

ALGORITMICA GRAFURILOR

C. Croitoru

2013-2014

0. Probleme, Algoritmi, Complexitate

Vom considera o **problemă** (computațională) ca fiind o aplicație

$$\mathcal{P} : I \rightarrow O,$$

(I mulțimea intrărilor problemei, mulțimea instanțelor problemei; O mulțimea ieșirilor, răspunsurilor, soluțiilor) care pentru fiecare intrare $i \in I$ oferă o ieșire $\mathcal{P}(i) \in O$.

Dacă $O = \{da, nu\}$ atunci \mathcal{P} se va numi **problemă de decizie**, $\mathcal{P}(i) \in \{da, nu\}$ va fi răspunsul (la întrebarea pusă de \mathcal{P}), iar forma uzuală de prezentare a problemei va fi:

\mathcal{P} Intrare: $i \in I$.

Intrebare: descriere ... ?

Exemplu:

Compus Intrare: $n \in \mathbb{N}$ $n \geq 2$.

Intrebare: Există $p, q \in \mathbb{N}$ $p, q \geq 2$ și $n = pq$?

Un alt tip de probleme care apar frecvent sunt cele de **căutare** : mulțimea O conține pentru fiecare $i \in I$ măcar o soluție acceptabilă în raport cu un anumit criteriu precizat, iar problema cere găsirea unei astfel de soluții.

O clasă specială de astfel de probleme este cea a problemelor de **optimizare**, care sunt de **maximizare** sau **minimizare**.

Exemple:

Drum **Intrare:** G un graf, a, b două vârfuri ale lui G .
Ieșire: P un drum în G de la a la b (dacă \exists).

DrMin **Intrare:** G graf, a, b vârfuri în G ,
 o funcție de lungime a muchiilor lui G .
Ieșire: P^* un drum în G de la a la b cu suma lungimilor muchiilor **minimă**, printre toate drumurile de la a la b în G .

MaxCut **Intrare:** G graf,
 o funcție de lungime a muchiilor lui G .
Ieșire: O bipartiție (S, T) a mulțimii vârfurilor grafului G cu suma lungimilor muchiilor dintre cele două clase **maximă**.

Oricărei probleme de optimizare i se poate asocia o problemă de decizie (care ne va da informații asupra dificultății ei computaționale). Pentru cele două probleme de optimizare de mai sus, avem:

DrMin-D **Intrare:** G graf, a, b vârfuri în G , $k \in \mathbb{N}$,
o funcție de lungime a muchiilor lui G .
Întrebare: Există P drum în G de la a la b
cu suma lungimilor muchiilor $\leq k$?

MaxCut-D **Intrare:** G graf, $k \in \mathbb{N}$,
o funcție de lungime a muchiilor lui G .
Întrebare: Există (S, T) bipartiție a mulțimii
vârfurilor lui G cu suma lungimilor
muchilor dintre cele două clase $\geq k$?

Vom considera în continuare doar probleme de decizie.

Pentru a fi rezolvate cu ajutorul calculatorului problemele sunt **codificate**.

Vom considera Σ o mulțime finită fixată numită **alfabet**, Σ^* mulțimea tuturor **cuvintelor** peste Σ .

Obiectele care apar în descrierea unei probleme (numere raționale, grafuri, funcții, matrici etc.) vor fi reprezentate cu ajutorul unor cuvinte $w \in \Sigma^*$. **Lungimea** cuvântului w se va nota cu $|w|$. Orice mulțime de cuvinte peste Σ , deci o submulțime a lui Σ^* se numește **limbaj** (peste Σ).

Problemele de decizie au forma *Dat un anumit obiect, are el o anumită proprietate?* Cum obiectele le reprezentăm cu ajutorul cuvintelor, înseamnă că problema se reduce la întrebarea: *Dat un cuvânt, are el o anumită proprietate?*

Vom considera **problemă de decizie** o funcție

$$P : \Sigma^* \rightarrow \{da, nu\}.$$

Limbajul asociat problemei P este

$$P^{-1}(da) = \{w | w \in \Sigma^* \text{ și } P(w) = da\}.$$

Vom considera că un algoritm este o funcție *

$$A : \Sigma^* \rightarrow \{da, nu, nedecidabil\}.$$

Limbajul $L \subseteq \Sigma^*$ este **acceptat** de algoritmul A dacă $L = \{w | w \in \Sigma^* \text{ și } A(w) = da\}$.

Limbajul $L \subseteq \Sigma^*$ este **decis** de algoritmul A dacă $\forall w \in L : A(w) = da$ și $\forall w \notin L : A(w) = nu$.

Limbajul $L \subseteq \Sigma^*$ este **acceptat** de algoritmul A **în timp polinomial** dacă L este acceptat de A și $\exists k \in \mathbb{N}$ astfel încât pentru orice $w \in L$ algoritmul A evaluează $A(w) = da$ în timpul $\mathcal{O}(|w|^k)$.

Limbajul $L \subseteq \Sigma^*$ este **decis** de algoritmul A **în timp polinomial** dacă $\exists k \in \mathbb{N}$ astfel încât pentru orice $w \in \Sigma^*$ algoritmul A evaluează $A(w) = da$, dacă $w \in L$ și $A(w) = nu$, dacă $w \notin L$, în timpul $\mathcal{O}(|w|^k)$.

*Preferăm un mod informal de prezentare a noțiunilor de algoritm și timp de execuție

Clasa de complexitate \mathbf{P} :

$$\mathbf{P} = \{L \subset \Sigma^* | \exists A \text{ alg. a.}\hat{\text{t.}} L \text{ e decis de } A \text{ \textbf{în timp polinomial}}\}.$$

Dacă P e o problemă de decizie, atunci ea este **rezolvabilă în timp polinomial** dacă limbajul $L = P^{-1}(da)$ satisface $L \in \mathbf{P}$. Se notează aceasta (cam abuziv) $P \in \mathbf{P}$.

De exemplu, problema *DrMIN-D* este rezolvabilă în timp polinomial dacă funcția de lungime (specificată în intrarea ei) este cu valori nenegative. Dacă se permit și valori negative, atunci nu se cunoaște nici o demonstrație a apartenenței *DrMin-D* $\in \mathbf{P}$ (și nici nu se crede c-ar exista una). Apartenența *Compus* $\in \mathbf{P}$ s-a demonstrat abia în anul 2002.

Observații. 1. Dacă notăm

$$\mathbf{P}' = \{L \subset \Sigma^* | \exists A \text{ alg. a.}\hat{\text{t.}} L \text{ e acceptat de } A \text{ \textbf{în timp polinomial}}\},$$

se observă imediat că $\mathbf{P} \subseteq \mathbf{P}'$ și nu e dificil să se arate și incluziunea inversă, deci $\mathbf{P} = \mathbf{P}'$.

2. Se verifică ușor că dacă $P \in \mathbf{P}$ atunci și $\Sigma^* \setminus P \in \mathbf{P}$.

Verificare în timp polinomial.

Un algoritm de verificare este o funcție

$$A : \Sigma^* \times \Sigma^* \rightarrow \{da, nu, nedecidabil\}.$$

Pentru $A(x, y)$, x este intrarea iar y este **certificatul**.

Limbajul $L \subseteq \Sigma^*$ este **verificat** de algoritmul de verificare A dacă

$$L = \{w | w \in \Sigma^* \text{ și } \exists y \in \Sigma^* \text{ a. î. } A(w, y) = da\}.$$

Clasa de complexitate **NP**:

$$\mathbf{NP} = \{L \subseteq \Sigma^* \mid \exists A \text{ algoritm de verificare} \\ \text{cu timp de lucru polinomial a.î.}$$

$$L = \{x \in \Sigma^* \mid \exists y \in \Sigma^*, \exists k \in \mathbf{N} \text{ a. î.}$$

$$|y| = \mathcal{O}(|x|^k) \text{ și } A(x, y) = da\}$$

$$\}.$$

Dacă P e o problemă de decizie, atunci ea este problemă (din) **NP** dacă limbajul $L = P^{-1}(da)$ satisface $L \in \mathbf{NP}$.

Observație. NP este mnemonic pentru **Nedeterminist Polinomial** și nu pentru **NePolinomial**.

Un argument că nu e bine să asimilăm NP cu nepolinomial este și faptul că $P \subseteq NP$.

