

# SEMINAR 1

Def.:  $f \in O(g)$  dacă  $\exists c, m_0 > 0$  a. i.  $\forall m \geq m_0, f(m) \leq cg(m)$   
 $m^2 \in O(m^3)$

$$100m^2 \in O(m^2)$$

$$m \log m \notin O(n \log \log n)$$

$$2^{m+1} \in O(2^m)$$

$$2^{2^m} \notin O(2^m)$$

R. prim red. la absurd că  $2^{2^m} \in O(2^m)$ .  $\Rightarrow$   
 $\Rightarrow \exists m_0, c$  a. i.  $\forall m \geq m_0, 2^{2^m} \leq c \cdot 2^m \Leftrightarrow 2^m \leq c (\infty)$

Def.:  $f \in \Omega(g)$  dacă  $\exists c, m_0 > 0$  a. i.  $\forall m \geq m_0, f(m) \geq cg(m)$

$$m^2 \in \Omega\left(\frac{m^2}{100}\right)$$

$$m^2 \notin \Omega(m^2 \log^* m) \xrightarrow{\text{de către ori aplic log ca}} \text{sa obtinem}$$

Def.:  $f \in \Theta(g)$  dacă  $\exists c_1, c_2, m_0 > 0$  a. i.  $\forall m \geq m_0, c_1 g(m) \leq f(m) \leq c_2 g(m)$ .

Pr.:  $f(m) \in \Theta(g)$  dacă  $f(m) \in O(g) \wedge f(m) \in \Omega(g)$

1) " $\Rightarrow$ "  $f \in O(g) \Rightarrow \exists c_1, c_2, m_0 > 0$  a. i.  $\forall m \geq m_0,$   
 $\begin{cases} c_1 g(m) \leq f(m) \\ c_2 g(m) \geq f(m) \end{cases} \Rightarrow \begin{cases} f \in \Omega(g) \\ f \in O(g) \end{cases}$

2) " $\Leftarrow$ "  $f \in O(g) \Rightarrow \exists c_1, m_1 > 0$  a. i.  $\forall m \geq m_1, f(m) \leq c_1 g(m)$

$f \in \Omega(g) \Rightarrow \exists c_2, m_2 > 0$  a. i.  $\forall m \geq m_2, f(m) \geq c_2 g(m)$

Fie  $M_0 = \max(m_1, m_2)$ .  $\Rightarrow$  ambele cond. sunt satisfăcute  $\Rightarrow \exists c_1, c_2, m_0 > 0$  a.t.  $\forall n \geq M_0$ ,  
 $c_2 g(n) \leq f(n) \leq c_1 g(n)$ .

$$n \in O(n \log n)$$

$$n \notin O(100n)$$

$$n \notin O(\log n)$$

Def.:  $f \in o(g)$  dacă  $\forall c > 0, \exists M_0 > 0$  a.t.  $\forall n \geq M_0$ ,  
 $f(n) < c g(n)$ .

$$n! \in o(n^n)$$

$$\log(n!) \notin o(n \log n)$$

$$2^m \in o(2^{2^m})$$

① Dem. că dacă  $\exists$  exact 2 noduri de grad impar  
dintr-un graf, sunt conectate.

P.p. prin red. la absurd că nodurile u și v  
de grad impar sunt în componente conexe dist.

Fie  $C_1$  componenta conexă a lui  $u$ .

$\sum_{v \in C_1} d(v)$  - impar, deoarece  $u$  este singurul nod  
de grad impar - contradicție, deoarece

$$\sum_{v \in C_1} d(v) = 2 \cdot \text{nr. de muchii}$$

$$② L = \{0^{k_1 k} \mid k \geq 1\}$$

- algoritm cu timp de rulare  $O(n^2)$

- algoritm cu timp de rulare  $O(n \log n)$

1) verific dacă nr. de 0 și nr. de 1 au aceeasi  
paritate

2) dacă nu, resping sirul

3) marchez 0 și 1 din 2 în 2

4) dacă am rămas cu exact un 0 și un 1, atunci accept, altfel merg la 1)

③ Se dau 2 nr.  $x$  și  $y$ . să se det. dacă  $x$  și  $y$  sunt prime între ele.

- Euclid:  $O(\log n)$

- scăderi repetitive:  $O(n)$

$$x+y \in O(n)$$

Mărimea inputului este  $O(\log n)$ , iar  $n$  este exponentional față de  $\log n$ !

Fie  $G$  un graf cu  $n \geq 6$ . Arătați că  $G$  sau  $\bar{G}$  are un ciclu de lungime 3.

$\Leftrightarrow$  Fie  $G$  un graf complet. Arătați că oricum am colorat muchiile lui  $G$  cu 2 culori, există întotdeauna un ciclu de lungime 3 colorat cu aceeași culoare

12. X. 2022

## CURS 2

P - probleme rezolvabile în timp polinomial

NP - probleme verificabile în timp polinomial

Problemele NP - complete au proprietăți:

-  $\in NP$ ;

- sunt NP-hard,

Bt. o def. nouă a de NP-hardness, avem nevoie de urm. 2 def.:

Def. 1: O funcție  $f: \Sigma^* \rightarrow \Sigma^*$  este calculabilă în timp polinomial dacă  $\exists$  o m. f. care rulează în timp polinomial și pt. orice sir  $m \in \Sigma^*$ , se opreste cu  $f(m)$  pe bandă (stare fin.).

Def. 2: O problemă A se red. în timp polinomial la o problemă B dacă  $\exists$  o funcție calculabilă în timp polinomial  $f$  a. s.  $\forall i \in A$ , avem  $i \in A \Leftrightarrow f(i) \in B$ . Not.  $A \leq_p B$  ( $A$  se red. în timp polinomial la  $B$ ).

O problemă P este NP-hard dacă  $\forall P' \in \text{NP}$ ,  $P'$  se red. în timp polinomial la problema P.

def. 2  $\Rightarrow$  dacă putem rezolva B în timp polinomial atunci putem rezolva A în timp polinomial

Prima problemă care a fost dem. NP-completă este SAT (problemă satisfacibilității) de către Cook și Levin, dem. în carteia lui Sipser: „Introduction to the theory of computing”.

Karp a dem. că 21 de probleme sunt NP-comple (1971).

Ex.: reducție de la 3-SAT la vertex cover

3-SAT:  $\phi = (x_1 \vee \bar{x}_2 \vee \bar{x}_3) \wedge (\bar{x}_1 \vee x_2 \vee \bar{x}_3) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee x_3)$   
 $x_1, x_2, x_3 = T \Rightarrow \phi - \text{true}$

Se dă o formulă booleană în care fiecare clauză are 3 literali. Sa se decidă dacă  $\exists$  o asignare a var care satisfac formula.

## VERTEX COVER:

Se dă un graf  $G$  și un nr.  $K$ . să se decidă dacă  $\exists$  un vertex cover cu  $\leq K$  noduri.

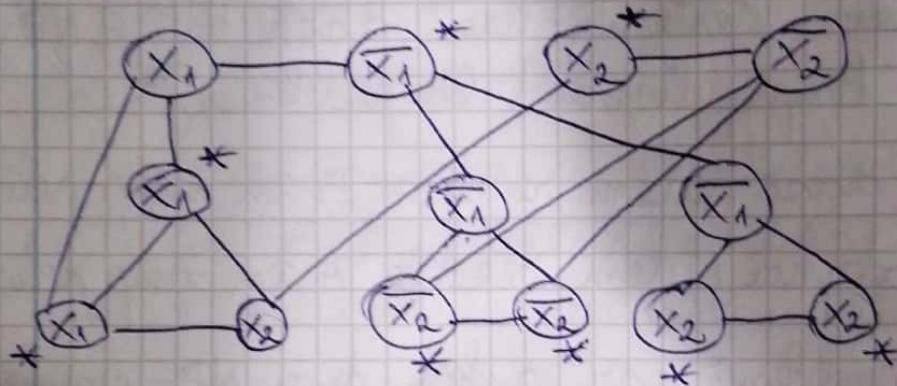
$$\begin{aligned}\emptyset = (x_1 \vee x_1 \vee x_2) \wedge (\overline{x}_1 \vee \overline{x}_2 \vee \overline{x}_2) \wedge (\overline{x}_1 \vee x_2 \vee x_2) \Rightarrow \\ \Rightarrow (G, K) \text{ a.z. } \emptyset \text{ este satisfiabilă} \Leftrightarrow G \text{ are vertex cover } \leq K\end{aligned}$$

$$x_1 = F, x_2 = T$$

$\emptyset$  - m var. și m clauze

$G = 2m + 3m$  noduri

$$K = m + 2m$$



1)  $\emptyset$ -satisfiabilă  $\Rightarrow G$  are vertex cover  $\leq m+2m$

2)  $G$  are vertex cover  $\leq m+2m \Rightarrow \emptyset$ -satisfiabilă

19.X.2022

## CURS 3

1) Dem. de NP-completitudine : 3-SAT  $\rightarrow$  clică

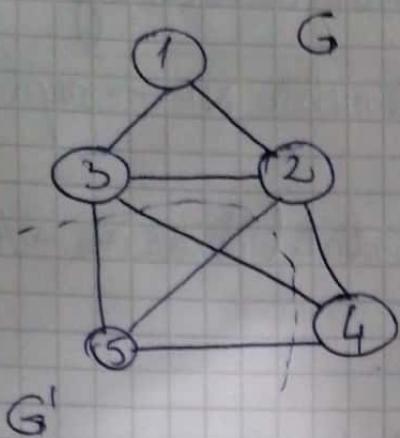
2) Introd. în alg. aprox.

Data trecută: 3-SAT  $\rightarrow$  vertex cover

În acest curs, vom arăta că problema clică este NP-completă printr-o reducție de la problema 3-SAT.

Def. (problema clicii): Se dă un graf neorientat  $G$  și un nr.  $K$ . Trebuie să decidă dacă  $G$  are o clică  $\geq K$  noduri.

