  are exact 42 fețe de grad 5

$$\text{Dem: } n - m + |F| = 2$$

$$\sum_{\text{gef}} \deg(f) = 2m$$

$$\sum_{v \in V} \deg(v) = 2m$$

} Grafuri Planare



$f_5$  - nr fețe gr 5

$f_6$  - nr fețe gr 6

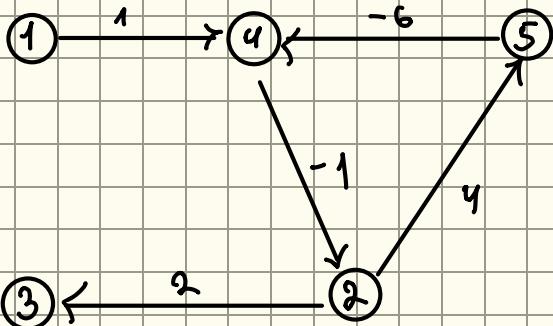
$$n - m + f_5 + f_6 = 2$$

$$5 \cdot f_5 + 6 \cdot f_6 = 2m$$

$$3n = 2m$$

$$\left\{ \begin{array}{l} f_5 = 2 - f_5 + m - n \\ 5f_5 + 12 - 6f_5 + 6m - 6n = 2m \\ 3n = 2m \\ \left\{ \begin{array}{l} 12 - f_5 + 4m - 6n = 0 \\ 3n = 2m \end{array} \right. \\ \left\{ \begin{array}{l} 12 - f_5 + 2 - 3n - 6n = 0 \\ \downarrow \\ f_5 = 12. \end{array} \right. \end{array} \right.$$

1.  $G = (V, E, w)$



$$S = \{1\}$$

$$E = \{(1, 4), (2, 3), (4, 2), (2, 5), (5, 4)\}$$

$$[0, -\frac{1}{3}, -4, -1, \frac{5}{3}]$$

pentru fiecare  $u \in V$  execută  
 $d[u] = \text{infinit}$

$$d[1] = 0$$

pentru  $i=1, |V|-1$  execută

- pentru fiecare  $u, v \in E$  execută

$$\text{daca } d[u] + w(u, v) < d[v] \text{ atunci}$$

$$d[v] = d[u] + w(u, v)$$

pentru fiecare  $u, v \in E$  execută

$$\text{daca } d[u] + w(u, v) < d[v]$$

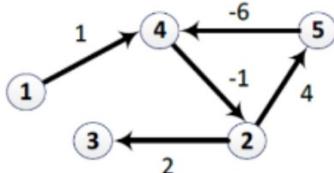
break

afis "ciclu negativ"

- Explicatie: Dacă după cele  $|V|-1$  iteratii s-ar mai face o actualizare a etichetelor de distanțe minime înseamnă că avem un ciclu negativ.

Într-un graf, orice lant elementar are maxim  $|V|-1$ , muchii. Dacă într-un drum minim între două noduri găsim o  $|V|$ -a muchie, înseamnă că suntem într-un ciclu negativ, din acest motiv, la finalul algoritmului ne asigură detectarea acestor cicluri.

- Desen:



$d : [ \begin{smallmatrix} 1 & 2 & 3 & 4 & 5 \\ \infty & \infty & \infty & \infty & \infty \end{smallmatrix} ]$  : pentru fiecare  $v \in V$  execută  
 $\sqsubseteq_v$   $d[v] = \text{infinit}$

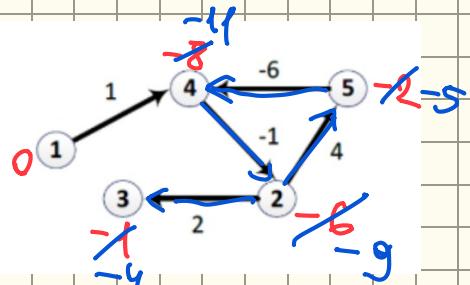
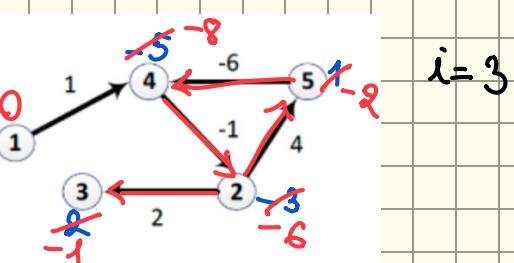
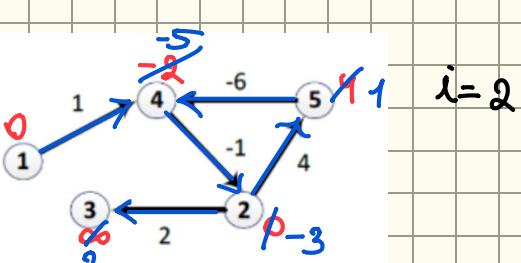
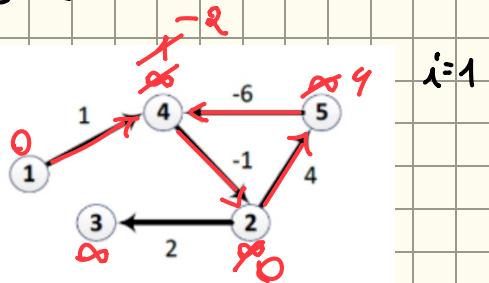
$d : [ \begin{smallmatrix} 1 & 2 & 3 & 4 & 5 \\ 0 & \infty & \infty & \infty & \infty \end{smallmatrix} ]$  :  $d[S] = 0$