Justificarea este imediată: Dacă $L \in P$, atunci există $A : \Sigma^* \rightarrow \{da, nu, nedecidabil\}$ algoritm care decide L în timp polinomial. Considerăm $A' : \Sigma^* \times \Sigma^* \rightarrow \{da, nu, nedecidabil\}$, satisfăcând $A'(x, x) = A(x)$ pentru orice $x \in \Sigma^*$. Se vede ușor că L este verificat de A' în timp polinomial.

Din definițiile de mai sus ar fi fost mai normal să notăm VP (verificabil polinomial). Sintagma **Nedeterminist Polinomial** se justifică dacă am considera **algoritmi nedeterminiști** în care controlul execuției este astfel încât după fiecare pas este posibil să se execute unul oarecare dintre pașii specificați într-o mulțime finită de pași succesori. Un astfel de algoritm acceptă un cuvânt de intrare dacă este posibilă o execuție care să conducă la rezultatul *da*. Se poate arăta că aceasta definiție a acceptării nedeterminate este echivalentă cu cea de verificare dată mai sus, în care *certificatul* este utilizat pentru efectuarea alegerilor corecte ale pașilor următori ai unei execuții a algoritmului nedeterminist.

NP notează clasa problemelor de decizie pentru care răspunsurile afirmative au *certificate* care pot fi folosite pentru a demonstra succint (în timp polinomial) corectitudinea lor.

Intuitiv, **NP** este clasa tuturor problemelor de decizie pentru care se poate verifica un răspuns pozitiv (*da*) rapid dacă ni se dă o soluție.

De exemplu, pentru problema *MaxCut-D* un răspuns afirmativ are drept certificat o partiție (S^*, T^*) a mulțimii vârfurilor grafului (iar proprietatea că suma lungimilor muchiilor cu o extremitate în S^* și cealaltă în T^* nu depășește pragul k se face în timp polinomial). Deci, $MaxCut-D \in \mathbf{NP}$.

Dacă am considera următoarea problemă de decizie:

UnDrum **Intrare:** G un graf, a, b două vârfuri ale lui G .
Întrebare: \exists un unic drum de la a la b în G ?

Cum s-ar putea justifica un răspuns *da* la o problemă *UnDrum*? Dacă prezentăm un drum anume drept certificat, el nu ne asigură că nu mai există și altele. O demonstrație succintă (graful e mare) pare a nu exista. Deci nu știm dacă $UnDrum \in \mathbf{NP}$.

Clasa de complexitate $co\text{-}\mathbf{NP}$:

$$co\text{-}\mathbf{NP} = \{L \subseteq \Sigma^* \mid \Sigma^* \setminus L \in \mathbf{NP}\}.$$

O problemă de decizie $P \in co\text{-}\mathbf{NP}$ dacă $L = P^{-1}(da) \in co\text{-}\mathbf{NP}$ (echivalent, $L = P^{-1}(nu) \in \mathbf{NP}$).

Exemplu de problemă din $co\text{-}\mathbf{NP}$:

NeHam **Intrare:** G un graf.

Întrebare: Este adevărat că în G nu există un circuit care să treacă exact o dată prin fiecare vârf al său?

Observație. $\mathbf{P} \subseteq \mathbf{NP} \cap co\text{-}\mathbf{NP}$.

Reduceri polinomiale.

Fie $L_1, L_2 \subseteq \Sigma^*$. Spunem că L_1 se reduce polinomial la L_2 , și notăm aceasta prin $L_1 \propto L_2$, dacă există $f : \Sigma^* \rightarrow \Sigma^*$ o funcție polinomial calculabilă astfel încât $\forall w \in \Sigma^*$: $w \in L_1$ dacă și numai dacă $f(w) \in L_2$.

f se numește funcție de reducere și algoritmul polinomial F care calculează f , algoritm de reducere polinomială.

Observații. 1. Dacă $L \in \mathbf{P}$ și $L' \propto L$, atunci $L' \in \mathbf{P}$.

Fie A un algoritm polinomial care decide L și F un algoritm de reducere polinomială a lui L' la L . Atunci, $A' = A \circ F$ este un algoritm polinomial care decide L' . ($\forall x \in \Sigma^*$, $A'(x) = da \Leftrightarrow A(F(x)) = da \Leftrightarrow F(x) \in L \Leftrightarrow x \in L'$; A' e polinomial deoarece mulțimea polinoamelor e închisă la operația de compunere).

2. Relația \propto este tranzitivă: $L_1 \propto L_2, L_2 \propto L_3 \Rightarrow L_1 \propto L_3$.

Limbajul $L \subseteq \Sigma^*$ este **NP-difil** (engl. NP-hard) dacă $\forall L' \in \mathbf{NP}$ are loc $L' \propto L$.

Limbajul $L \subseteq \Sigma^*$ este **NP-complet** dacă $L \in \mathbf{NP}$ și L este NP-difil.

Terminologia se transferă pentru probleme de decizie:

Spunem că problema de decizie P_1 se reduce polinomial la problema de decizie P_2 , și notăm $P_1 \propto P_2$, dacă $L_1 = P_1^{-1}(da) \propto L_2 = P_2^{-1}(da)$.

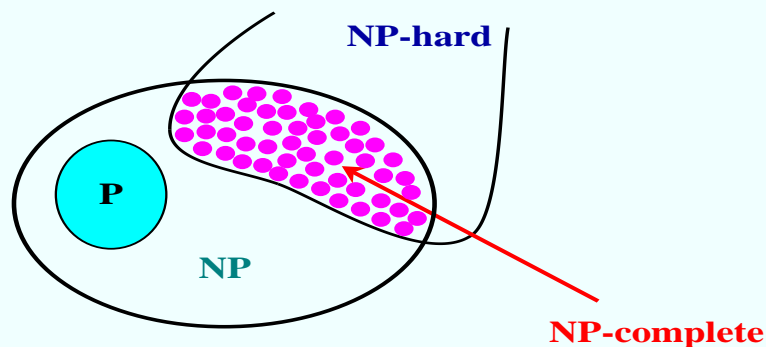
Problema de decizie P este **NP-dificilă** dacă $\forall P' \in \mathbf{NP}$ are loc $P' \propto P$.

Problema de decizie P este **NP-completă** dacă $P \in \mathbf{NP}$ și P este NP-dificilă.

Folosind observația 1 și faptul că $\mathbf{P} \subseteq \mathbf{NP}$ rezultă următoarea

Teoremă *Dacă P o problemă oarecare NP-completă satisface $P \in \mathbf{P}$ atunci $\mathbf{P} = \mathbf{NP}$.*

Rezultă că problemele NP-complete formează submulțimea lui **NP** care conține cele mai dificile probleme. Dacă găsim un algoritm polinomial pentru una dintre ele, putem construi un algoritm polinomial pentru oricare altă problemă din **NP**. Din păcate, deși nu există o demonstrație, oamenii cred că, de fapt, aceste probleme nu admit algoritmi polinomiali de rezolvare. Diagrama următoare sugerează relațiile care se crede că există între aceste clase de complexitate:



Se obișnuiește să se spună că o problemă de optimizare este **NP**-dificilă, dacă problema de decizie asociată este așa.

Pentru ca să ne aliniem la definițiile precedente, vom spune că o problemă oarecare P (nu neaparat de decizie) este **NP**-dificilă dacă existența unui algoritm polinomial pentru P implică $P = NP$.

Comentariu metaforic: Saying that a problem is NP-hard is like saying ‘If I own a dog, then it can speak fluent English.’ You probably don’t know whether or not I own a dog, but you’re probably pretty sure that I don’t own a talking dog. Nobody has a mathematical proof that dogs can’t speak English. Nevertheless, no sane person would believe me if I said I owned a dog that spoke fluent English. So the statement ‘If I own a dog, then it can speak fluent English’ has a natural corollary: No one in their right mind should believe that I own a dog! Likewise, if a problem is NP-hard, no one in their right mind should believe it can be solved in polynomial time. [Jeff Erickson, Univ. of Illinois]

Singura clasă de complexitate pentru care nu am sugerat un exemplu este cea a problemelor **NP**–complete. Primul om care a demonstrat existența unei astfel de probleme este **Cook**, care în 1971 a arătat că **SAT** \in **NP**, unde

SAT

Instanță: $U = \{u_1, \dots, u_n\}$ o mulțime finită de variabile booleene.