Ex.:



$$G, K=3 \rightarrow DA\{1, 2, 3\}$$

$$G, K=4 \rightarrow NU$$

$$G', K=4 \rightarrow DA\{2, 3, 4, 5\}$$

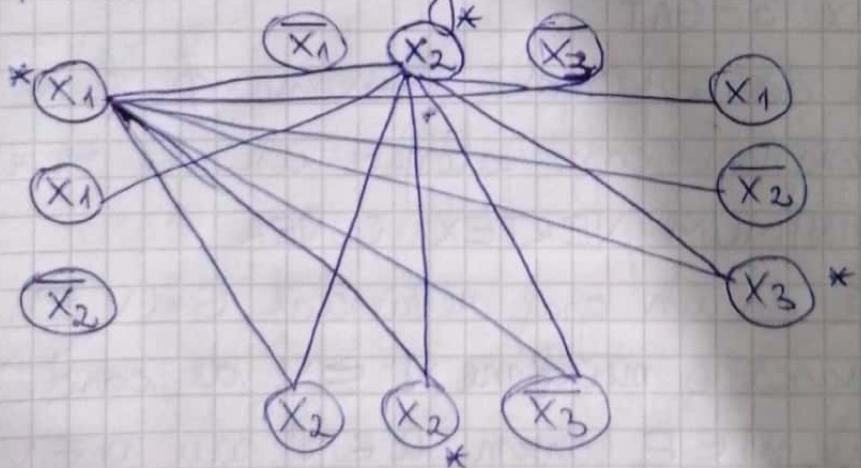
$$G', K=5 \rightarrow NU$$

Def.: Fie  $G=(V, E)$  un graf neorientat. O submulțime  $V' \subseteq V$  este o clică dacă  $\forall u, v \in V'$ , atunci  $(u, v) \in E$ .

$$\Phi = (x_1 \vee x_2 \vee \bar{x}_2) \wedge (\bar{x}_1 \vee x_2 \vee \bar{x}_3) \wedge (x_1 \vee \bar{x}_2 \vee x_3) \wedge (x_2 \vee x_2 \vee \bar{x}_3)$$

$$x_1=T, x_2=T, x_3=T$$

$m$  clauze  $\Rightarrow m$  gr. de 3 noduri



$\Phi$  - satisfiabilă  $\Leftrightarrow G$  are o clică de mărimea  $m$  care nu are muchie între 2 noduri și îndată:

- a)  $u \neq v$  și  $u, v$  nu sunt în același grup;
- b)  $u \neq v$  și  $u, v$  nu au etichete care corespund lui  $x_i$  și  $\bar{x}_i$ .

" $\Rightarrow$ "  
" $\Leftarrow$ "

Un compendiu de probleme NP-complete: Garey & Johnson - "Computers and Intractability".

Metode de abordare a problemelor NP-complete:

1. ~~backtracking~~
2. alg. de aprox. cu fact. garantat (Vazirani - Approximation algorithm)
3. alg. fixed param.
4. euristici
5. probleme de programare pe întregi

### Algoritmi de aproximare

Obs.: Până acum, ne-am concentrat pe probleme de decizie, acum vom discuta despre probleme de optimizare.

Ex.: - MAX 3-SAT

Se dă o formulă booleană  $\Phi$ . să se găsească o așezare care satisfacă un nr. maxim de clauze

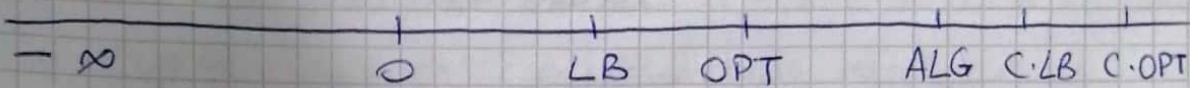
- MINIMUM VERTEX COVER

Se dă un graf neorientat  $G = (V, E)$ . să se găsească o mulțime  $V' \subseteq V$  de card. min. a.i.  $\forall (u, v) \in E$  avem  $u \in V'$  sau  $v \in V'$ .

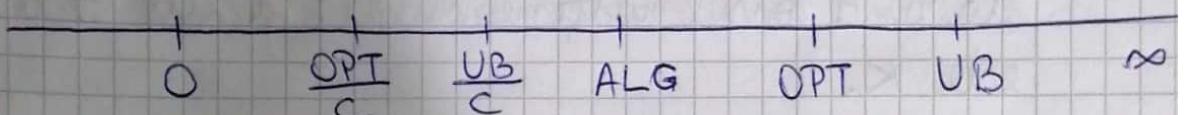
Def.: Un alg. A este o c-aprox. pt. o problema P, dacă pe orice instanță i a problemei P, avem  $A(i) \leq c \cdot OPT(i)$  dacă P este problema de min. sau  $A(i) \geq OPT(i)/c$ , dacă P este problema de max., unde  $A(i) =$  val. funcției

obiectiv pe instanță și  $\text{OPT}(i) = \text{val. sol. optimă pe instanță } i$ ; A rulează în timp polinomial.

- minimizare (ex. vertex cover)



- maximizare



Def.: Fix  $G = (V, E)$  un graf neorientat.  $M \subseteq E$  este un cuplaj dacă  $\forall e, e' \in M$ ,  $e$  și  $e'$  nu au capete în comun.

Def.: Un cuplaj  $M$  este maximal dacă nu se poate extinde adăugând o muchie.

Lemă: Într-o un graf  $G$ ,  $\text{vertex cover} \geq |M|$ ,  $M$  este un cuplaj maximal.

Algoritm de 2-aprox. pt. vertex cover:

1. Calculăm un cuplaj maximal

2. Adăugăm ambele capete în vertex cover

Dem.: a) alg. este corect

b) alg. este 2-aprox.

a) Prin red. la absurd, dacă  $\exists (a, b) \in E$

neacop.  $\Rightarrow (a, b) \cup M$  - cuplaj  $\Rightarrow M$  nu este maximal

b)  $\text{ALG} = 2 \cdot |M| \leq 2 \cdot |VC|$

- este algoritmul analizat precis? DA
- putem avea un alg. < 2-aprox. pt. VC? \$ \$\$

20. X. 202

## SEMINAR 2

### ① Subset sum

Se dă o multime de nr. S și un nr. K. să se decida dacă  $\exists$  o submultime de nr. a lui S a cărei sumă este K.

Scopul este să dem. că Subset sum este NP-complet printr-o reducție de la problema 3-SAT.

$$\emptyset \rightarrow (S, K)$$

$$\emptyset\text{-Sat.} \Leftrightarrow (S, K) = \text{DA}$$

$$\emptyset = (x_1 \vee x_2 \vee \bar{x}_3) \wedge (x_2 \vee x_3 \vee \bar{x}_3) \wedge (x_1 \vee x_3 \vee \bar{x}_2) \wedge (x_3 \vee \bar{x}_1 \vee \bar{x}_2)$$

	1	2	3	4	5	6	7	
$*y_1$	1	0	0	1	0	1	0	$y_1$
$\bar{z}_1$	1	0	0	0	0	0	1	$\bar{z}_1$
$y_2$	0	1	0	1	1	0	0	$y_2$
$\bar{z}_2$	0	1	0	0	0	1	1	$\bar{z}_2$
$*y_3$	0	0	1	0	1	1	1	$y_3$
$\bar{z}_3$	0	0	1	1	1	0	0	$\bar{z}_3$
$C_1$	0	0	0	1	0	0	0	$C_1$
$C_1'$	0	0	0	1	0	0	0	$C_1'$
$C_2$	0	0	0	0	1	0	0	$C_2$
$C_2'$	0	0	0	0	1	0	0	$C_2'$
$C_3$	0	0	0	0	0	1	0	$C_3$
$C_3'$	0	0	0	0	0	1	0	$C_3'$
$C_4$	0	0	0	0	0	0	1	$C_4$
$C_4'$	0	0	0	0	0	0	1	$C_4'$

$$\begin{array}{c} \Rightarrow \\ \Leftarrow \end{array} \begin{array}{l} y_i \in SOL \Rightarrow x_i = T \\ z_i \in SOL \Rightarrow x_i = F \end{array}$$

## ② Maximum independent set

Se dă un graf neorientat  $G = (V, E)$ . Să se găsească multimea  $i \subseteq V$  de cardinalitate maximă astfel încât:

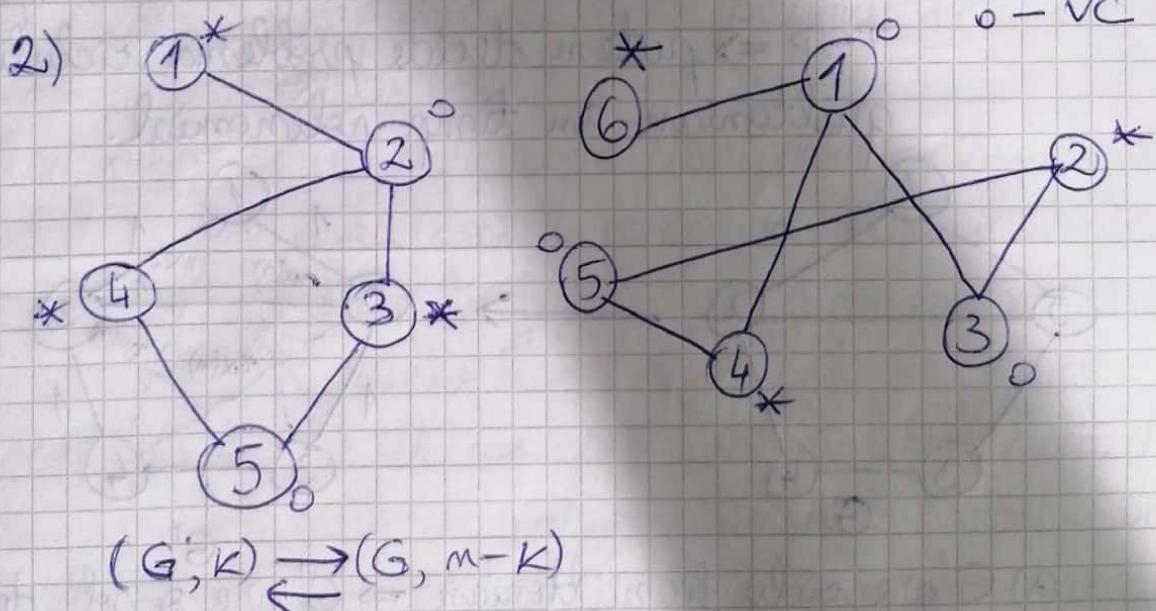
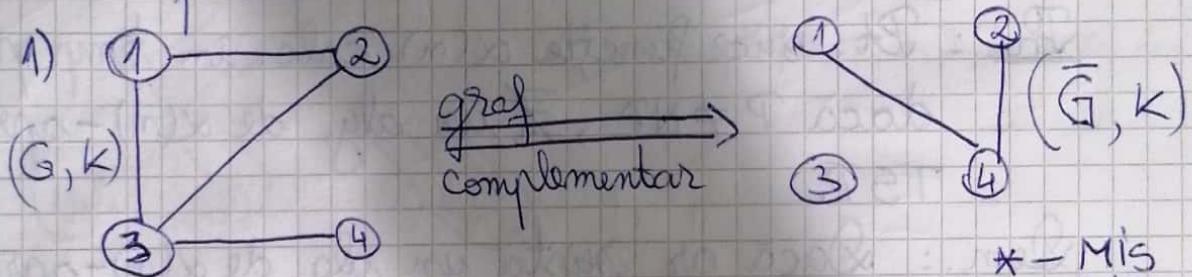
$\forall a, b \in i, \text{ să avem } (a, b) \notin E$ .