$0 + 1 < \infty \vee$   
 $\infty + 2 < \infty \times$   
 $1 - 1 < \infty \checkmark$   
 $0 + 4 < \infty$   
 $4 - 6 < 1 \vee$

$0 + 1 < -1 \times$   
 $0 + 2 < \infty \checkmark$   
 $-2 - 1 < -2 \checkmark$   
 $-3 + 4 < 4$   
 $1 - 6 < 1$

$0 + 1 < -5 \times$   
 $-3 + 2 < 2$

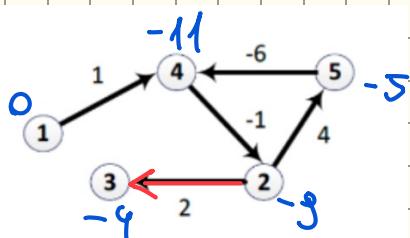
$d : [ \begin{smallmatrix} 1 & 2 & 3 & 4 & 5 \\ 0 & \infty & \infty & \infty & \infty \end{smallmatrix} ]$  :



Pas de verificare:

$d : [ \begin{smallmatrix} 1 & 2 & 3 & 4 & 5 \\ 0 & -9 & -11 & -11 & -5 \end{smallmatrix} ]$  :

$\downarrow$   
**STOP**



**CICLU NEGATIV.**

2. a) fie  $g$  - noul flux după revizuire

Pentru ca  $g$  să fie un flux valid demonstrăm:

$$1. 0 \leq g(e) \leq c(e); \forall e \in E(G) \text{ cond. de mărginime}$$

$$2. \sum_{uv \in E} g(uv) = \sum_{vu \in E} g(vu) \text{ cond de conservare}$$

Fie  $L$  un lant f-nesaturat

capacitatea residuală a lui  $L$  este  $\pi(L) = \min(c(uv) - f(uv); uv \in L) > 0$

Construim noul flux  $g$

$$g(uv) = f(uv) + \pi(L), \forall uv \in L$$

Verificăm dacă respectă condițiile

$$1. 0 \leq f(uv) + \pi(L) \leq c(uv)$$

$$\pi(L) = \min(c(uv) - f(uv); uv \in L)$$

$$\Rightarrow 0 \leq \pi(L) \leq c(uv) - f(uv)$$

$$0 \leq f(uv) + \pi(L) \leq c(uv)$$

$$0 \leq g(uv) \leq c(uv) \text{ Adevarat.}$$

2.  $g$ : flux valid

$$\Rightarrow \sum_{uv \in E} f(uv) = \sum_{vu \in E} f(vu) + \pi(L), \forall uv \in L$$

$$\sum_{uv \in E} f(uv) + \pi(L) = \sum_{vu \in E} f(vu) + \pi(L) \quad \forall uv \in L$$

$$\sum_{uv \in E} g(uv) = \sum_{vu \in E} g(vu) \text{ Adevarat}$$

$\Rightarrow$  Prin operația de revizuire a unui flux de -a lungul unui s-t lant f-nesaturat se obține tot un flux.

b) fie  $g$  un flux

$$1. EK_1: g(uv) = 0, \forall uv \in E \text{ Adevarat}$$

$$2. \bar{EK}_k: g(uv) - \text{par } \forall uv \in E$$

$EK_k \rightarrow EK_{k+1}$ : fie  $L$  un drum de creștere

$$\pi(L) = \min \{ c(uv) - f(uv); (u, v) \in L \}$$

$c(uv)$  par

Presupunem EK este adevărat  $\Rightarrow f(uv)$  par  $\forall uv \in L$   
adăvărat.

$\Rightarrow \pi(L)$  par

$\Rightarrow f(uv) + \pi(L)$  par  $\forall uv \in L$

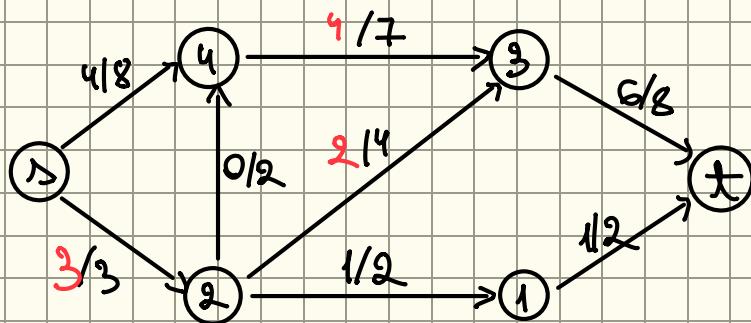
fie  $g$  nodul flux

$$g(uv) = f(uv) + \pi(L) \text{ par } \forall uv \in L$$

$$g(uv) = f(uv) \text{ par } \forall uv \in L \setminus E$$

$\Rightarrow$  Prin inducție matematică și algoritm EK  $\Rightarrow$  EKn adevărat.