$C = C_1 \wedge C_2 \wedge \dots \wedge C_m$ o formulă în formă conjunctivă peste U :

$C_i = v_{i_1} \vee v_{i_2} \vee \dots \vee v_{i_{k_i}} \quad \forall i = \overline{1, m}$, unde

$\forall i_j \quad \exists \alpha \in \{1, \dots, n\}$ astfel încât

$v_{i_j} = u_\alpha$ sau $v_{i_j} = \overline{u_\alpha}$.

Intrebare: Există o atribuire $t : U \rightarrow \{A, F\}$ a. î.
 $t(C) = A$?

Evaluarea lui $t(C)$ se bazează pe formulele uzuale din calculul boolean:

$$A \wedge A = A, \quad F \wedge A = F \wedge F = A \wedge F = F,$$

$$F \vee F = F, \quad F \vee A = A \vee F = A \vee A = A,$$

$$\overline{\overline{F}} = A, \quad \overline{\overline{A}} = F, \quad (F) = F, (A) = A,$$

și e clar că se poate face în timp polinomial.

Apartenența lui SAT la NP e clară, un certificat pentru răspunsul da este o atribuire t_0 care poate fi verificată în timp polinomial.

O demonstrație că orice limbaj din NP se reduce polinomial la SAT necesită o abordare formală noțiunilor de algoritm și timp de execuție (pe care le-am considerat aici la nivel intuitiv) și o omitem.

$3SAT$ este restricția lui SAT în care fiecare clauză C_i are exact trei **literali** ($k_i = 3$), un literal $v_{i,j}$ fiind, așa cum este descris mai sus, o variabilă sau negația ei.

Se poate arăta ușor că $SAT \propto 3SAT$ și deci se obține că **$3SAT$ este NP -completă** (apartenența la NP e clară fiind o restricție a lui SAT care e din NP , iar tranzitivitatea relației \propto împreună cu teorema lui Cook termină demonstrația).

Schița de demonstrație de mai sus este uzuală în demonstrațiile de **NP**-completitudine și o explicăm:

Pentru a arăta că o problemă de decizie P este **NP**-completă se procedează astfel:

1. Se arată că $L = P^{-1}(da)$ satisface $L \in \mathbf{NP}$.
2. Se selectează un limbaj L' despre care știm că este **NP**-complet.
3. Se încearcă construirea unui algoritm de reducere F de la L' la L .
4. Se demonstrează că F e algoritm de reducere.
5. Se arată că F este algoritm polinomial.

Pentru pasul 2 se poate consulta

<http://www.nada.kth.se/~viggo/problemlist/>

I. Vocabular al Teoriei grafurilor

1. Definiția unui graf

Un **graf** este o pereche $G = (V(G), E(G))$, unde

- $V(G)$ este o mulțime finită nevidă, iar
- $E(G)$ este o submulțime a mulțimii $\mathcal{P}_2(V(G))$ a părților cu două elemente ale lui $V(G)$.

$V(G)$ se numește **mulțimea vîrfurilor** grafului G și numărul elementelor sale

$|V(G)|$, este **ordinul** grafului G ;

$E(G)$ este **mulțimea muchiilor** grafului G și numărul său de elemente, $|E(G)|$, este **dimensiunea** grafului G .

Atunci cînd nu există posibilitatea confuziilor, vom nota simplu, $G = (V, E)$.

Dacă $e = \{u, v\} \in E(G)$ este o muchie a grafului G vom nota $e = uv$ (pentru simplificarea scrierii) și vom spune că:

- muchia e **unește** vîrfurile u și v ;
- vîrfurile u și v sunt **adiacente** în G ;
- muchia e este **incidentă** cu vîrfurile u și v ;
- vîrfurile u și v sunt **vecine** în G ;
- vîrf. u și v sunt **extremitățile** muchiei e .

Dacă $v \in V(G)$, atunci mulțimea

$$N_G(v) = \{w | w \in V(G), vw \in E(G)\}$$

se numește **vecinătatea** vîrfului v în G .

Se mai notează $N_G(v) = \Gamma_G(v)$.

Remarcăm faptul că graful G poate fi definit (în mod echivalent) ca o pereche $(V(G), \Gamma_G)$ unde,

$$\Gamma_G: V(G) \rightarrow \mathcal{P}(V(G))$$

asociază fiecărui vîrf vecinătatea sa, și satisfăcând condițiile: $\forall v \in V(G) : v \notin \Gamma_G(v)$ și $\forall u, v \in V(G) : u \in \Gamma_G(v) \Leftrightarrow v \in \Gamma_G(u)$.

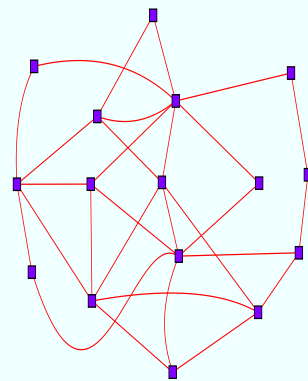
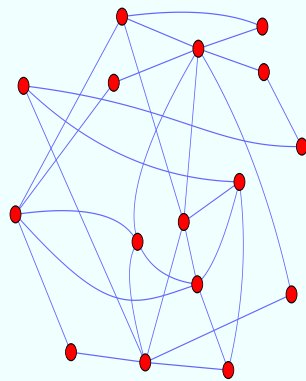
Două muchii e și e' care au o extremitate comună se numesc **adiacente**.

Intuitiv, un graf $G = (V(G), E(G))$ poate fi reprezentat (după cum sugerează și numele său) cu ajutorul unei figuri plane formată dintr-o mulțime de **mici forme geometrice** aflată în corespondență cu mulțimea de vîrfuri $V(G)$, două forme fiind unite printr-o **curbă simplă** dacă și numai dacă, perechea de vîrfuri corespunzătoare lor este o muchie a grafului G .

Corespondența dintre vîrfurile grafului și figurile geometrice considerate este vizualizată

uneori cu etichete atașate vârfurilor. De asemenea, muchiile pot fi etichetate. În plus, sunt utilizate diferite atribute grafice pentru expresivitatea desenului, diagramei.

De exemplu,



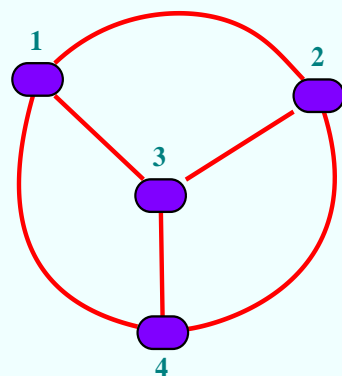
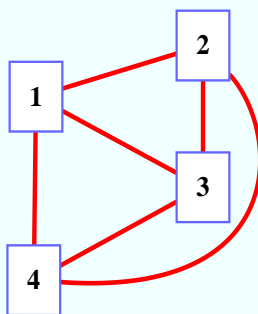
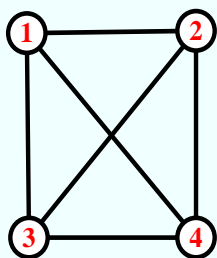
figura

și

reprezintă același graf, deși lipsa etichetelor face dificilă realizarea acestui fapt.

Următoarele trei reprezentări ale grafului

$G = (\{1, 2, 3, 4\}, \{12, 13, 14, 23, 24, 34\})$ sunt mult mai clare:



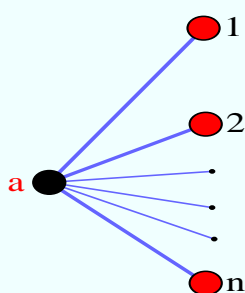
O **mulțime independentă de vîrfuri** (sau **mulțime stabilă**) în G este o mulțime $S \subseteq V(G)$ de vîrfuri cu proprietatea că

$$\mathcal{P}_2(S) \cap E(G) = \emptyset$$

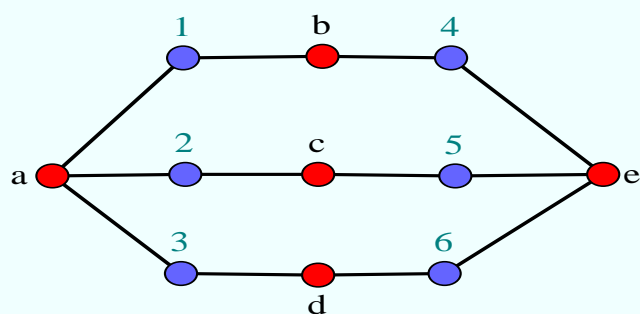
(adică o mulțime de vîrfuri neadiacente două cîte două).