Problema de decizie:

Se dă  $(G, k)$ . Decideti dacă  $G$  are un M.I.S. de mărime  $\geq k$ .

Ex.: 1) Faceti o reducție de la problema M.I.S. la problema ciclu.

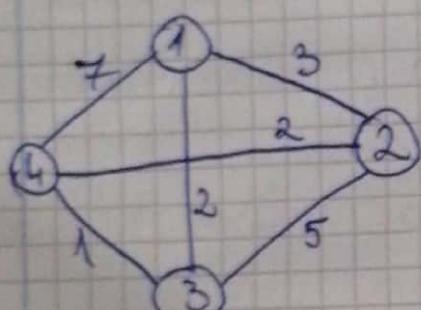
2) Faceti o reducție de la problema M.I.S. la problema vertex cover.



# CURS 4

- 1) Travelling Salesman Problem
- 2) Minimum Steiner Tree

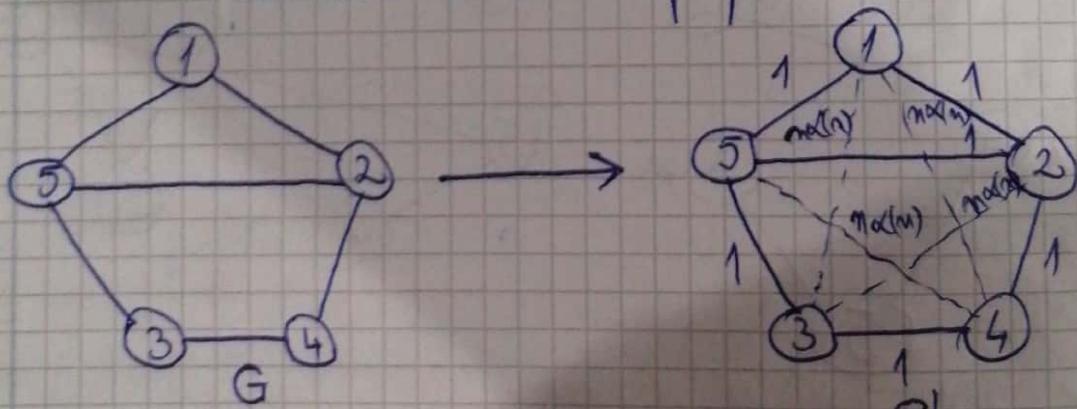
Problema 1. Se dă un graf neorientat complet și o dist.  $d: V \times V \rightarrow \mathbb{R}_+$ . Se cere să se găsească un tur de cost min., care vizitează fiecare nod exact o dată.



$$\text{cost}(1, 2, 4, 3, 1) = 3 + 2 + 1 + 2 = 8$$

Tez.: Pt. orice funcție  $\alpha(n)$  calc. în timp polinomial dacă  $P \neq NP$ ,  $\exists$  un alg. de  $\alpha(n)$ -aprox. pt TSP.

Dem.: Dacă ar exista un alg. de  $\alpha(n)$ -aprox. pt TSP  $\Rightarrow$  putem decide problema ciclului hamiltonian în timp polinomial.



- $G$  are ciclu hamiltonian  $\Rightarrow G'$  are o sol. de cost  $\leq n \cdot \alpha(n)$   $\Rightarrow$  alg. aprox. îoa returnă  $\leq n \cdot \alpha(n)$
- $G$  nu are ciclu hamiltonian  $\Rightarrow$  în  $G'$ , toate sol. au cost  $> n \cdot \alpha(n)$

Dacă TSP ar avea un alg. polinomial de  $\mathcal{O}(n)$ -opera  
am rula alg. pe graful  $G'$  și am putea distinge între  
casurile a) și b).  $\Rightarrow$  putem decide dacă  $G$  are ciclu  
hamiltonian.

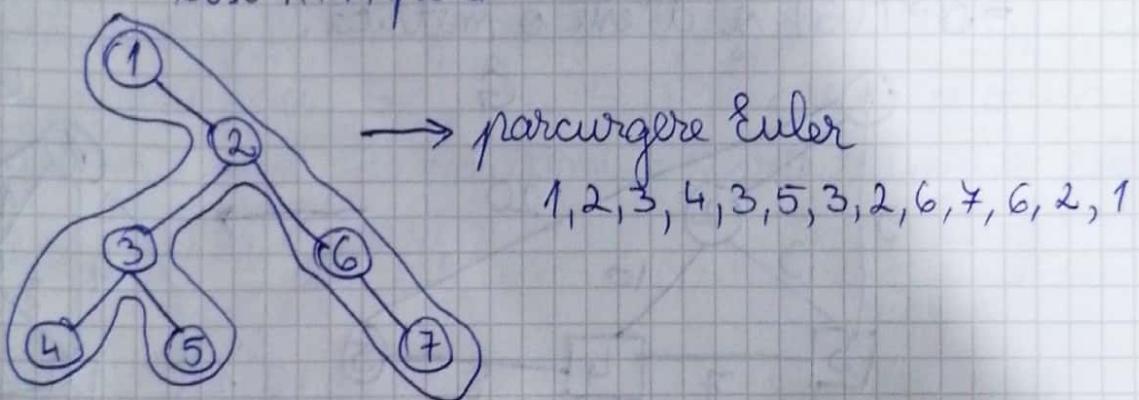
Propri. metrice:

- $\forall a \in X, d(a, a) = 0$
- $d(a, b) = d(b, a), \forall a, b \in X$
- $\forall a, b, c \in X, d(a, b) + d(b, c) \geq d(a, c)$

Metric TSP: 2-aprox.

1. Găsim LB pt. TSP

Obs. 1:  $MST(G) \leq TSP(G)$   
cost "APM pe  $G$



Obs. 2: O parcursere Euler a unui APM are cost  $\leq 2 \cdot TSP$ .

Obs. 3: O parcursere Euler neînt-circuitată va avea cost  $\leq$  parcurserea Euler initială (datorită faptului că dist. este o metrică) și implicit va avea cost  $\leq 2 \cdot TSP$ .

Problema 2. Se dă un graf neorientat și o dist.  $d: E \rightarrow \mathbb{R}$ . De asemenea, nodurile sunt partitionate în  $V = S \cup R$ , unde  $S =$  multime noduri Steiner și  $R =$  multime noduri necentrali. Se cere să se găsească un APM care conține toate nodurile din  $R$  și nu împreună alcătuiează un ciclu.

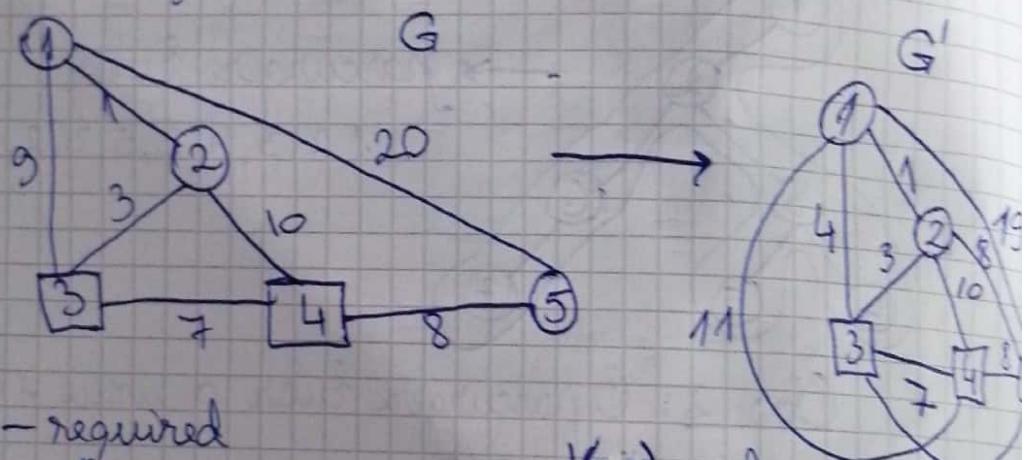
## CURS 5

Def.: Minimum Steiner Tree

Le dă un graf neorientat  $G = (V, E)$  și  $d : E \rightarrow \mathbb{N}$   
 $V = SUR$ ,  $S \cap R = \emptyset$ . Le cere să se găsească un graf  
care conectează toate medurile din  $R$  și pe cel  
meduri din  $S$ .  $\leq$

Obs.: Este suficient să studiem această problemă în cazul în care d este o metrică.

Th.: Fix  $G = (V, E)$  un graf. Putem construi într-un  
polinomial un graf  $G'$  și o dist.  $d'$  a. z.  $\text{OPT}(G) = \text{OPT}(G')$   
dacă și  $d'$  este o metrică.



O - required

- Feiner

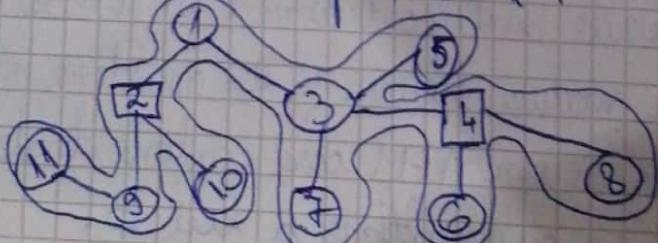
$d_{\text{G}}(u, \emptyset) = \text{cel mai scurt distanță}$

de la u la s en

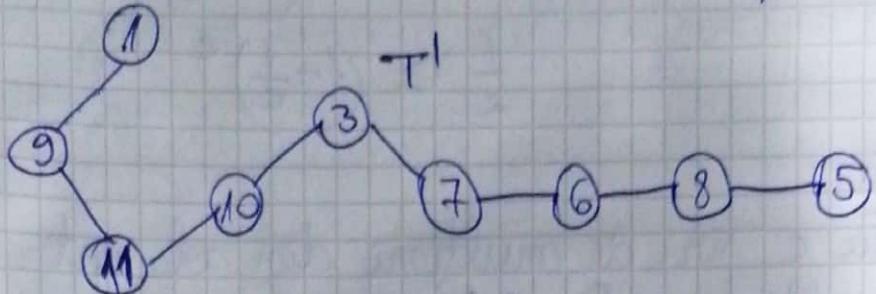
Alg. de 2-aprox. pt. Steiner Tree pe dist. metri

Construim un arbore parțial de cost minim, pe nodurile required.

Fie o sol. optimă  $T$ .