$\Rightarrow \exists$  un flux maxim cu proprietatea că pentru orice arc  $e$ ,  $f(e)$  este par,



pe nodul 4:  $4/8(5) + 0/2(2)$

$\Rightarrow$  Iesed din 4 : 4

Pe nodul 3 :  $4/7(9) + x/4(2) = 6/7$

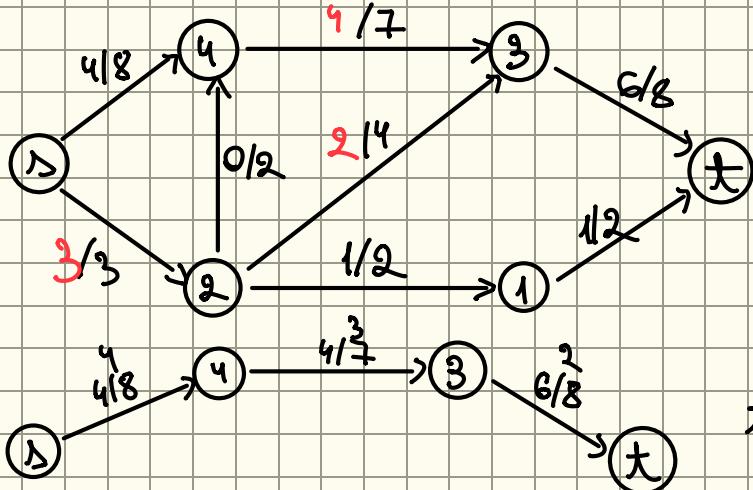
$$\Rightarrow x/4(2) = 2/1/...$$

$$x = 2$$

Pe nodul 2 :  $x/3(5) = 2/4 + 1/2$

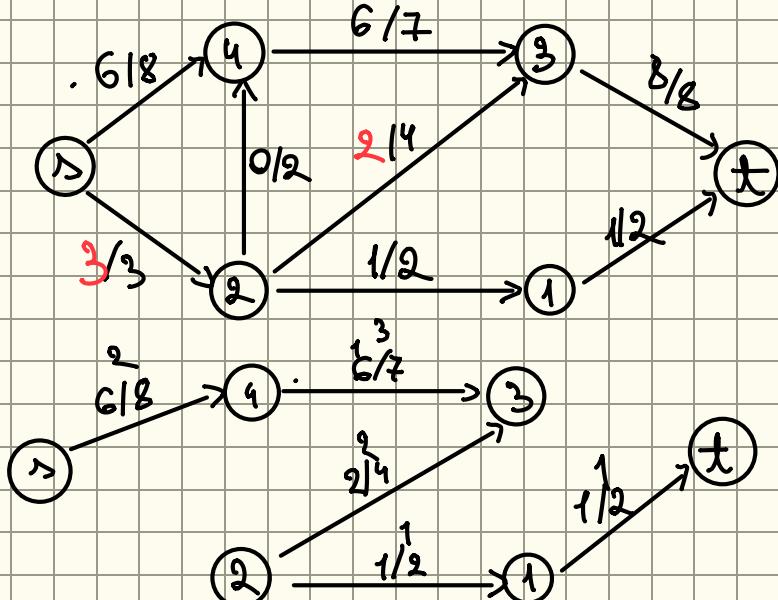
$$\Rightarrow 3/3(5)$$

EK :



$$\pi(L) = 2$$

EK2:

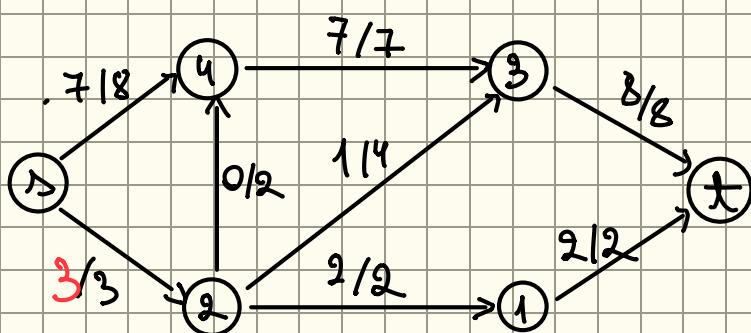


$$\pi(L) = 1$$

Scădem din 2,3 pe  $\pi(L)$

$$2/4 - 1 = 1/4$$

EK3:



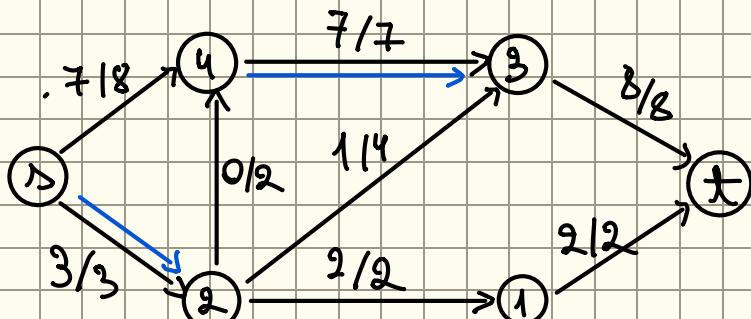
d) Considerăm  $S =$  "multimea „nodurilor accesibile plecând din sursă"

$T =$  "nodurile la care nu putem ajunge din sursă"

$\Rightarrow$  Fie  $K = (S, T)$  o căietură  
Unde:  $\forall y \in E \quad \exists x; y \in Y$   
 $y \notin E \quad \forall x; y \notin Y$

$\Rightarrow$  anele căieturii vor fi multimea acestor muchii ce leagă  $S$  și  $T$ .

în cazul nostru:



$S = \{1, 4\}$ ; 3 este inaccesibil are 3/3

$T = \{1, 2, 3, 4\}$ ;

Ancele tăieturi sunt:  $(4, 3)$  și  $(1, 2)$

Capacitatea este  $\sum_{\substack{x \in S \\ y \in T \\ (x,y) \in E}} c(x,y) = c(4,3) + c(1,2) = 7 + 3 = 10$ .

3. Descriere: Dacă graful este aciclic, putem realiza o sortare topologică a nodurilor. Aceasta ne va returna o listă a futuror nodurilor care trebuie să ne apără parcursul înainte de a ajunge la vârful s. Astfel evităm algoritmi mai ineficienți ca complexitate precum Dijkstra sau Bellman-Ford.

Dacă în graf există muchia  $A \rightarrow B$ , atunci A va fi obligatoriu înaintea lui B în lista returnată de top. sort.

În continuare ne vom folosi de 2 vectori:

$d[u] =$  "distanța min de la sursă la u"

$nr[u] =$  nr de drumuri minime care duc la u"

Parcurgem lista returnată de top-sort, și pentru fiecare  $u$ : lista, verificăm toți vecinii săi  $v$  unde există muchie  $uv$  cu costul  $c$  și relaxăm muchile:

- Dacă găsim un drum mai scurt ( $d[v] + c < d[v]$ )
  1. Actualizăm distanța minimă  $d[v] = d[u] + c$
  2. Resetăm numărul de drumuri  $nr[v] = nr[u]$ .
- Dacă găsim un drum egal ( $d[u] + c == d[v]$ )
  1. Adunăm drumurile  $nr[v] += nr[u]$

În final, rezultatul se va afăla în  $nr[>]$

Complexitate: Algoritmul este eficient datorită naturii grafului (aciclic) pentru că putem folosi top sort.

Timp: Sortare Topologică:  $O(N+E)$  (DFS)  
Parcugere Listă + Relaxare:  $O(N+E)$

Total:  $O(N+E)$

Spațiu: Vectorii d, nr și lista top sort:  $O(N)$

Graf în liste de adiacență:  $O(N+E)$

Total:  $O(N+E)$

PSEUDOCOD: listTopSort = sortTopologică(G)

pentru  $\forall$  nod  $\in G$

|     d[nod] ← infinit

|     nr[nod] ← 0

d[start] ← 0

nr[start] ← 1

pentru  $\forall$  nod  $\in$  listTopSort

|     dacă  $d[nod] = \text{infinit}$

|         continuă

|     pentru  $\forall$  vec  $\in$  vecini(nod)

|         dacă  $d[nod] + c < d[vec]$

|              $d[vec] \leftarrow d[nod] + c$

|             nr[vec] ← nr[nod]

|         dacă  $d[nod] + c = d[vec]$

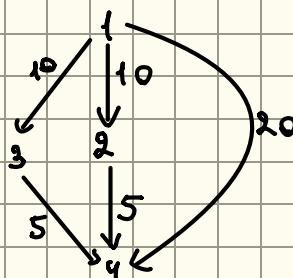
|             nr[vec] ← nr[vec] + nr[nod]

afisează nr[1]

Ilustrare pe Exemplu:

Fie Graful

nod start: 1.