Cardinalul maxim al unei mulțimi stabile se numește **numărul de stabilitate** sau **numărul de independență** al grafului G și se notează cu $\alpha(G)$.

De exemplu, în graful G de mai jos mulțimea $\{a\}$ este mulțime stabilă (maximală în raport cu incluziunea), dar numărul de stabilitate este n , mulțimea $\{1, \dots, n\}$ fiind stabilă de cardinal maxim ($n \geq 1$). În graful H s-au evidențiat două mulțimi stabile care partiționează mulțimea vîrfurilor, iar $\alpha(H) = 6$.



G



H

Problema următoare este naturală, ușor de formulat și apare deseori în diferite aplicații:

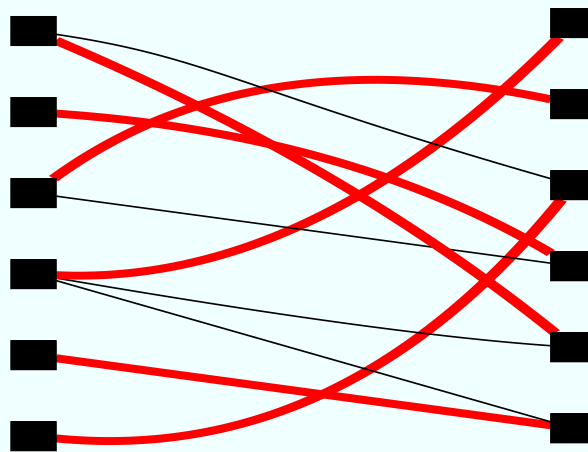
P1 Intrare: G un graf.
Ieșire: $\alpha(G)$ și un "martor":
 S m.stabilă în G , cu $|S| = \alpha(G)$.

Deși foarte simplă (de formulat, la o primă vedere), problema este NP-dificilă. Problema de decizie corespunzătoare,

SM Intrare: G un graf, $k \in \mathbb{N}$.
Intrebare: Există S m.stabilă în G ,
 cu $|S| \geq k$?

este NP-completă (Karp, 1972). Probabil că o cauză a dificultății acestei probleme este faptul că două mulțimi stabile maximale (în raport cu incluziunea) pot avea raportul cardinalelor oricât de mare (vezi graful G din figura de la pagina precedentă).

O **mulțime independentă de muchii** sau **cuplaj** în graful G este o mulțime de muchii neadiacente două câte două . Cardinalul maxim al unei mulțimi independente de muchii în G se numește **numărul de muchie-independență** al grafului G și se notează $\nu(G)$.
Exemplu, pentru graful G :



numărul de muchie independentă, $\nu(G)$ este 6, un cuplaj cu acest număr de muchii fiind pus în evidență. Problema

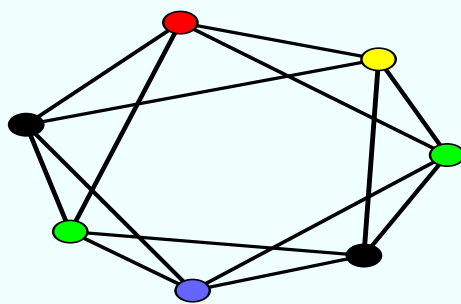
P2 Intrare: G un graf.
 Ieșire: $\nu(G)$ și un "martor":
 M cuplaj în G , cu $|M| = \nu(G)$.

este foarte asemănătoare cu $P1$ (este de fapt, o restricție a lui $P1$ pentru intrări care sunt line-grafuri) și totuși s-a arătat că este în \mathbf{P} (Edmons, 1965).

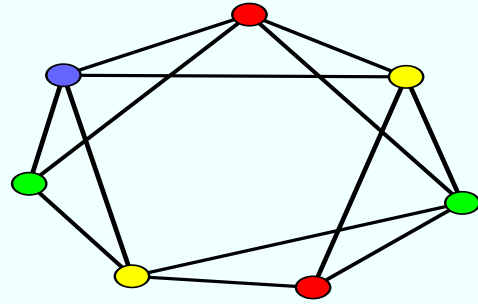
Diferența de complexitate provine, probabil, din faptul că raportul dintre cardinalele a două cuplaje maximale în raport cu incluziunea nu poate depăși 2.

Dacă $G = (V(G), E(G))$ este un graf și $p \in \mathbf{N}^*$, se numește p -**colorare a (vîrfurilor)** lui G o aplicație $c : V(G) \rightarrow \{1, \dots, p\}$ cu proprietatea că $c^{-1}(i)$ este o mulțime stabilă în G , $\forall i \in \{1, \dots, p\}$ (remarcăm că, din definiția mulțimilor stabile, \emptyset este o mulțime stabilă). **Numărul cromatic** al grafului G , notat $\chi(G)$, este cea mai mică valoare a lui $p \in \mathbf{N}^*$ pentru care G admite o p -colorare.

Exemplu: În graful G desenat mai jos, sunt evidențiate 2 colorări una cu 5 culori și una cu 4 culori. Se poate argumenta că $\chi(G) = 4$.



5-colorare



4-colorare

rosu= culoarea 1
galben= culoarea 2
verde=culoarea 3
albastru=culoarea 4
negru=culoarea 5

Problema următoare apare în diverse situații practice (de exemplu în problemele de orar, sau în problemele de acoperire din wireless networks):

P3 Intrare: G un graf.
 Ieșire: $\chi(G)$ și un "martor":
 o $\chi(G)$ -colorare a lui G .

Din problema P1 știm că mulțimile stabile ale grafului sunt greu de "stăpânit", și cum problema P3 cere de fapt să partiționăm mulțimea de vârfuri a grafului într-un număr cât mai mic de mulțimi stabile, este normal ca și această problemă să fie NP-dificilă. Problema de decizie

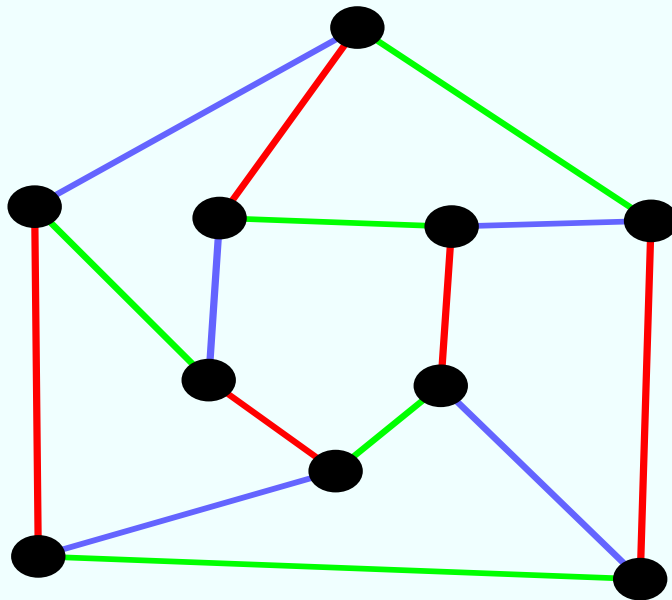
COL Intrare: G un graf, $k \in \mathbb{N}$.
Intrebare: Admite G o k -colorare?

este NP-completă chiar dacă o restricționăm la cazul particular $k = 3$. Există însă restricții ale problemei pentru care avem apartenență la **P** (de exemplu, dacă G este un graf perfect).

O p -**colorare a muchiilor** lui G este o aplicație $c : E(G) \rightarrow \{1, \dots, p\}$ cu proprietatea că $c^{-1}(i)$ este un cuplaj al lui G , $\forall i \in \{1, \dots, p\}$.

Indicele cromatic al grafului G , notat $\chi'(G)$, este cea mai mică valoare a lui $p \in \mathbb{N}^*$ pentru care G admite o p -colorare a muchiilor.

Exemplu: In graful de mai jos



este evidențiată o 3-colorare a muchiiilor; este clar că este și optimă, întrucât muchiile incidente în același vârf trebuie să aibă culori distincte. Problema

P4 Intrare: G un graf.
Ieșire: $\chi'(G)$ și un "martor":
o $\chi'(G)$ -colorare a muchiiilor lui G .

s-a dovedit a fi NP-dificilă, deși era de așteptat ca (lucrând cu cuplaje, iar problema P2 fiind din **P**) ca ea să fie rezolvabilă polinomial.