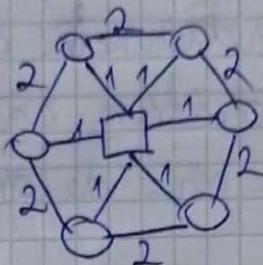


Pot construi un arbore parțial  $T'$  pe nodurilele  
requisite a.i.  $\text{cost}(T') \leq 2 \cdot \text{cost}(T)$ .  $\Rightarrow$   
 $\Rightarrow$  arborele parțial de cost minim.  $T''$  nu are  
 $\text{cost}(T'') \leq \text{cost}(T') \leq 2 \cdot \text{cost}(T)$  (pe nodurile R)



$$\text{OPT} = m$$

$$\text{ALG} = 2(m-1)$$



Set cover: Se dă o mulțime de elem.  $U = \{e_1, \dots, e_m\}$   
și  $n$  mulțimi  $S_1, \dots, S_n \subseteq U$ . Se cere să se găsească  
un nr. min. de mulțimi a căror reuniune este  $U$ .

Ex.:  $U = \{1, 2, 3, 4\}$

$$\begin{aligned} S_1 &= \{1\} \\ S_2 &= \{1, 3\} \\ S_3 &= \{2, 3\} \\ S_4 &= \{3, 4\} \\ S_5 &= \{2, 4\} \end{aligned} \quad \left. \right\} \Rightarrow S_2 \cup S_5 = U$$

Obs.: Vertex cover este un caz particular al set  
cover.

Alg. de ln m - aprox. pt. set cover

La fiecare pas, alegem mulțimea care acoperă cele  
mai multe noduri.

$S_1$	
$S_2$	

$$\text{OPT} = 2$$

$$\text{ALG} = \log_2 m$$

$$e^z = 1 + z + \frac{z^2}{2!} + \frac{z^3}{3!} + \dots$$

$$\begin{aligned} e^z &\geq 1 + z \\ z := -\frac{1}{c} &\Rightarrow e^{-\frac{1}{c}} \geq 1 - \frac{1}{c} \\ e^{-1} &\geq \left(1 - \frac{1}{c}\right)^c \\ \frac{1}{e} &\geq \left(1 - \frac{1}{c}\right)^c \end{aligned}$$

Dem.:

Fie  $c$  nr. de multimi din sol. opt. Fie  $g$  nr. de multimi returnate de alg. de approx.

Vrem să arătăm că  $g \leq c \cdot \ln m$ .

Def.  $m_i$  = nr. de elem. neacoperite la pasul  $i$ .

$$m_0 = m$$

Vrem să calculăm  $m_i$ .

$$m_i \leq m_0 - \frac{m_0}{c} \leq m_0 \left(1 - \frac{1}{c}\right)$$

$$\underbrace{\quad}_{\in \text{multimi}} \mid m$$

$\exists$  o multime care acoperă  $\frac{m}{c}$  elem.

$$m_2 \leq m_1 - \frac{m_1}{c} \leq m_1 \left(1 - \frac{1}{c}\right) \leq m_0 \left(1 - \frac{1}{c}\right)^2$$

|

$$m_i \leq m_{i-1} - \frac{m_{i-1}}{c} \leq m_0 \left(1 - \frac{1}{c}\right)^i$$

$$1 \leq m_g \leq m \left(1 - \frac{1}{c}\right)^g$$

$$1 \leq m \left(1 - \frac{1}{c}\right)^g = m \left[\left(1 - \frac{1}{c}\right)^c\right]^{\frac{g}{c}} \leq m \left(\frac{1}{e}\right)^{\frac{g}{c}}$$

$$1 \leq m \left(\frac{1}{e}\right)^{\frac{g}{c}} \mid \ln()$$

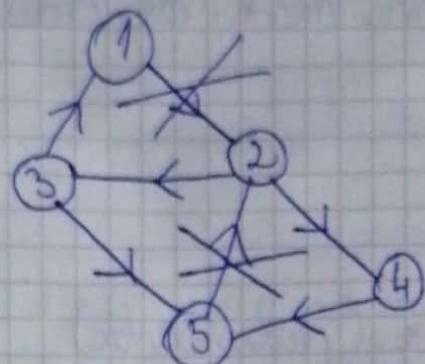
$$0 \leq \ln m - \frac{g}{c}$$

$$\frac{g}{c} \leq \ln m \mid \cdot c$$

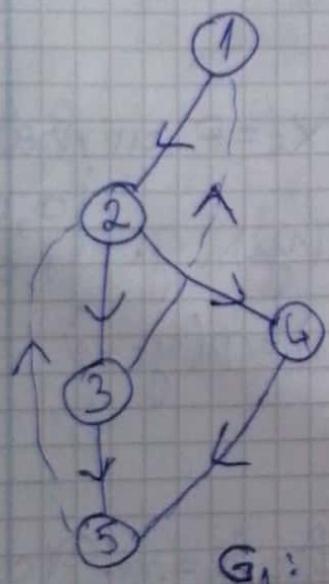
$$g \leq c \cdot \ln m$$

# SEMINAR 3

① Se dă un graf orientat  $G = (V, E)$ . să se prezinte un nr. maxim de muchii a.t. graful să devină aciclic. să se găsească un alg. de 2-aprox. pt. problema.



UB trivial este  $|E|$ .

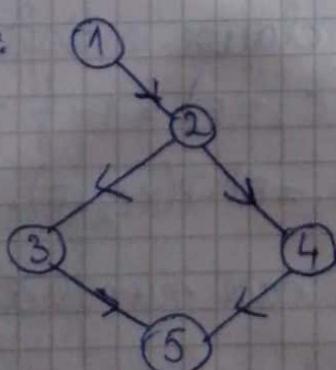
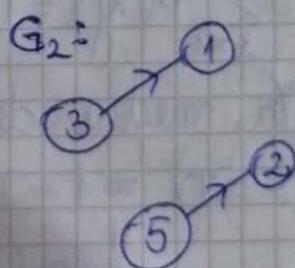


$$E = E_1 \cup E_2$$

$$E_1 \cap E_2 = \emptyset$$

$$\begin{aligned} G_1 &= (V, E_1) \\ G_2 &= (V, E_2) \end{aligned} \quad \left\{ \text{aciclice} \right.$$

$$\Rightarrow \max(|E_1|, |E_2|) \geq \frac{|E|}{2}$$



② 2-SAT

$$\phi = C_1 \wedge C_2 \wedge \dots \wedge C_m$$

$$C_i = (x_{i1} \vee x_{i2})$$

2-SAT  $\in P$

MAX 2-SAT: să se găsească o asignare care satise face un nr. maxim de clauze.

$$C_i = (x_{i1} \vee x_{i2})$$

$$C_i = F \Rightarrow \begin{cases} x_{i1} = F \\ x_{i2} = F \end{cases}$$

③ MAX 3-SAT:  $\frac{7}{8}$ -aprox. (probabilistic)

$E \rightarrow$  medie (expectation)

$$E(ALG) \geq \frac{7}{8} OPT$$

Se căută  $x_i = T$  cu prob.  $\frac{1}{2}$  și  $x_i = F$  cu prob.  $\frac{1}{2}$ .

$$E(\text{nr. clauze satisfăcute}) = \sum_{i=1}^m E(y_i), y_i = \begin{cases} 0, c_i-\text{nrasf.} \\ 1, c_i-\text{satisf.} \end{cases}$$

$$E[y_i] = 0 \cdot P(y_i=0) + 1 \cdot P(y_i=1) = P(y_i=1)$$

$$= \sum_{i=1}^m P(y_i=1)$$

Nr. medie de clauze satisfăcute =  $m \cdot P(c_i-\text{satisf.})$

$$P(c_i = T) = 1 - P(c_i = F) = 1 - \frac{1}{8} = \frac{7}{8}$$

$$= 1 - P(x_{i1}=F) \cdot P(x_{i2}=F) \cdot P(x_{i3}=F)$$

$Y =$  var. aleatoare care numără clauzele satisfăcute

$$Y = Y_1 + \dots + Y_m$$

④ Să se găsească un „maximal matching” cu nr. min. de muchii.  $\downarrow$  cuplaj maximal.

2-aprox.

Trebuie să dem. că un cuplaj maxim are  $\leq$  2. muchii dintr-un cuplaj maximal.

Oare muchie dintr-un cuplaj maximul atinge cel puțin o muchie din cuplajul maxim.

9. XI. 2022

## CURS 6

### Integer Linear Programming (ILP)

#### - Linear programming (LP)

Ex.: I) maximizăm  $2x + 3y$  cu urm.

constrângeri:  $x + y \leq 30$ ,  $x \leq 20$ ,  $y \leq 12$ ;  $x, y \geq 0$   
 $x, y \in \mathbb{R}$

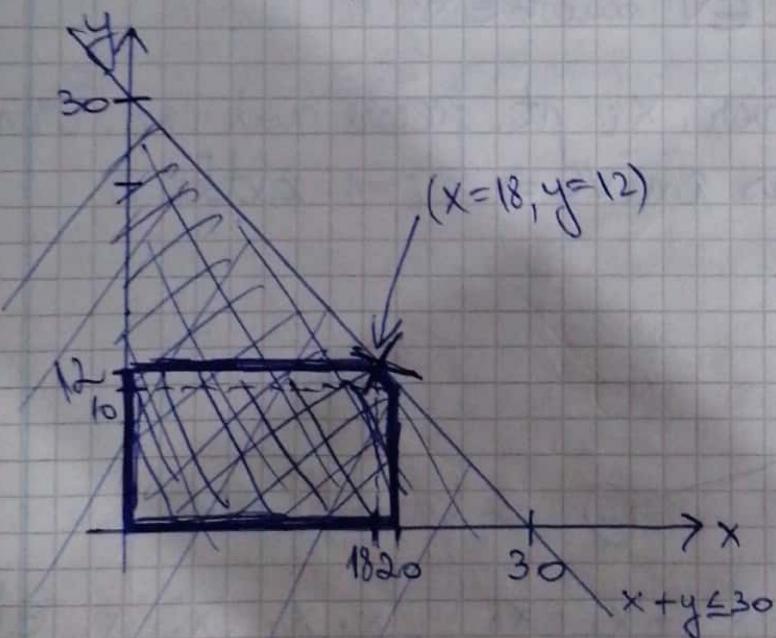
Pe caz general: maximizăm  $c_1x_1 + c_2x_2 + \dots + c_nx_n$   
a. z.:  $a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n \leq b_1$

$$a_{21}x_1 + a_{22}x_2 + \dots + a_{2n}x_n \leq b_2$$

⋮

⋮

$$a_{m1}x_1 + a_{m2}x_2 + \dots + a_{mn}x_n \leq b_m$$



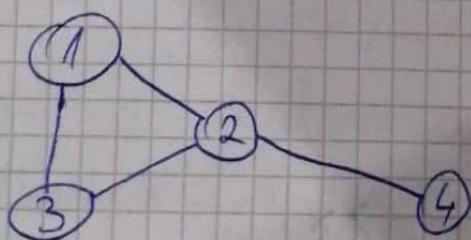
- Problemele de programare liniară se pot rezolva în timp polinomial.
- Algoritmul comun pt. a rezolva LP este simplex
- Simplex, deși rapid în practică, nu rulează în timp polinomial.
- Totuși, putem rezolva LP folosind metoda ellipsoidului, care este în timp polinomial.

iLP este NP-completă.