listTopSort: [1, 2, 3, 4]

d : [ $\begin{smallmatrix} 0 & 10 & 3 & 4 \\ 10 & 0 & 10 & 10 \end{smallmatrix}$ ]  
15

nr: [ $\begin{smallmatrix} 1 & 2 & 3 & 4 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 2 \end{smallmatrix}$ ]

nod 1 : vec 2 :  $0 + 10 < \text{inf}$   
 vec 3 :  $0 + 10 < \text{inf}$   
 vec 4 :  $0 + 20 < \text{inf}$   $\Rightarrow$  distanță min: 15 ( $d[4]$ )  
 nod 2: vec 4:  $10 + 5 < 20$  nr de drumuri: 2 ( $nr[4]$ )  
 nod 4: vec 4:  $10 + 5 = 15$

4.  $G = (V, E)$   
 $n = |V|$   
 $m = |E|$

**Descriere:** Pentru fiecare muchie  $A \rightarrow B$  putem considera că aceasta are costul 0 (nicio operatie rev). Dacă adăugăm muchia complementară  $B \rightarrow A$  cu costul 1 (o operatie rev) și rulăm un algoritm pentru drumul minim dintre două noduri (Bellman-Ford sau Dijkstra) vom afla drumul minim dintre cele două noduri  $x, y$ .

Stim că greutățile  $0, 1 > 0$ , deci Dijkstra este mai bun decât BF.

De asemenea, stim că există doar două costuri, deci putem să ne nodurile într-un degeu sortate astfel: nodurile de pe care am venit cu cost 0 în față, cele cu cost 1 în spate. Astfel evităm complexitatea  $O(\log n)$  pe care o are coada de priorități doar ca să îmi returneze „cel mai mic nod”. Practic, ne gestionăm singuri coada de priorități cu operații realizate în  $O(1)$ , în loc de  $O(\log n)$ .

**Complexitate:** Spatiu: Liste de adiacență pentru graf:  $O(N+M)$

Vector de distanțe:  $O(N)$

Degeue:  $O(N)$

Total:  $O(N+M)$

Timp: Construire Graf:  $O(M)$

Initializare Vect. dist:  $O(N)$

Extragere Toate noduri din degeue:  $O(N)$

După extragere, procesăm toți vecinii:  $O(2M) = O(M)$

Total:  $O(N+M)$

PSEUDOCOD: pentru  $i=1, i \leq m, i=i+1$  execută

citeste  $u, v$   
 $G[u].\text{adaug}(v, 0)$   
 $G[v].\text{adaug}(u, 1)$

pentru  $i=1, i \leq n, i=i+1$

$\text{dist}[i] = \infty$

$\text{dist}[x] = 0$

Deque  $Q$   
 $Q.\text{adauga\_fată}(x)$

căt timp  $Q$  nu este vida execută

$u \leftarrow Q.\text{fată}$   
dacă  $u \neq v$  atunci  
afis  $\text{dist}[u]$   
stop

pentru  $v \in G[u]$  execută

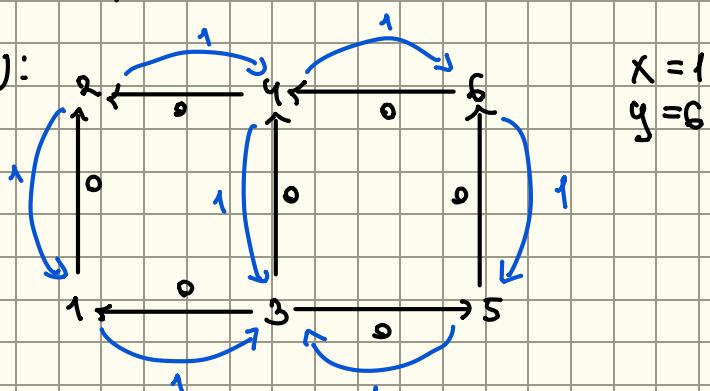
daca  $\text{dist}[u] + \text{cost} < \text{dist}[v]$  atunci  
 $\text{dist}[v] \leftarrow \text{dist}[u] + \text{cost}$

daca  $\text{cost} = 0$  atunci  
 $Q.\text{adauga\_fată}(v)$   
altfel  
 $Q.\text{adauga\_spate}(v)$

daca  $\text{dist}[v] = \infty$

afis „nu se poate”  
ele afis  $\text{dist}[v]$

EXEMPLU:



	1	2	3	4	5	6
dist [	0	∞	∞	∞	∞	∞
	0	1	1	1	1	1

de que:

6 ~~5~~ ~~2~~ ~~1~~ ~~8~~ 4

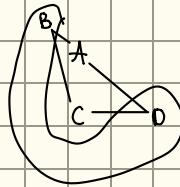
↓  
Stop

afis dist[6]=1.