Diferența față de P3 (P4 este de fapt o restricție a lui P3 pe clasa line-grafurilor) este că în timp ce P3 s-a demonstrat că nu poate fi ușor aproximabilă (în timp polinomial), problema P4 poate fi rezolvată "aproximativ" cu o eroare de o unitate (deci dacă se greșeste, atunci colorarea obținută are cel mult o culoare în plus față de cea optimă).

Două grafuri, $G = (V(G), E(G))$ și $H = (V(H), E(H))$ se numesc **izomorfe**, și notăm aceasta prin $G \cong H$, dacă există o bijecție

$$\varphi: V(G) \rightarrow V(H)$$

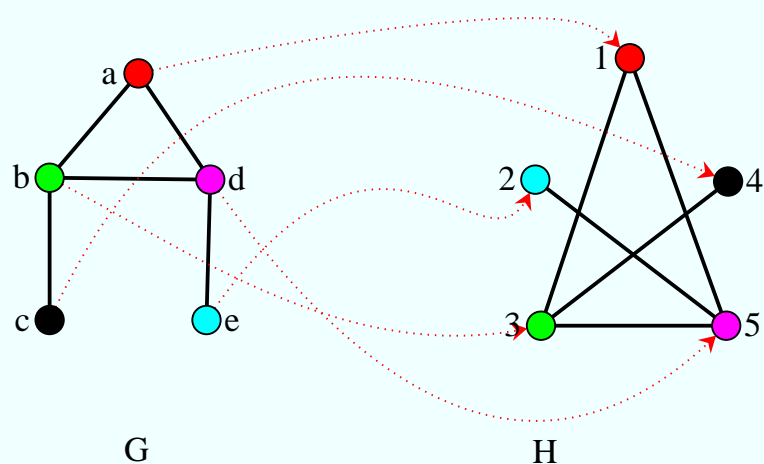
cu proprietatea că aplicația

$$\psi: E(G) \rightarrow E(H),$$

definită pentru orice $uv \in E(G)$ prin $\psi(uv) = \varphi(u)\varphi(v)$ este o bijecție.

(deci, două grafuri sînt izomorfe dacă există o bijecție între mulțimile lor de vîrfuri care induce o bijecție între mulțimile lor de muchii).

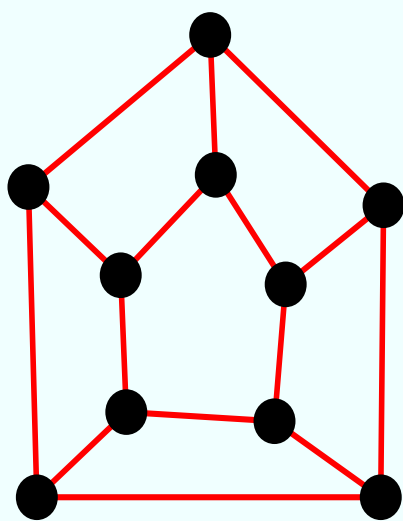
Grafurile următoare sunt izomorfe, așa cum se sugerează în figură



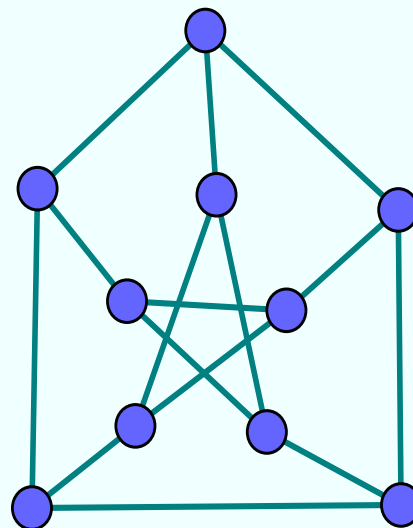
Problema următoare este utilă în multe probleme de modelare discretă (de exemplu în chimie)

ISO Intrare: G, H grafuri.
 Intrebare: $G \cong H?$

Nu s-a demonstrat dacă această problemă este sau nu NP-completă (apartenența la NP este clară). Se pare că face parte dintr-o clasă de probleme aflată între **P** și **NP**). Exemplul următor arată dificultatea problemei (cele două grafuri au același ordin, aceeași dimensiune, vârfurile au același număr de muchii incidente, și totuși ele nu-s izomorfe: în primul apar circuite de lungime 4, iar în al doilea nu !)



G



H

Dacă $G = (V(G), E(G))$ este un graf, un **auto-morfism** al lui G este o permutare a lui $V(G)$

($\varphi: V(G) \rightarrow V(G)$, φ bijectivă) cu proprietatea că induce o permutare a lui $E(G)$

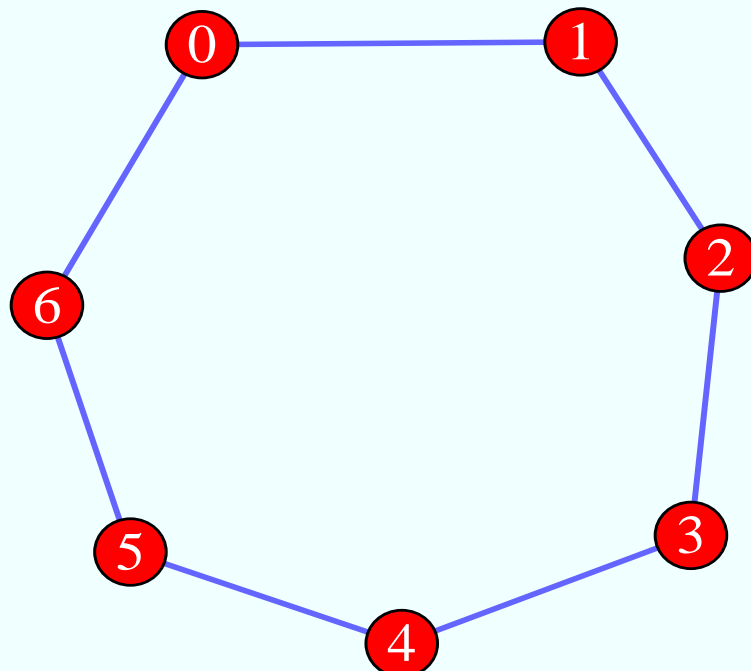
($\psi: E(G) \rightarrow E(G)$, $\psi(uv) = \varphi(u)\varphi(v)$, $\forall uv \in E(G)$, este bijectivă).

Mulțimea automorfismelor grafului G formează, în raport cu operația de compunere a aplicațiilor, un grup numit **grupul automorfismelor grafului** G , notat $Aut(G)$.

$Aut(G)$ este **tranzitiv** dacă

$$\forall v \in V(G), \{w \mid \exists \varphi \in Aut(G) : \varphi(v) = w\} = V(G)$$

Exemplu:

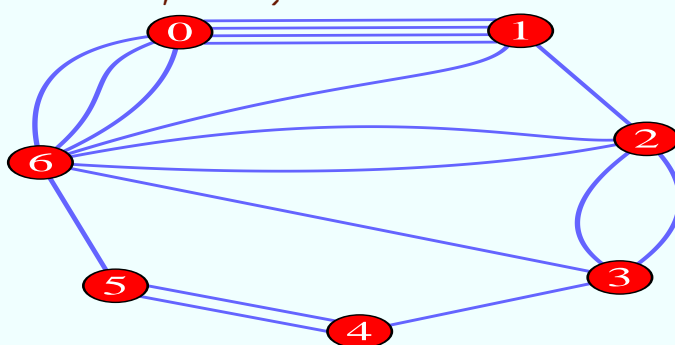


2. Variații în definiția unui graf

a) Dacă în definiția unui graf, se consideră $E(G)$ o **multimulțime** pe $\mathcal{P}_2(V(G))$, adică este dată o funcție $m: \mathcal{P}_2(V(G)) \rightarrow \mathbb{N}$, se obține noțiunea de **multigraf**.