Plan: Formulăm vertex cover ca iLP, apoi folosim LP-ul rezultat ca LB pt. problema  $\Rightarrow$  vom rezolva LP în timp polinomial și vom reconstrui o sol. pt. vertex cover

Vertex cover: Se dă un graf neorientat  $G = (V, E)$ . Să se găsească o mulțime  $V' \subseteq V$  de cardinalitate minimă astfel încât  $\forall (a, b) \in E$ , avem  $a \in V'$  sau  $b \in V'$

Creăm o var.  $x_i$  pt. fiecare nod  $v_i$ . Vrem ca  $x_i$  să fie ales în  $V' \Rightarrow x_i = 1$  altfel  $x_i \in \{0, 1\}$



$$\min. x_1 + x_2 + x_3 + x_4, \quad x_1, x_2, x_3, x_4 \in \{0, 1\}$$

$$\text{constr.: } x_1 + x_2 \geq 1$$

$$x_1 + x_3 \geq 1$$

$$x_2 + x_3 \geq 1$$

$$x_2 + x_4 \geq 1$$

$$\min \sum_{i=1}^m x_i \quad \text{a. i. } \forall (x_i, x_j) \in E, x_i \in \{0, 1\}$$

Rezolvăm relaxarea LP în timp polinomial și obținem sol.  $x_i^* \in \mathbb{R}$

$$\textcircled{1} \rightarrow \textcircled{2} \quad \min. x_1 + x_2$$

$$x_1 + x_2 \geq 1$$

$$0 \leq x_1, x_2 \leq 1$$

$$x_1, x_2 \in \mathbb{R}$$

Lop.: vrem să transformăm  $x_i^*$  în nr. întregi  $x_i'$   
 a. i.  $\sum_{i=1}^m x_i' \leq 2 \sum_{i=1}^m x_i^*$ , cu cond. ca toate  
 constrângările să se respecte.

Rotunjire (LP rounding)

$$x_i^* \geq \frac{1}{2} \Rightarrow x_i' = 1$$

$$x_i^* < \frac{1}{2} \Rightarrow x_i' = 0$$

Trebui să arătăm:

a)  $\sum_{i=1}^m x_i' \leq 2 \cdot \sum_{i=1}^m x_i^*$  (fact. de 2-aprox.)

$$x_i' \leq 2 \cdot x_i^*, \forall i = 1, n$$

b) Vrem ca sol.  $x_i'$  să fie fezabilă, adică  $x_i' + x_j' \geq 1$   
 $\forall (i, j) \in E$

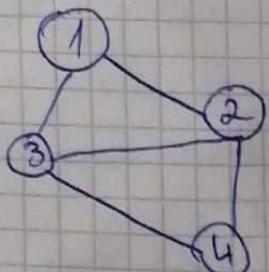
$$x_i^* + x_j^* \geq 1 \Rightarrow x_i^* sau x_j^* \geq \frac{1}{2} \Rightarrow x_i' sau x_j' \geq \frac{1}{2}$$

Set cover: Se dă o multime de elem.  $U$  și o colecție de multimi  $S$ . Se re găsească o submultime  $S' \subseteq S$  de cardinalitate min. a căror reunie este  $U$ .

$\forall s \in S$ ,  $x_s = 1$  dacă  $s$  este aleasă în sol.

$$\text{min. } \sum_{s \in S} x_s \text{ a. z. } \sum_{\substack{s \in S \\ x_s \neq 1}} x_s \geq 1, \quad \forall e \in U$$

$x_s \in \{0, 1\}$



VC

$$U = \{(1,2); (1,3); (2,3); (3,4); (3,4)\}$$

$$S = \{S_1 = \{(1,2); (1,3)\}; S_2 = \{(2,1); (2,3); (2,4)\}$$

$$S_3 = \{(3,1); (3,2); (3,4)\}; S_4 = \{(4,2); (4,3)\}$$

SC

Algoritm: i) Rezolvăm varianta relaxată și fie  $x^*$  sol. obținută - Fix  $C = \emptyset$ .

2) Repetăm  $c$ -lnc(n) urm. operație ( $c = ct.$  pe care să alegem mai târziu). Adăugăm multimea la sol.  $C$  cu prob.  $x^*$

1) Vrem să calculăm costul mediu al lui  $C$ .

2) Vrem să arătăm că  $C$  este o sol. fezabilă.

Fix  $e \in U$  care apare în  $k$  multimi, care este prob. ca  $e$  să nu fie acoperit după o iteratie a pasului 2? ( $e$  apare în  $S_1, \dots, S_k$ )

$$\Pr(e \text{ să nu fie acoperit}) = \Pr(e \text{ să nu fie acop. de } S_1 \text{ de către } k \text{ multimi})$$

$\dots \Pr(e \text{ să nu fie acop. de } S_n)$

$$\leq \left(1 - \frac{1}{k}\right)^k \leq \frac{1}{e}$$

$$\Pr(e \text{ să nu fie acop.}) \leq \left(\frac{1}{e}\right)^{\ln(kn)} = \frac{1}{e^{kn}} \leq \frac{1}{c^n} \cdot \frac{1}{c^k} = \frac{1}{c}$$

$$\Pr(\exists e \in U \text{ a. z. } e \text{ nu e acop.}) \xleftarrow{\text{după ln}(cn) pași}$$

$$\mathbb{E}[\text{cost } C] \leq \ln cm \cdot \text{OPT}_{LP}$$

$$\begin{aligned} \text{La un pas, avem } \mathbb{E}\left[\sum_{j \in S} x_j^*\right] &= \sum_{j \in S} \mathbb{E}[x_j^*] = \\ &= \sum_{j \in S} P(x_j^* = 1) = \\ &= \sum_{j \in S} X_j^* = \text{OPT}_{LP} \end{aligned}$$

$$\mathbb{P}(\text{cost } C \geq 2 \ln cm \cdot \text{OPT}_{LP}) \leq \frac{1}{2}$$

$$\mathbb{P}(X \geq a) \leq \frac{\mathbb{E}[X]}{a} \quad (\text{Markov})$$

16. XI. 2022

## CURS 7

1. Set Cover - Dual Fitting

2. Knapsack - FPTAS (Full Polynomial Time Approximation Scheme)

① Set Cover

$$U = \{e_1, \dots, e_m\}$$

$$S = \{S_1, \dots, S_m\}, S_i \subseteq U, \forall i \in \overline{m}$$

Fă și găseșcă un nr. minim de multimi din  $S$  care acoperă  $U$ .

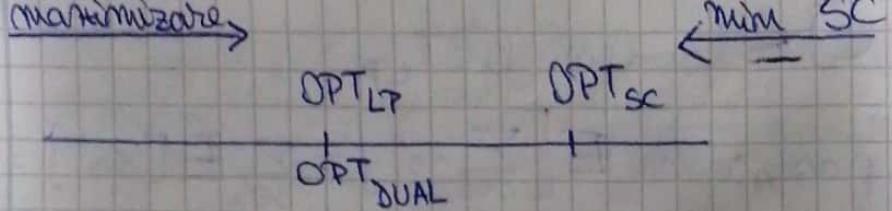
Algoritm Greedy de  $O(\log n)$  - aprob.

$$C = \emptyset$$

while  $C \neq U$

alegem  $S \in S$  a. z.  $S \setminus C$  să fie max.

$$C = C \cup S$$



1º. Cum construim un program dual?

Ex.: max.  $x_1 + x_2$

a. x.  $2x_1 + x_2 + 4x_3 \leq 3$

$x_1 + x_2 - 3x_3 \leq 1$

$x_1, x_2, x_3 \in \mathbb{R}$

DUAL

min.  $3y_1 + y_2$

a. x.  $2y_1 + y_2 \geq 1$

$y_1 + y_2 \geq 1$

$4y_1 - 3y_2 \geq 0$

PRIMAL

min.  $\sum_{s \in S} x_s$

a. x.  $\sum_{S \not\models e \in X_0} x_s \geq 1, \forall e \in U$

$x_s \in \{0, 1\} \quad 0 \leq x_s \leq 1$

DUAL

max.  $\sum_{e \in U} y_e$

a. x.  $\sum_{e \in S} y_e \leq 1, \forall S \subseteq S$

$0 \leq y_e \leq 1$

Def.  $\text{price}_e = \frac{1}{15 \setminus c_l}$ , unde  $S$  este multimea care acoperă elem.  $e$ .

Ex.: ①  $\mathcal{U} = \{1, 2, 3, 4, 5\}$

$$\mathcal{S}_1 = \{1, 2\}; \mathcal{S}_2 = \{2, 3\}; \mathcal{S}_3 = \{1, 4, 5\}; \cancel{\mathcal{S}_4 = \{3, 4, 5\}}$$

$$S_2: P_2 = P_3 = P_4 = \frac{1}{3}$$

$$S_3: P_1 = P_5 = \frac{1}{2}$$

$\sum_{e \in \mathcal{U}} \text{price}_e = \text{nr. de multimi din sol. returnata de alg.}$

②  $\mathcal{U} = \{1, 2, 3, 4, 5, 6\}$

$$S_1 = \{1, 2\}$$

$$S_4 = \{2, 4, 5\}$$

$$S_2 = \{2, 3, 4\}$$

$$S_5 = \{1, 3, 6\}$$

$$S_3 = \{1, 4, 5\}$$

$C = \text{multimea de elem. acop. la pasul curent}$

$$\text{Pas 1: } -C = \emptyset$$

$$\text{- alegem } S_2: -P_2 = P_3 = P_4 = \frac{1}{3}$$

$$-C = \{2, 3, 4\}$$

$$\text{Pas 2: - alegem } S_3: -P_1 = P_5 = \frac{1}{2}$$

$$-C = \{1, 2, 3, 4, 5\}$$

$$\text{Pas 3: - alegem } S_5: -P_6 = 1$$

$$-C = \{1, 2, 3, 4, 5, 6\}$$

$$\text{Iată } \text{price}_e = \frac{\text{price}_e}{H_n}, H_n = 1 + \frac{1}{2} + \dots + \frac{1}{n} \in O(\log n)$$

Vrem să arătăm că  $y$  este o sol. fezabilă.  
(care respectă constrângările) pt. DUAL.

$$\text{Adică } \sum_{e \in S} \frac{\text{price}_e}{H_n} \leq 1, \forall S \in \mathcal{P}.$$

Fie  $S$  o multime  $\Rightarrow S = \{e_1, \dots, e_k\}$ .