5. a)  $m \leq 2n - 4$

$G$  graf planar  $\Rightarrow n - m + f = 2$

$G$  graf bipartit  $\Rightarrow$  nu contine ciclu nici impar



$\Rightarrow$  cel mai mic ciclu are cel putin lungimea 4

$\Rightarrow$  fiecare fată are cel putin 4 muchii

$\Rightarrow \sum_{\text{fete}} \text{latuni (fata)} \geq 4 \text{ fete}$

Fiecare muchie aparține în cel mult două fete,

$$\begin{aligned} \Rightarrow 4f \leq 2m &\Rightarrow 2f \leq m \\ n - m + f = 2 & \end{aligned} \quad \left. \begin{array}{l} \Rightarrow n - m + \frac{m}{2} \geq 2 \\ n - \frac{m}{2} \geq 2 \\ 2n - m \geq 4 \end{array} \right.$$

$2n - 4 \geq m$

$m \leq 2n - 4$

b)  $\exists x \in V$  cu  $d(x) \leq 3$

Pp că  $\forall x \in V$  are  $d(x) \geq 4$

$\sum_{x \in V} d(x) = 2m$

Dacă  $\forall x \in V$  are  $d(x) \geq 4$

$\sum_{x \in V} d(x) \geq 4n$

$$\Rightarrow 2m > 4n$$

$$m > 2n$$

dar la a) am demăs că  $m \leq 2n - 4$  CONTRADICȚIE!

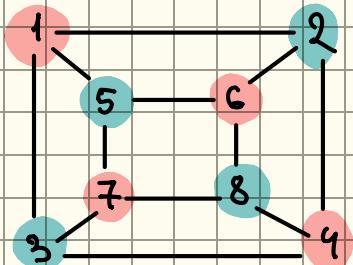
$\Rightarrow$  Pp făcută este falsă, deci  $\exists x \in V$  cu  $d(x) \leq 3$ .

c)

$$n=8$$

$$m=2n-4 = 2 \cdot 8 - 4 = 12 \text{ muchii}$$

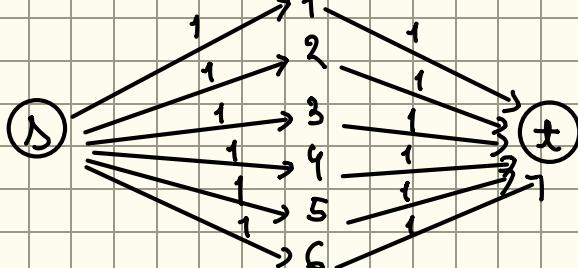
exemplu?



este graf planar: nu se intersectează muchii

este bipartit cu partitiile:  $\{1, 4, 6, 7\}$  și  $\{2, 3, 5, 8\}$

6.

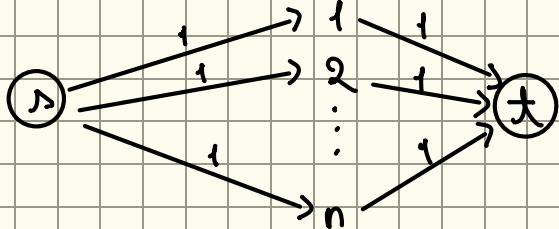


$$\text{nr de muchii} = 2n$$

$$m = 2n \Rightarrow n = \frac{1}{2} m$$

Algoritmul găsește toate drumurile s-i-t și le sătuează.  
 $i = 1, n$  și le sătuează.

În acest sens, EK face  $n$  etape  $\Leftrightarrow \frac{1}{2} m$  etape  $\Leftrightarrow \frac{1}{2} |E|$  etape



Fiecare etapă face un BFS cu  $O(|E|)$

În total EK face  $O(\frac{1}{2}|E|^2)$  pași  $\in \Sigma(|E|^2)$

7.  $C : E \rightarrow \{\text{roșu, verde, albastru}\}$

graf neorientat conex ponderat.  $G = (V, E, w)$

lant s-t cu cost minim de culori

Descriere: Problema determină un drum de cost minim într-un graf în care costul de parcurgere al unei muchii depinde de muchia anterioară (costul arc). Dijkstra nu reține parcurgerea, ci doar costul minim la un moment dat.

O soluție este să expandăm fiecare nod după cum

un mecanism:

nod: ( $u$ , culoare)

$u = "nodul"$

culoare = "culoarea lui" {rosu, albastru, verde} cu care  
am ajuns în nodul respectiv"

Apoi aplicăm Dijkstra pe acest nou graf.

Într-un graf normal este de ajuns să știm nodul curent și  
taxa, însă aici nu este suficient pentru cănăstii dacă să  
plateli taxa  $w_{v,u}$  sau nu.

Triplând nodurile și asociindu-le culorile, algoritmul stie dacă  
trebuie să platească taxa sau nu. De exemplu, când  
ajunge în nodul ( $u$ , rosu); stie că ultima muchie  
a fost rosie.

Pentru acest nou graf mențin aplicații Dijkstra pentru că  
decizia luării muchiei poate fi făcută doar cu informații  
de la pasul în care ne aflăm ((nod, culoare) + muchie  
următoare).