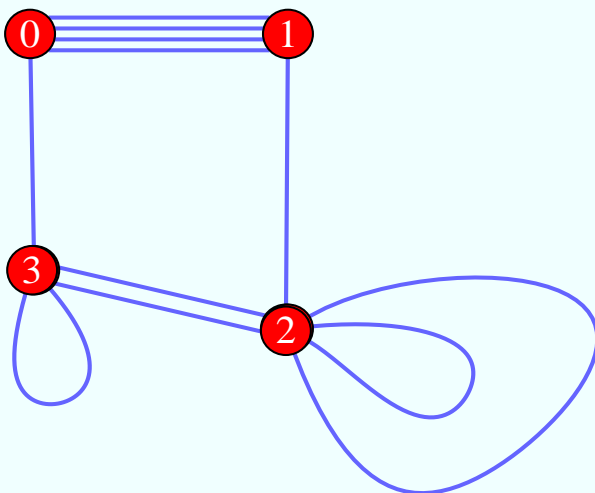
Un element $e \in \mathcal{P}_2(V(G))$ cu $m(e) > 0$ este muchie a multigrafului, **simplă** dacă $m(e) = 1$, **multiplă** dacă $m(e) > 1$.

Oricărui multigraf M i se poate asocia un graf $G(M)$, numit **graful suport** al lui M , obținut prin înlocuirea fiecărei muchii multiple cu o singură muchie cu aceleași extremități. Pictural, modificările de reprezentare sunt evidente; graful suport al multigrafului desenat mai jos, este graful desenat pe pagina precedentă la care se adauga muchiile 61, 62 și 63.

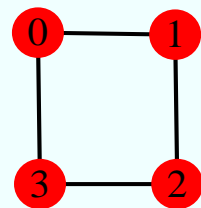


b) Dacă în definiția unui graf se consideră $E(G)$ ca o multimulțime pe mulțimea părților nevide cu cel mult două elemente ale lui $V(G)$, atunci G se numește **graf general** sau **pseudograf**. O muchie $e \in E(G)$, $e = \{v\}$ se numește **buclă** în vârful v .

Exemplul următor arată un graf general M și graful său suport.



M



$G(M)$

Pentru evitarea confuziilor, uneori grafurile – așa cum le-am definit– se mai numesc și *grafuri simple*.

c) Un **digraf** este o pereche $D = (V(D), A(D))$ unde $V(D)$ este o mulțime finită nevidă (mulțimea vîrfurilor digrafului D), iar $A(D) \subseteq V(D) \times V(D)$ este mulțimea **arcelor** digrafului D .

Dacă $a = (u, v)$ este arc în D , notăm $a = uv$ și spunem că

u și v sînt **adiacente**;

a este **incident din** u ;

a este **incident spre** v ;

u **domină** pe v ;

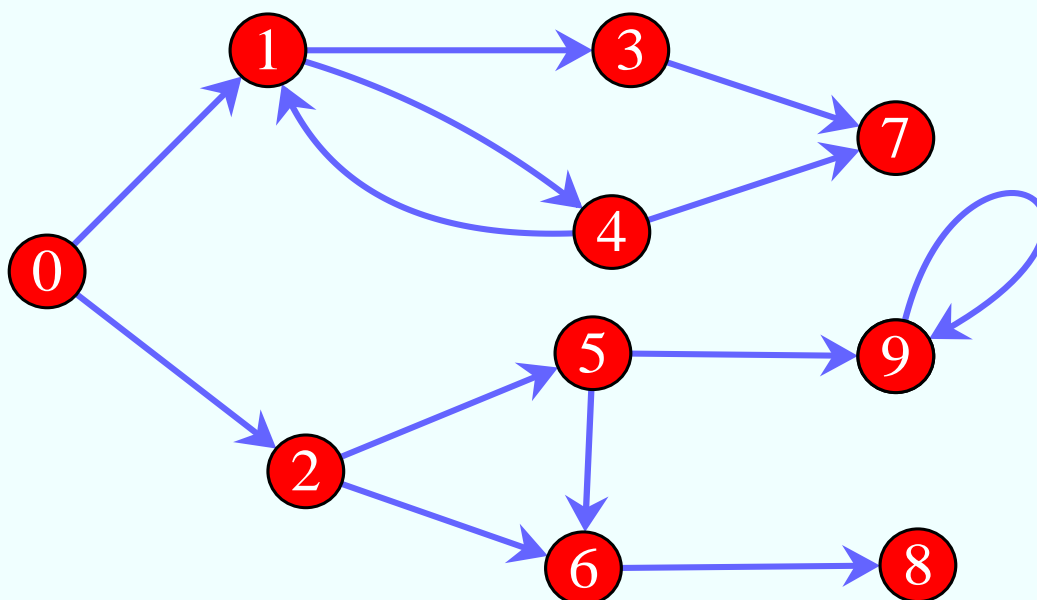
a este **incident** cu u **spre exterior**;

a este **incident** cu v **spre interior**;

u este **extremitatea inițială** a lui a și v este **extremitatea finală** a lui a .

Pictural, digrafurile se reprezintă la fel ca și grafurile, adăugînd curbei ce unește două figuri asociate vîrfurilor o săgeată pentru a preciza **perechea** de vîrfuri corespunzătoare arcu-lui desenat.

Exemplu:



O pereche de arce de forma vw și wv se numește **pereche simetrică** de arce.

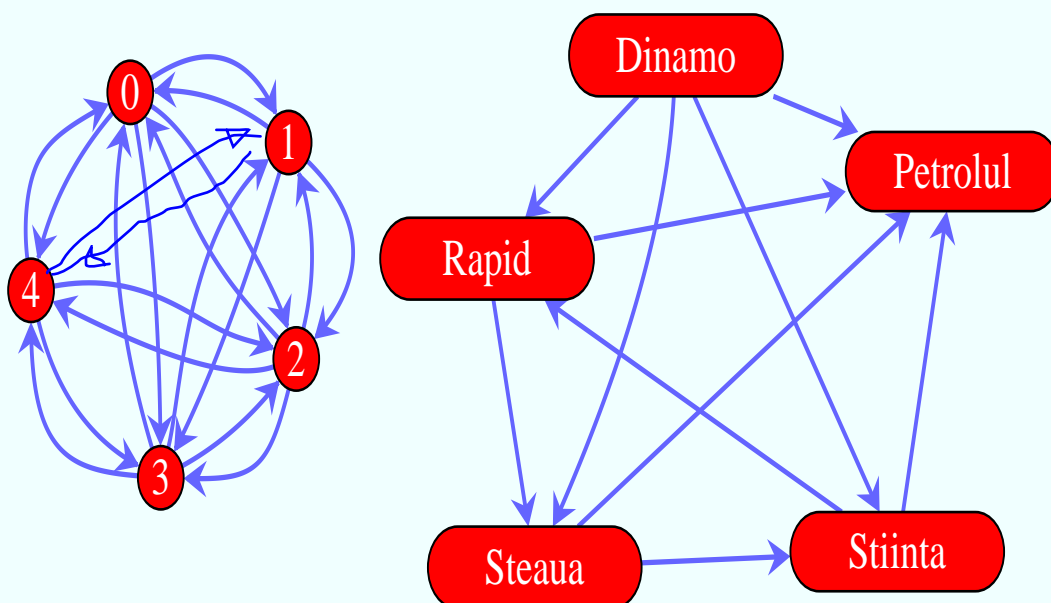
Dacă D este un digraf, **inversul** său D' este digraful obținut din D prin înlocuirea fiecărui arc vw cu opusul său wv .

Dacă D este un digraf, atunci înlocuind fiecare arc cu mulțimea de vîrfuri care îl formează, obținem, în general, un graf general $M(D)$. Graful suport al acestuia se numește **graful suport** al digrafului D .

Dacă $M(D)$ este graf atunci D se numește **graf orientat** (poate fi gândit ca obținut prin "orientarea" fiecărei muchii a grafului $M(D)$).

Un **digraf complet simetric** este un digraf în care fiecare pereche de vîrfuri este unită prin exact o pereche de arce simetrice.

Un **turneu** este un digraf în care orice două vîrfuri sînt unite prin exact un arc.



d) Grafurile **infinite** se obțin prin înlăturarea condiției de finitudine a mulțimii de vârfuri și (sau) muchii. Acestea se consideră a fi numărabile, iar tratarea lor utilizează instrumente care nu sunt neaparat combinatorii (de exemplu, mecanisme generative). Un graf G **local finit** este un graf infinit în care $N_G(v)$ este finită pentru orice vârf v .