$e_1, \dots, e_k$  au fost acop. de alg. în ordinea  
creștească.  $\Rightarrow \text{price}_{e_1} \leq \frac{1}{k}, \text{price}_{e_2} \leq \frac{1}{k-1}, \dots, \text{price}_{e_k} \leq 1$

$$\Rightarrow \sum_{e \in S} \text{price}_e \leq H_K \Rightarrow \frac{\sum_{e \in S} \text{price}_e}{H_M} \leq \frac{H_K}{H_M} \leq 1 \Rightarrow$$

$\Rightarrow$  Y este sol. fezabilă

$$\begin{aligned} \text{Am arătat că } OPT_{SC} &\geq \frac{\sum_{e \in U} \text{price}_e}{H_M}. \\ \text{Dar } ALG &= \frac{\sum_{e \in U} \text{price}_e}{H_M} \end{aligned} \quad \left. \right\} \Rightarrow$$

$$\Rightarrow ALG \leq H_M \cdot OPT_{SC}$$

## ② Knapsack

Se dă un rucsac de dim.  $K \leq n$  elem.  $a_1, \dots, a_m$ .  
Fiecare elem. are un profit  $P(a_i)$  și o mărime  $\text{size}(a_i)$ . să se găsească o multime de elem. care au suma mărimilor  $\leq K$  și profit maxim.

Cst.  $P$  - profitul maxim al unui elem. din multime

$\exists$  un alg. de rezolvare care rulează în timp  $O(n^2)$

Knapsack este NP-completă.

Def. (FPTAS): Un alg. de aprox. pt. o problemă dătă este FPTAS dacă  $\forall i$ -instanță a problemei și  $\forall \epsilon > 0$ , avem :

$ALG(i) \leq (1 + \epsilon) OPT(i)$  dacă dătă este problema de min.

SAU

$ALG(i) \geq (1 - \epsilon) OPT(i)$  dacă dătă este problema de max.

și alg. rulează în timp polinomial  
în  $n$  și  $\frac{1}{\epsilon}$   $\leftarrow$  fără această cond. (PTAS)

# SEMINAR 4

① Greedy:

Sortăm obiectele descr. după  $\frac{P_i}{S_{i+1}}$  și le punem în rucsac în această ordine.

$$\mathcal{O} = \{ \cancel{(1,1)}, \cancel{\{(8,4); (6,3); (3,2)\}}, \cancel{(7,1)} \}$$

capac. rucsac = 5

greedy  $\Rightarrow (8,4) : \text{cost} = 8$

optim  $\Rightarrow (6,3); (3,2) : \text{cost} = 9$

$K = 100\ 000$

$P_1 = 1, S_1 = 1$

$P_2 = 99\ 999, S_2 = 100\ 000$

ALG: 1, OPT: 99 999

Să se găsească o 2-aprox. pt. problema rucsacului.

$$\frac{P_1}{S_1} \geq \frac{P_2}{S_2} \geq \dots \geq \frac{P_k}{S_k} \geq \frac{P_{k+1}}{S_{k+1}} \geq \dots$$

încap în rucsac

Trebuie să arătăm că  $\underbrace{P_1 + P_2 + \dots + P_k}_A + \underbrace{P_{k+1}}_B \geq \text{OPT.}$

$$\text{ALG} = \max(A, B) \geq \frac{\text{OPT}}{2}$$

② Set cover: fiecare elem. apare în cel mult f multimi

$$\text{Ex.: } U = \{1, 2, 3, 4\}$$

$$S_1 = \{1, 2, 3\}$$

$$S_2 = \{1, 2, 4\} \quad f=3$$

$$S_3 = \{1, 4\}$$

$$S_4 = \{3, 4\}$$

f-aprox. pt. set cover

$x_i$  = de cate ori am ales elem. i

$$x_i \geq 1, \forall i = \overline{1, n}$$

$$\min. x_1 + x_2 + \dots + x_n$$

$$\text{constr.: } x_i \geq 1, \forall i = \overline{1, n}$$

$$VC = SC \text{ cu } f=2$$

$x_s = \begin{cases} 1, & \text{dacă } s \text{ este în } \text{sol.} \\ 0, & \text{altfel} \end{cases}$

$$\min. \sum_{s \in \Psi} x_s$$

a.i.

$$\sum_{s \in \text{sol.}} x_s \geq 1, \forall l \in U$$

$$x_s \in \{0, 1\} \text{ ILP}$$

$$0 \leq x_s \leq 1 \text{ LP}$$

Fie  $x_s^*$  sol. optimă la LP.

Cum construim  $x_s^*$  pe baza  $x_s^*$  a.i.  $\sum_{s \in \Psi} x_s^* \leq f \cdot \sum_{s \in \Psi} x_s^*$  (f-aprox.)  $\Rightarrow \sum_{s \in \text{sol.}} x_s^* \geq 1, \forall l \in U$  (iezabilitate).

$$x_s^* = \begin{cases} 1, & \text{dacă } x_s^* \geq \frac{1}{f} \\ 0, & \text{altfel} \end{cases}$$

$$x_s^* \leq f \cdot x_s^* \\ x_s^* + \dots + x_s^* \geq 1 \Rightarrow \exists x_{s,i}^* \geq 1$$

## CURS 8

## Knapsack

n obiecte  $\alpha_1, \dots, \alpha_m$

$\delta_i$ :  $p(\delta_i)$ -profit

$$\sigma(\delta_i) - \text{size}$$

D-capac. rucsacului; P-profitul maxim obiect  
Pcă P este polinomial în n.

Profitul max. pe care-l putem obtine este  $\pi_P$ .

Vrem să facem un alg. de programare dinamică.  
Considerăm o ordine arbitrară a obiectelor:  $o_1, \dots, o_n$ .

$A(z, p)$  = capac. min. a unei submultimi a obiectelor  $\sigma_1, \dots, \sigma_i$  care are profit exact  $p$  (sau  $\infty$  dacă  $\emptyset$ )

Nem calcula  $A(n_1, 0), \dots, A(n_r, mP)$  și alegem  $A(n, j)$  a. z.  $A(n, j) \leq D$  și  $j$ -max.

$A(m, j)$  a.s.  $A(m, j) \subseteq D$  n.s.  $j$ -max.

Cum calculăm  $A(1, p) = \begin{cases} n(\alpha_1), & \text{dacă } n(\alpha_1) = p \\ \infty, & \text{altele} \end{cases}$

$$A(i, p) = \min [A(i-1, p), \delta(x_i) + A(i-1, \overbrace{p - p(x_i)}^{\text{非负} \geq 0})]$$

Ex.: 3 objects, D = 3

$$P(\theta_1)=1, P(\theta_1)=2$$

$$P(\theta_2) = 2, \quad P(\theta_2) = 3$$

$$\rho(\theta_3) = 2, \sigma(\theta_3) = 1$$

	0	1	2	3	4	5	6
A[1]	0	2	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$
A[2]	0	2	3	5	$\infty$	$\infty$	$\infty$
A[3]	0	2	1	3	4	6	$\infty$

$$\mathcal{O}(n^2 \cdot p)$$

Algoritmul de aproximare are un factor de  $(1+\epsilon)$ ,  $\forall \epsilon > 0$

și rulează în timp polynomial în  $\frac{1}{\epsilon}$ .

Def.  $K = \frac{\epsilon \cdot P}{m}$ . Construim o multime de obiecte  $\omega_1', \dots, \omega_n'$ , unde  $p(\omega_i') = \left| f(\omega_i) \right|^K$ . Rezolvăm problema pe  $\omega_1', \dots, \omega_n'$  folosind alg. de programare dinamică.

Cel mai profitabil obiect va avea profit  $\frac{P}{K} = \frac{x \cdot m}{\epsilon P} = \frac{m}{\epsilon}$ .

Deci alg. va avea complexitatea  $\mathcal{O}\left(\frac{m^3}{\epsilon}\right)$ .

Faț. de aprox.:

$OPT =$  optimul pe instanță inițială

$OPT' =$  optimul pe instanță truncată

Vrem să arătăm că  $OPT' \geq (1-\epsilon)OPT \Rightarrow$

$$\cancel{K \cdot OPT'} \geq OPT - \epsilon \cdot OPT$$

$$\cancel{OPT - K \cdot OPT'} \leq \epsilon \cdot OPT$$

$$\cancel{K \cdot OPT'} \quad \cancel{OPT} \leq \epsilon \cdot OPT$$

$$p(\omega_i) \leq K \cdot p(\omega_i') \leq p(\omega_i)$$

$$\cancel{OPT - K \cdot OPT'} \leq m \cdot K \leq \epsilon \cdot P$$

$$\Rightarrow K \cdot OPT' \geq OPT - \epsilon \cdot OPT$$

$$\epsilon \cdot OPT \geq OPT - K \cdot OPT'$$

$$OPT - K \cdot OPT' \leq \epsilon \cdot OPT$$

$$OPT - K \cdot OPT' \leq m \cdot K = m \cdot \frac{\epsilon P}{m} = \epsilon P \leq \epsilon \cdot OPT$$

$$p(\omega_i) - K \cdot p(\omega_i') < K$$

$$P \leq OPT(A)$$

## Fixed parameter algorithms

Def.: O problemă este fixed parameter tractable dacă se poate rezolva cu un alg. cu timp de rulare  $f(K) \cdot P(n)$ , unde  $K$  este un param. indep. de input.

Ex. de param.: - mărimea sol.

- diam. unui graf

- gradul min. / max.

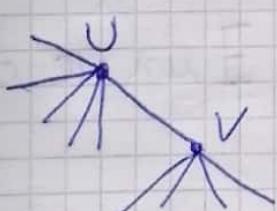
Ex.: vertex cover res. independent set

$\exists$  alg. cu timp  $\exists$  un alg. cu timp  
de rulare  $O(2^K \cdot n^2)$  de rulare  $n^{O(K)}$

$K$  = mărimea vertex cover

Vertex cover: Se dă un graf  $G$  și un nr.  $K$ . să se det. dacă  $G$  are un vertex cover de mărime  $\leq K$ .

Alg. brute force naiv va avea complexitate  $O(n^K)$ .  $\leftarrow$  nu este alg. fixed-param.