Complexitate:  
 $n = \text{nr de noduri}$   
 $m = \text{nr de muchii}$   
3 culori

$$\Rightarrow 3n = \text{nr de noduri în graful nou}$$
$$3m = \text{nr de muchii în graful nou}$$

Dijkstra pe graful nou:  $O(3m \log(3n))$   
:  $O(m \log n)$

Spătiu:  
Matrice distanțe:  $d[n][3] = O(n)$   
Liste adiacență:  $O(m)$   
Min-heap:  $O(m)$   
 $\Rightarrow O(m+m)$

Pseudocod:

pentru  $\forall u \in V$   
pentru  $\forall c \in \text{culoare}$   
d[u][c] = infinit

Pentru  $\forall$  vecin ( $s$ )

Pentru  $\forall$  culoare

$$d[\text{vecin}(s)][\text{culoare}] = \text{cost\_munchie}$$

Q.push(cost, vecin(s), culoare)

Cat timp Q nu este vida

(cost, nod, culoare)  $\leftarrow$  Q.pop

daca cost > d[nod][culoare]

continua

Pentru  $\forall$  vecin al lui nod

$$\text{penalizare} = 0$$

daca culoare != culoare\_vecin

$$\text{penalizare} \leftarrow \text{WC}$$

$$\text{cost\_nou} \leftarrow \text{cost} + \text{cost\_vecin} + \text{penalizare}$$

daca cost\_nou < d[vecin][culoare\_vecin]

$$d[\text{vecin}][\text{culoare\_vecin}] \leftarrow \text{cost\_nou}$$

Q.push(cost\_nou, vecin, culoare\_vecin)

daca  $\min(d[t][1], d[t][2], d[t][3]) = \text{infinit}$ .  
afis "nu se poate".

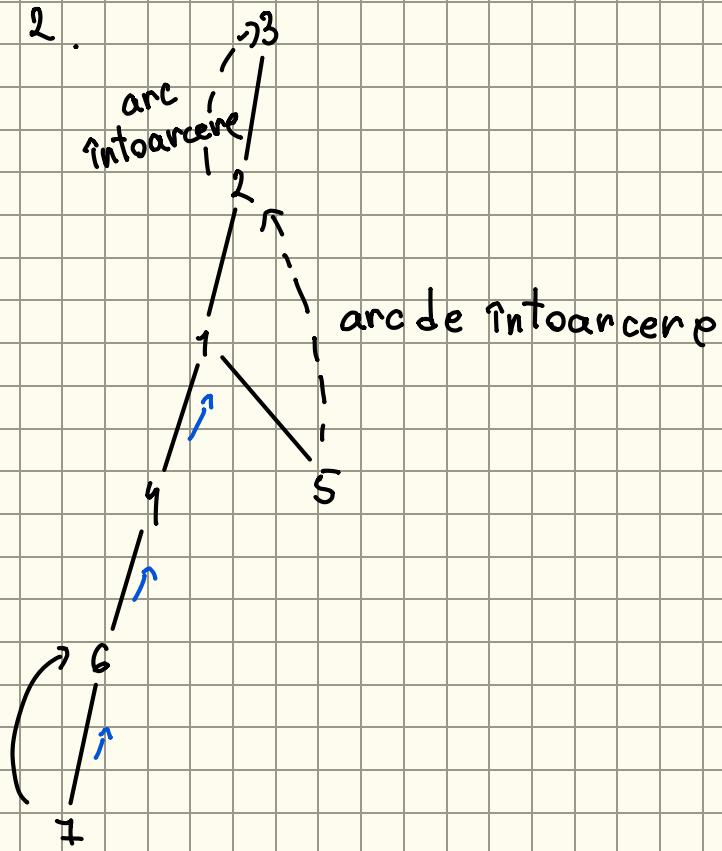
else

afis  $\min(d[t][1], d[t][2], d[t][3])$

8 ianuarie | Seminar 7

1.

2.

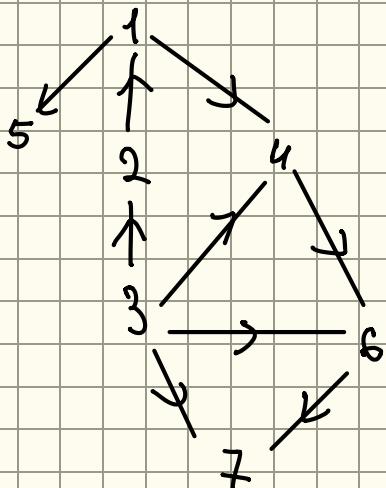


3.  $\exists$  sort top  $\Leftarrow \not\exists$  m intoarcere

(2, 3)

(7, 6)

(5, 2)