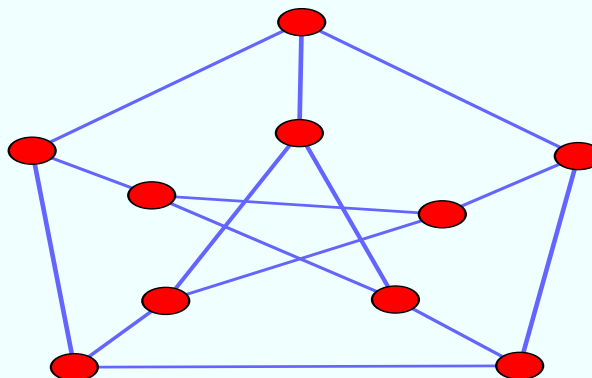
e) **Hipergrafurile** se obțin renunțând la condiția că muchiile pot avea cel mult două vârfuri, (se obțin astfel **hipermuchiile**). Ele se mai numesc **sisteme finite de mulțimi** și vom arăta că pot fi studiate via grafurile bipartite, deși există rezultate combinatorii importante și cu aplicații directe (de exemplu în bazele de date) în formalismul care urmează extinderea tratării grafurilor (din punct de vedere combinatoriu sau algebric).

3. Grade

Dacă $G = (V, E)$ este un graf și $v \in V$ un vîrf al său, atunci **valența** sau **gradul** lui v în G , notat $d_G(v)$ sau $\rho_G(v)$ este

$$|\{e \mid e \in E, e \text{ incidentă cu } v\}|.$$

Un vîrf de grad 0 se numește **izolat**; un vîrf de grad 1 se numește **pendant**. Dacă toate vîrfurile lui G au aceeași valență ρ atunci G se numește **graf ρ -valent** sau **ρ -regulat**. Un graf 0-valent se numește **graf nul**. Un graf 3-valent se numește **graf trivalent** sau **cu-bic**. Un exemplu de graf trivalent este graful lui *Petersen*:

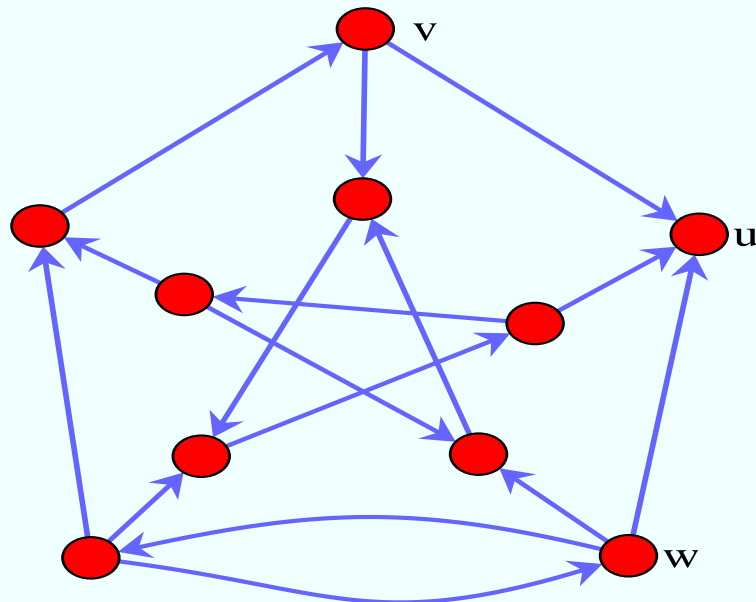


Gradul maxim al unui vîrf al grafului G se notează cu $\Delta(G)$, iar gradul minim $\delta(G)$.

Concepte analoage se pot defini și pentru digrafuri. Dacă v este un vîrf al digrafului D atunci **valența interioară** sau **gradul interior** notat $\rho_{in}(v)$ sau $\rho_D^-(v)$ sau $d_D^-(v)$, este numărul arcelor incidente cu v spre interior; **valența exterioară** sau **gradul exterior** notat $\rho_{out}(v)$ sau $\rho_D^+(v)$ sau $d_D^+(v)$, este numărul arcelor incidente cu v spre exterior.

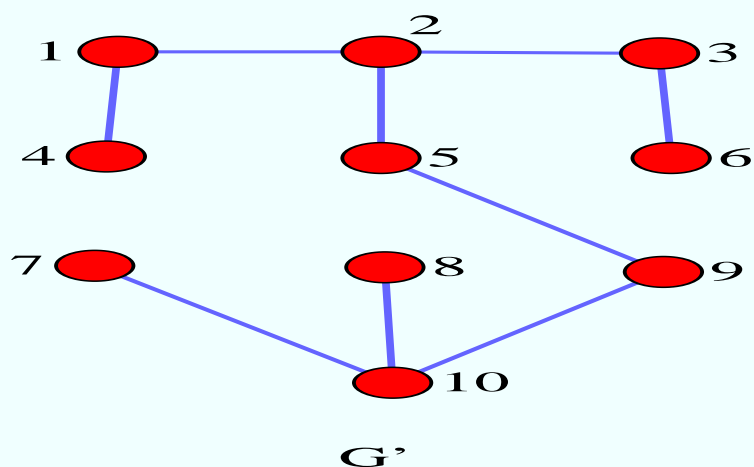
De exemplu, în digraful D desenat mai jos avem

$$\begin{aligned} d_D^-(v) &= 1, \quad d_D^+(v) = 2; \\ d_D^-(u) &= 3, \quad d_D^+(u) = 0; \\ d_D^-(w) &= 1, \quad d_D^+(w) = 3; \end{aligned}$$



Subgraful $[V(G) \setminus A]_G$ se notează $G - A$ și este **subgraful** lui G **obținut prin îndepărtarea vîrfurilor din** A ; în particular, dacă $A = \{v\}$, atunci $G - \{v\}$ se notează $G - v$.

Dacă $E' \subseteq E(G)$ atunci $\langle E' \rangle_G = (V(G), E')$ este **graful parțial secționat de** E' **în** G . $G - E'$ este prin definiție $\langle E(G) \setminus E' \rangle_G$, iar $G - e = G - \{e\}$ ($e \in E(G)$). Pentru G graful din figura precedentă și $E' = \{12, 14, 23, 25, 36, 59, 710, 810, 910\}$, obținem că $\langle E' \rangle_G$ este graful G' :

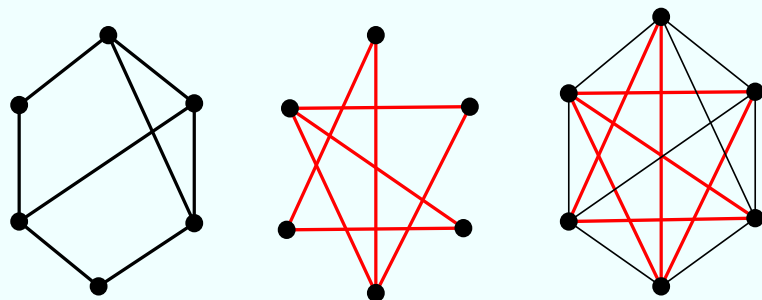


Concepte similare se pot defini în mod analog pentru multigrafuri, grafuri generale sau digrafuri.

5. Operații cu grafuri

Dacă $G = (V(G), E(G))$ este un graf, atunci :

-**complementarul** său este graful \overline{G} cu
 $V(\overline{G}) = V(G)$ și $E(\overline{G}) = \mathcal{P}_2(V(G)) \setminus E(G)$.

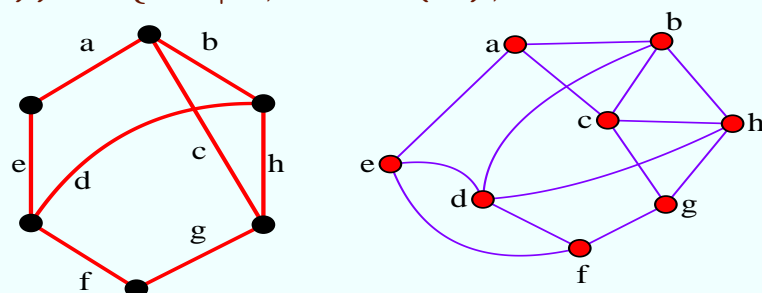


Graful initial

Complementarul

Graful complet

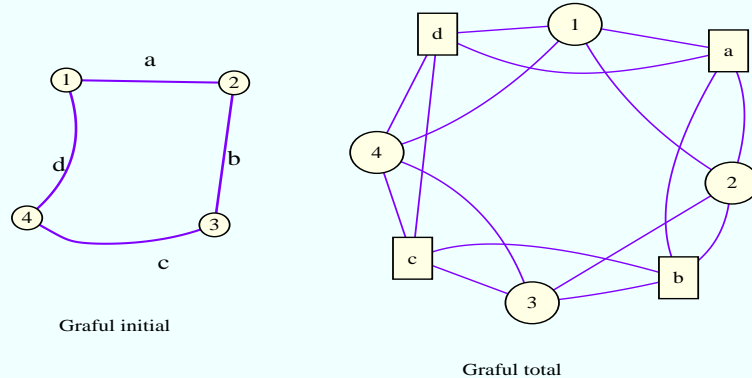
-**graful reprezentativ al muchiilor lui G** este
 graful $L(G)$ cu $V(L(G)) = E(G)$ și
 $E(L(G)) = \{ee' \mid e, e' \in E(G), e \text{ și } e' \text{ adiacente în } G\}$.