$\Rightarrow$  u va fi în vertex cover sau

v va fi în vertex cover

Alg.:  $(G, K)$

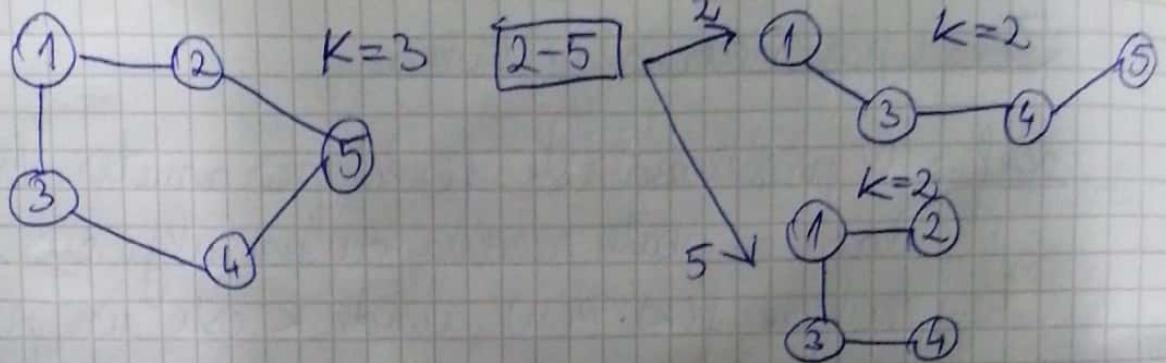
- dacă  $G = \emptyset \Rightarrow$  DA

- dacă  $K = 0 \Rightarrow$  STOP

- alegem o muchie arbitrară  $(u, v)$

apelăm  $ALG(G \setminus \{u\}, K-1)$ ;  $ALG(G \setminus \{v\}, K-1)$ ;

- afisăm NU



$$T(m, k) = 2T(m-1, k-1) + O(m)$$

7. XII. 2022

## CURS 9

Algoritmi fixed parameter (FPT)

Rularea în timp  $O(f(k) \cdot n^c)$ , unde:

- $c = \text{ct.};$
- $m = \text{dim. datelor de intrare};$
- $k = \text{param. indep. de datele de intrare (ex.: nr. de noduri din VC / val. sol. optime / diam. / alfabetul dacă avem o problemă cu siruri);}$

### Vertex cover

Se dă un graf  $G$  și un nr.  $K$ .  $\exists$  un VC de mărime  $\leq K$ ?

$$T(m, k) = 2T(m-1, k-1) + O(m).$$

### Kernelization

Un kernel este un alg. polinomial care transformă o instanță  $(I, k)$  a unei probleme într-o instanță  $(I', k')$  cu urm. proprieți:

- $k' \leq k;$
- $|I'| \leq f(k);$
- $(I, k) = \text{YES} \Leftrightarrow (I', k') = \text{YES}$

Th.: O problemă este FPT  $\Leftrightarrow$  admite un kernel.

" $\Rightarrow$ " problema este FPT  $\Rightarrow$  are un alg.  $O(f(k) \cdot n^c)$

- a)  $n > f(k) \Rightarrow$  alg. rulează în  $O(n^{c+1}) \Rightarrow$  YES NO
- b)  $n \leq f(k) \Rightarrow$  avem un kernel

" $\Leftarrow$ "  $I = I'$ ,  $k = k'$

Kernel pt. vertex cover:

- 1) dacă  $\exists v$  cu  $d(v) = 0 \Rightarrow (G, k) \rightarrow (G \setminus \{v\}, k)$
- 2) dacă  $\exists v$  cu  $d(v) > k \Rightarrow (G, k) \rightarrow (G \setminus \{v\}, k-1)$

$\Rightarrow G$  are  $\leq k(k+1)$  noduri sau  $G$  nu are  $VC \leq k$

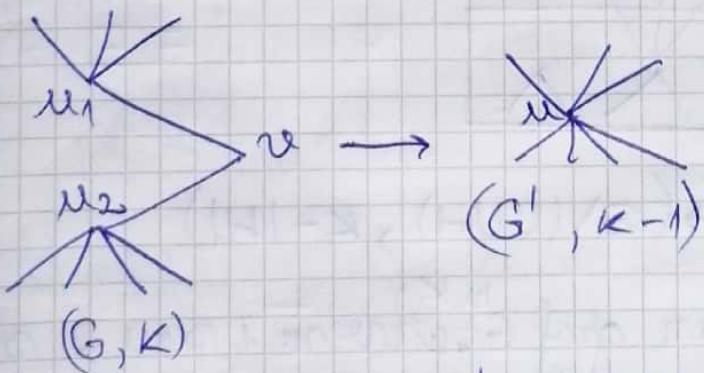
- 3) dacă  $\exists v$  cu  $d(v) = 1$ , fie  $u$  vecinul lui  $\Rightarrow$   
 $\Rightarrow (G, k) \rightarrow (G \setminus \{u, v\}, k-1)$

$$|V| \leq |E|$$

$$|V| \leq k^2$$

- 4) dacă  $\exists v$  cu  $d(v) = 2$  și vecinii lui,  $u_1, u_2$  sunt conectați  $\Rightarrow (G, k) \rightarrow (G \setminus \{u_1, u_2, v\}, k-2)$

- 5) dacă  $\exists v$  cu  $d(v) = 2$  și vecinii lui,  $u_1, u_2$  nu sunt conectați  $\Rightarrow (G, k) \rightarrow (G', k-1)$



$G$  are  $VC \leq k \Leftrightarrow G'$  are  $VC \leq k-1$

" $\Rightarrow$ "  $v$  este în  $VC$  și  $u_1, u_2$  nu sunt  
 $u_1$  sau  $u_2$  este în  $VC \Rightarrow v$  este în  $VC \Rightarrow$   
 $\Rightarrow$  în  $G'$  alegem  $u$  în  $VC$   
 $u_1$  și  $u_2$  sunt în  $VC \Rightarrow$  în  $G'$  alegem  $u$

" $\Leftarrow$ "  $u$  este un VC în  $G' \Rightarrow u, v, u_2$  sunt în VC  
în  $G$ , altfel  $v$  este un VC în  $G$

$$\sum_{v \in G} d(v) = 2|E|$$

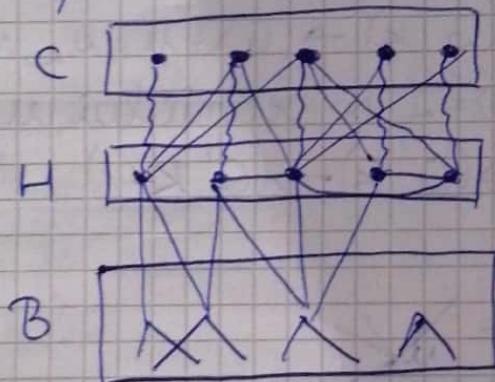
$$d(v) \geq 3 \Rightarrow \sum_{v \in G} d(v) \geq 3|V| \Rightarrow |V| \leq \frac{1}{3} \cdot 2|E| = \frac{2}{3}|E|$$

$$|E| \leq K^2 \Rightarrow |V| \leq \frac{2}{3}K^2$$

### Crown decomposition

Dându-se un graf  $G$ , un "crown decomposition" este o partitie a nodurilor  $C \cup H \cup B$ , unde:

- 1)  $C$  este o multime indep.;
- 2)  $\exists$  un cuplaj între nodurile din  $C$  și nodurile din  $H$ , care facec. toate nodurile din  $H$ ;
- 3)  $\nexists$  muchii între nodurile din  $C$  și cele din  $B$ .



$$(G, K) \rightarrow (G \setminus (C \cup H), K - |H|)$$

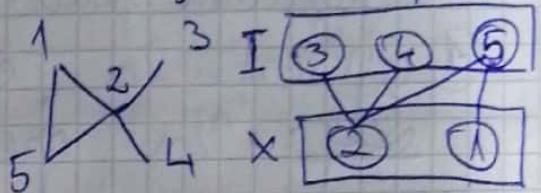
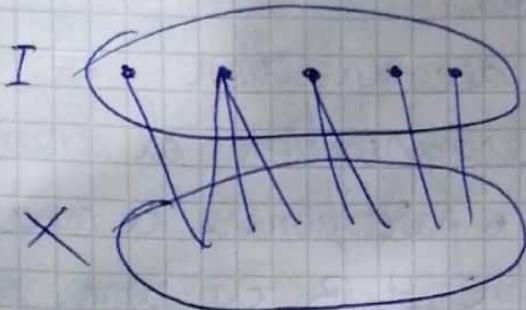
Lemă: Fie un graf  $G$ . Avem 1 sau 2 sau 3:

- 1)  $G$  are un cuplaj de mărime  $K+1$ ;  $\Rightarrow$  NU
- 2)  $G$  are crown decomposition;  $\Rightarrow$  reducere
- 3)  $G$  are  $\leq 3K$  noduri.  $\Rightarrow$  kernel

Fișe un cuplaj maximal  $\times$  în graf.

Dacă  $|X| \geq k+1 \Rightarrow$  cazul 1

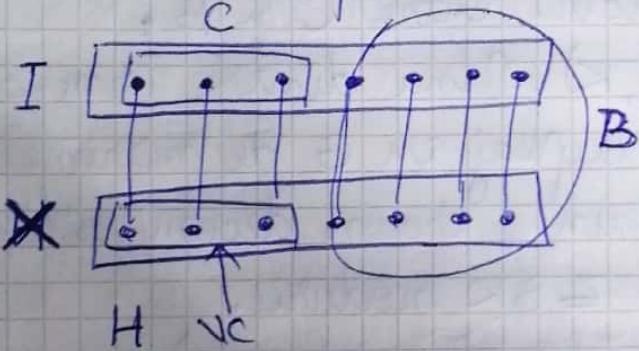
$G \setminus X$  este o multime indep. Graful în care considerăm doar muchii între  $X$  și  $I$  este bipartit.



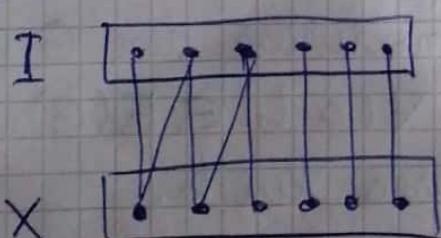
Pe grafuri bipartite, putem rezolva exact VC.

a) pe graful  $X \cup I$ , VC conține noduri din  $X \Rightarrow$

$\Rightarrow$  ele vor forma  $H$  din crown decomposition



b) toate nodurile din  $I$  sunt în VC



$$|I| \leq k$$
$$|X| \leq 2k$$

Dacă  $|I| > k$ , atunci  $VC > k$   $\rightarrow$

# CURS 10

## Algoritmi fixed parameter

① Aplicație la "crown decomposition":

Def.: Un "crown decomposition" al unui graf orientat  $G$  este o partitionare a nodurilor  $G$  în 3 multimi:  $C, H, B$  cu urm. prop.:

- $C$  este multime independentă;
- $\exists$  un cupluj între  $C$  și  $H$  care acoperă toate nodurile din  $H$ ;
- $\nexists$  muchii între  $C \times B$ ;

Lemă: Fie  $(G, K)$ . Una dintre urm. 3 este ade-

- 1)  $\exists$  un cupluj în  $G$  de mărime  $K+1$ ;
- 2)  $G$  are un "crown decomposition";
- 3)  $G$  are  $\leq 3K$  noduri.