A<sub>1</sub>: DFS pus în stivă

A<sub>2</sub>: BFS pornind cu n�degini = 0

Sort top : 3 2 , 1, 5, 4, 6, 7

4. gr in = gr out  $\forall v$  cu 2 except i i

$$gr\ in = gr\ out + 1$$

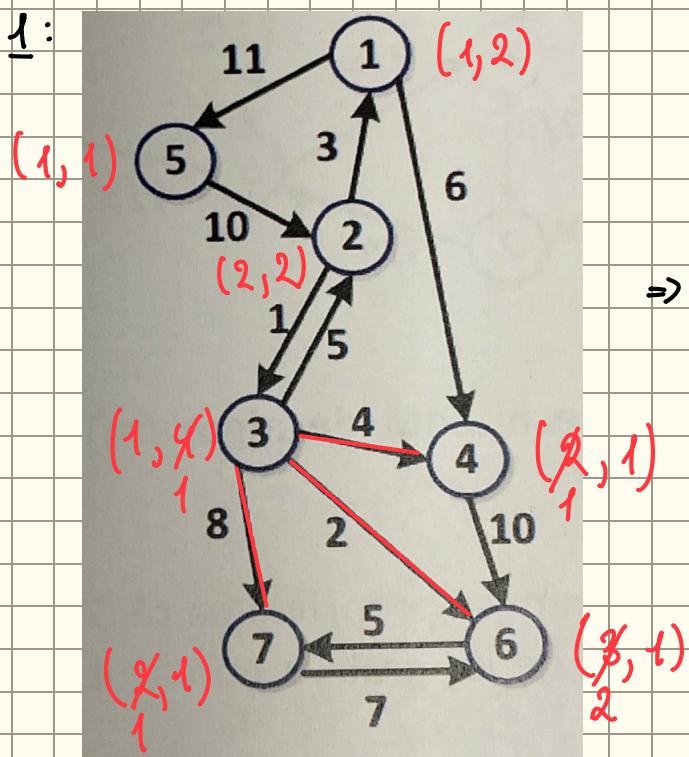
$$\exists x, y \in V \quad d^-(x) = d^+(x) - 1$$

$$d^-(y) = d^+(y) + 1$$

$$d^f(v) = d^-(v) \quad \forall v \neq x, y$$

C! conex + vf isolate

1:



$$\Rightarrow d^-(1) \notin d^+(1).$$

1, 5, 2, 3, 2, 1, 4, 6, 7, 6.

5. if  $D[j][k] > D[j][i] + D[i][k]$   $\forall (j, k)$

$i=1$  vf 1 ca vf intermediar :  $[2, 1, 4] ; [2, 1, 5] \Rightarrow D[2][4]; D[2][5]$ .

$i=2$  vf 1, 2 ca vf intermediar :  $[5, 2, 3]; [3, 2, 1]; [3, 2, 1, 5]; [3, 2, 1, 4]$   
 $D[5][3] \quad D[3][1] \quad D[3][5] \quad \text{nu se act.}$

$i=3$

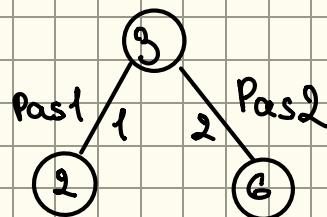
[5,2,1,4]

[5,2,1,5]

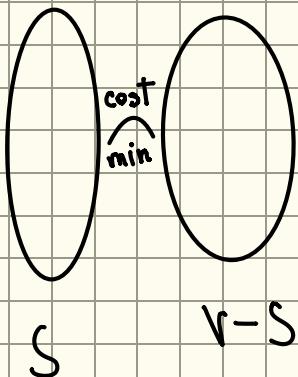
"  
NU

6) Pas 1  $V = \{3\}$  - alege (3,2)

Pas 2  $V = \{3,2\}$  - alege (3,6)  
cost 2



7)

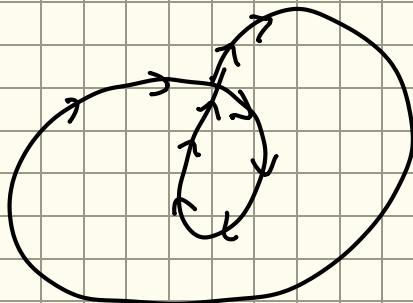


9) a) graf complet cu  $\neq$  noduri

$\forall$  graf planar e 6-colorat  $\Rightarrow$  nr cromatic  $\leq 6$

b)

c) pt fiecare comp conexă privită ca un graf există un ciclu eulerian  $(x_0, x_1, \dots, x_{n-1}, x_n)$  orientată muchiile în sensul dat de ciclul  $x_i, x_{i+1}$



De către ori un vf x apără în ciclu va avea un arc

$$b) M = (V, E, F)$$

$$\text{gr min} = 2$$

Arătați că  $\exists$  cel puțin 3 vf cu  $\text{gr} \leq 5$ .

$$\sum_{f \in F} d_M(f) = 2m$$

$$\sum_{v \in V} d_M(v) = 2m \rightarrow 6(n-2) + 2 + 2$$

$$m \leq 3n - 6$$

$$n-m+|F|=2$$

$$\Rightarrow 2m \geq 6n - 12 + 4 = 6n - 8$$

$$m \geq 3n - 4$$

$$\boxed{m \leq 3n - 6}$$