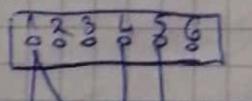
Problema (Dual of vertex coloring):

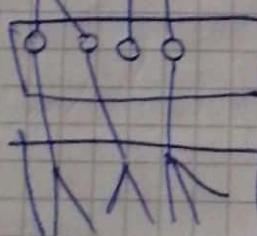
Se dă un graf  $G$  și un nr.  $K$ . Să se decid dacă  $\exists$  o colorare a nodurilor din  $G$  cu  $V(G)$ -câteva culori distincte, a. z.  $\forall (a, b) \in E(G)$  să avem culoarea lui  $a \neq$  culoarea lui  $b$ .

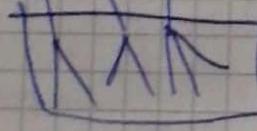
lema [1]  $\Rightarrow$  YES

lema [3]  $\Rightarrow$  kernel

lema [2]  $\Rightarrow$  vom lucra cu  $\overline{G}$

  $C$  independent in  $\overline{G} \Rightarrow$  clique in  $G$

  $H$  pt. nodurile din  $H$ , putem reflecă culorile din  $C$

  $B \quad (G, K) \rightarrow (G \setminus (C \cup H), K - |H|)$

② Se dau  $n$  puncte în plan și un nr.  $K$ . să se decidă dacă  $\exists K$  liniile drepte a  $\mathbb{R}$ . fiecare din cele  $n$  puncte să fie pe cel puțin  $K$  dreptă. (în timp  $f(K) \cdot n^c$ ).

$$\text{kernel} \leq K^2$$

Reducția:  $K+1$  puncte col.  $\Rightarrow$  le elimin și scad  $K+1$

Dacă avem  $> K^2$  puncte  $\Rightarrow$  NU

③ Se dă un graf  $G$  și un nr.  $K$ . să se decidă dacă putem elimina  $K$  muchii din  $G$  a. z. graful să nu aibă triunghiuri (circluri de lungime 3).

Căutăm un  $\Delta$ . Dacă găsim  $\Rightarrow$  OK. Altfel, eliminăm pe rând cele 3 muchii și aplicăm recursiv alg.

④ Color coding

Se dă un graf și 2 noduri  $s, t$  și un nr.  $K$ . să se decidă dacă  $\exists$  un drum de la  $s$  la  $t$  care conține  $K$  noduri distincte.

Algoritm probabilist:

Asignăm aleator o culoare din multimea  $\{1, \dots, K\}$  fiecărui nod și det. în (temp  $f(K) \cdot n^c$ ) dacă  $\exists$  un drum de la  $s$  la  $t$  care conține un nod din fiecare culoare.

Dacă  $\exists$  un drum de la  $s$  la  $t$  în  $G$  cu  $K$  noduri, care este probabilitatea să  $\exists$  un drum de la  $s$  la  $t$  în graful rezultat după colorare, care să conțină  $K$  culori distincte?

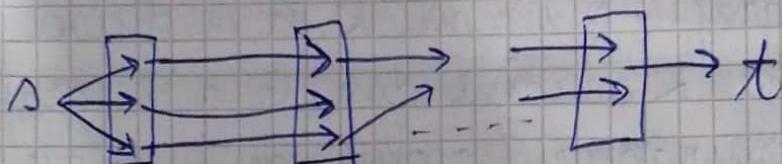
$$\frac{K!}{K^K} > e^{-K}$$

$$e^z = 1 + z + \dots + \frac{z^K}{K!}, z = K \Rightarrow e^K = 1 + K + \dots + \frac{K^K}{K!} \Rightarrow e^K > \frac{K^K}{K!}$$

Dacă repetăm alg. de  $e^k$  ori, probabilitatea să nu găsim o sol. dacă ea  $\exists$  (alg. să nu ruleze cum trebuie)  $= (1-e^{-k})^{e^k} < \frac{1}{e}$ .

Iterăm toate cele  $K!$  permutări și pt. fiecare dintre ele, verificăm dacă  $\exists$  un drum care conține culorile în această ordine.

$\pi(1)$     $\pi(2)$  ...    $\pi(K)$



- ⑤ Se dă un graf  $G$  și un nr.  $K$ . să se decidă dacă pot elimina  $K$  muchii a.ș. toate componentele conexe din graful rezultat să fie cíci!

21.XII.202

## CURS 11

### Algoritmi competitivi - online

- sunt alg. la care nu stăm tot inputul de la Ex.: ① Ski rental problem

-  $1 \$ / zi$  ca să închiriez schiuri

-  $B \$$  ca să cumpăr schiuri

- nu stiu câte zile are sezonul de schi

- în fiecare zi i va trebui să decid dacă închiriez sau cumpăr

Alg.: - închiriez  $(B-1)$  zile

- în ziua  $B$ , cumpăr

Def.: Spunem că un alg. A este  $\alpha$ -competitiv dacă pe orice instanță  $V$  avem  $A(V) \leq \alpha \cdot OPT(V)$ , unde  $A(V) =$  val. returnată de A pe instanța  $V$ , iar  $OPT(V) =$  val. returnată de sol. optimă pe instanța  $V$ .

Alg. pt. schi este  $(2 - \frac{1}{B})$ -competitiv.

$$1 \leq OPT \leq B \Rightarrow ALG = OPT \Rightarrow \alpha = 1$$

$$OPT \geq B \Rightarrow ALG = B + (B-1) = 2B-1$$

$$\frac{ALG}{OPT} = \frac{2B-1}{B} = 2 - \frac{1}{B} \Rightarrow \alpha = 2 - \frac{1}{B}$$

cumpărare      închiriere

- ② L secunde durează călăteria cu liftul de la etajul curent până la dest.

S sec. durează mersul pe scări

Nu știm după cât timp vine liftul. Cât așteptăm până decidem să urcăm pe scări?

Răsp.:  $S-L$  sec.

Cazuri:

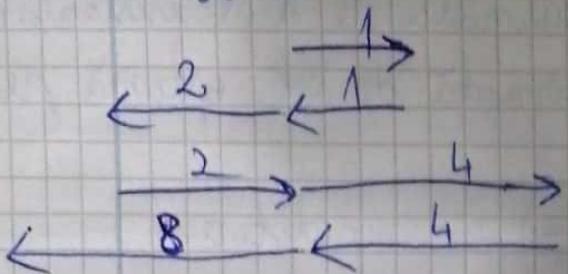
1°. liftul vine după ~~S~~  $S-L$  sec.

$$\begin{aligned} OPT &= T + L \\ ALG &= T + L \end{aligned} \quad \left\{ \Rightarrow \alpha = 1 \right.$$

2°. liftul vine după  $T > S-L$  sec.

$$\begin{aligned} OPT &= S \\ ALG &= 2S-L \end{aligned} \quad \left\{ \Rightarrow \alpha = \frac{2S-L}{S} = 2 - \frac{L}{S} \right.$$

$$\textcircled{3} \quad 2(1+2+2^2+\dots+2^{\log_2 D}) = 2 \cdot (2^{\log_2 D+1}-1) = \\ = 2^{\log_2 D+2}-2 = 4D-2 \Rightarrow \alpha =$$



$$1 + 1 + 2 + 2 + 4 + 4 + 8 + \dots + \frac{D}{2} + D = \\ = 1 + 3 \left( 1 + 2 + \dots + \frac{D}{2} \right) = 1 + 3 \cdot \frac{\frac{D}{2} \cdot \frac{D+2}{2}}{2} = \\ = 1 + 3 \cdot \frac{D(D+2)}{8}$$

$$= 1 + 3(1 + 2 + 2^2 + \dots + 2^{\log_2 D}) = 1 + 3(2D - 1)$$

#### (4) Steiner Tree - offline

Se dă un graf neorientat  $G = (V, E)$  și o mulțime de noduri  $R$  (required). Pe muchii sunt costuri  $C: E \rightarrow \mathbb{R}_+$ . Vrem un arbore parțial de cost minim care acoperă nodurile din  $R$  și posibil și altele din  $V \setminus R$ .

NP-hard, 2-aprox.

#### Steiner Tree - online

Se dă graful și nodurile din  $R$  apăr pe rând. Trebuie să conectăm nodul  $r_i$  de la pasul  $i$  în arborele construit până la pasul ant.

$$\alpha = \log_2 n$$

La fiecare pas, alegem unul dintre nodurile required de la pasul ant. I.e.  $d(r_i, u)$  să fie min. Adăugăm drumul  $r_i \rightarrow u$  în sol.

Pt. un nr.  $L$ , def.  $S_L = \{u \in R \mid c(u) \geq L\}$ .  
 $|S_L| \leq \frac{2 \cdot OPT}{L}$ ,  $OPT =$  arbore Steiner optim pe instanta data.

$$ALG = c(r_1) + \dots + c(r_k)$$

$$\text{Not. } L_1 = c(r_1)$$

$$\begin{matrix} | \\ | \\ L_k = c(r_k) \end{matrix} .$$

$$|S_{L_1}| = k$$

$$|S_{L_2}| = k-1$$

$$\begin{matrix} | \\ | \\ |S_{L_k}| = 1 \end{matrix}$$

$$|S_{L_i}| \leq \frac{2 \cdot OPT}{L_i} \Rightarrow L_i \leq \frac{2 \cdot OPT}{|S_{L_i}|} \leq \frac{2 \cdot OPT}{k-i+1}$$

$$\begin{aligned} ALG &= L_1 + \dots + L_k \leq \frac{2 \cdot OPT}{k} + \frac{2 \cdot OPT}{k-1} + \dots + \frac{2 \cdot OPT}{1} \leq \\ &\leq 2 \cdot OPT \underbrace{\left(1 + \frac{1}{2} + \dots + \frac{1}{k}\right)}_{\log_2 k} \end{aligned}$$

### Demonstratie lemei

Consider numai nodurile din  $S_L$ . Fix 2 noduri  $u, v \in S_L$ ,  $d(u, v) \geq L$ .

Un ciclu hamiltonian pe  $S_L$  va avea lungimea  $\geq |S_L| \cdot L$ .

OPT Steiner Tree pe multimea  $S_L$

$|S_L| \cdot L \leq$  ciclu hamiltonian  $\leq 2 \cdot OPT$