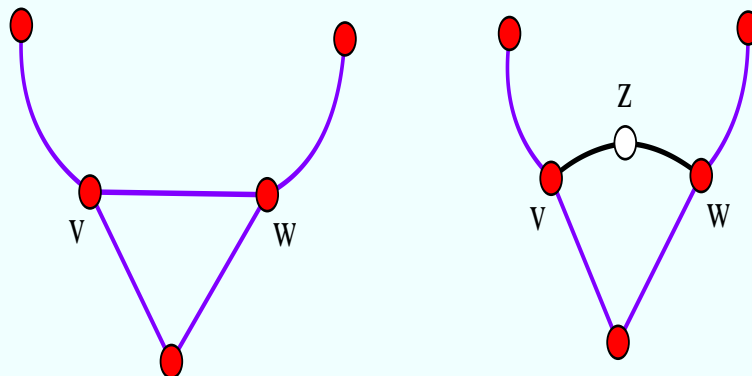
Graful initial

Line-graful sau

- graful total** al grafului G este graful $T(G)$ cu $V(T(G)) = V(G) \cup E(G)$ și $E(T(G)) = \{xy | x, y \in V(G) \cup E(G), x \text{ și } y \text{ adiacente sau incidente în } G\}$.



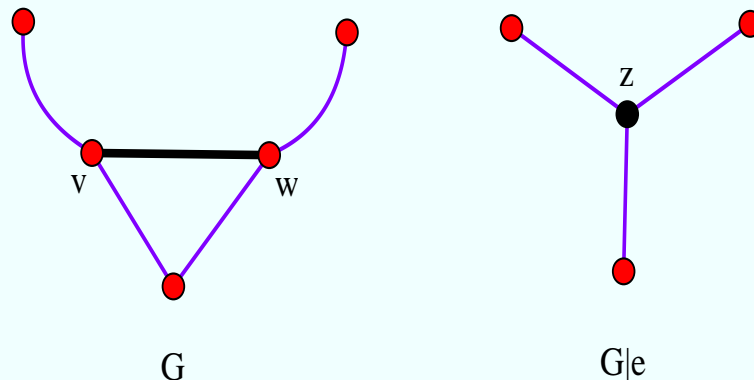
- graful obținut din G prin inserția unui vârf** (z) pe o muchie ($e = vw$) este graful $G' = (V(G) \cup \{z\}, E(G) \setminus \{vw\} \cup \{vz, zw\})$ ($z \notin V(G), e \in E(G)$).



Două grafuri obținute prin inserții succesive de vârfuri pe muchiile aceluiași graf se numesc **homeomorfe**.

-**graful obținut din G prin contracția muchiei**

$e = vw \in E(G)$ este graful $G|e = (V(G) \setminus \{v, w\} \cup \{z\}, E([V(G) \setminus \{v, w\}]_G) \cup \{yz \mid yv \text{ sau } yw \in E(G)\})$.



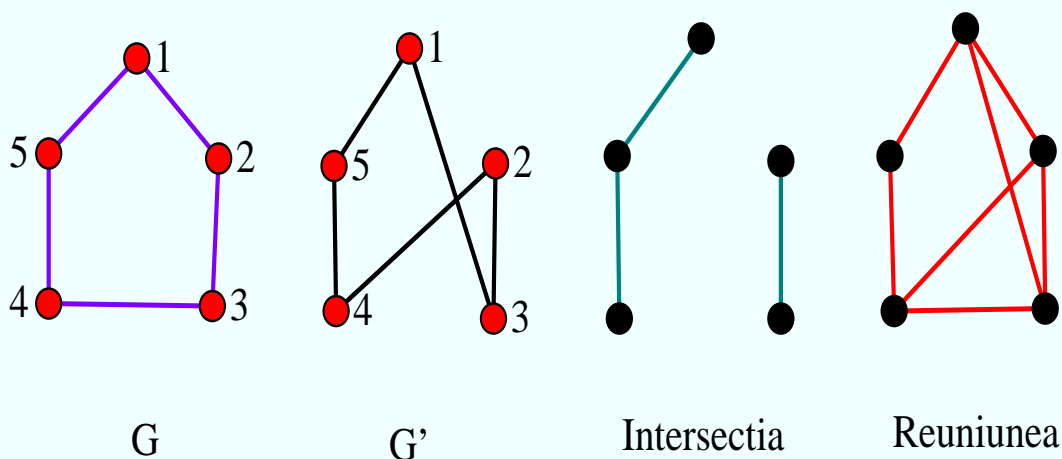
Dacă H se poate obține prin contracții succesive de muchii din graful G , se spune că G **este contractibil la H** .

Fie $G = (V(G), E(G))$ și $G' = (V(G'), E(G'))$ două grafuri.

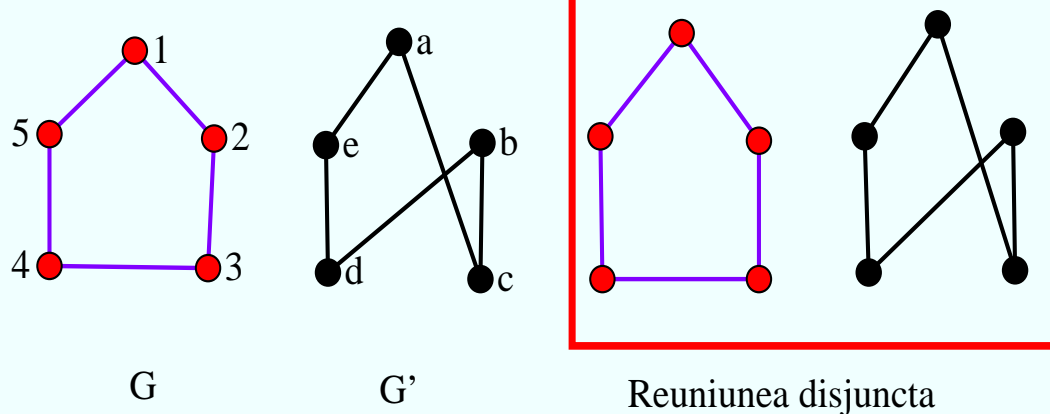
- Dacă $V(G) = V(G')$ atunci **reuniunea** celor două grafuri și **intersecția** lor se definesc

$$G \cup G' = (V(G), E(G) \cup E(G')),$$

$$G \cap G' = (V(G), E(G) \cap E(G')).$$

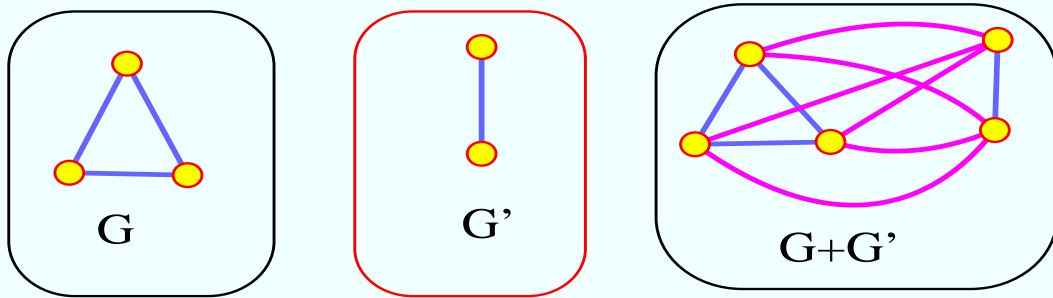


-Dacă $V(G) \cap V(G') = \emptyset$ atunci $G \cup G' = (V(G) \cup V(G'), E(G) \cup E(G'))$ se numește **reuniunea disjunctă** a grafurilor G și G' . Reuniunea disjunctă a k grafuri izomorfe cu G se notează kG .



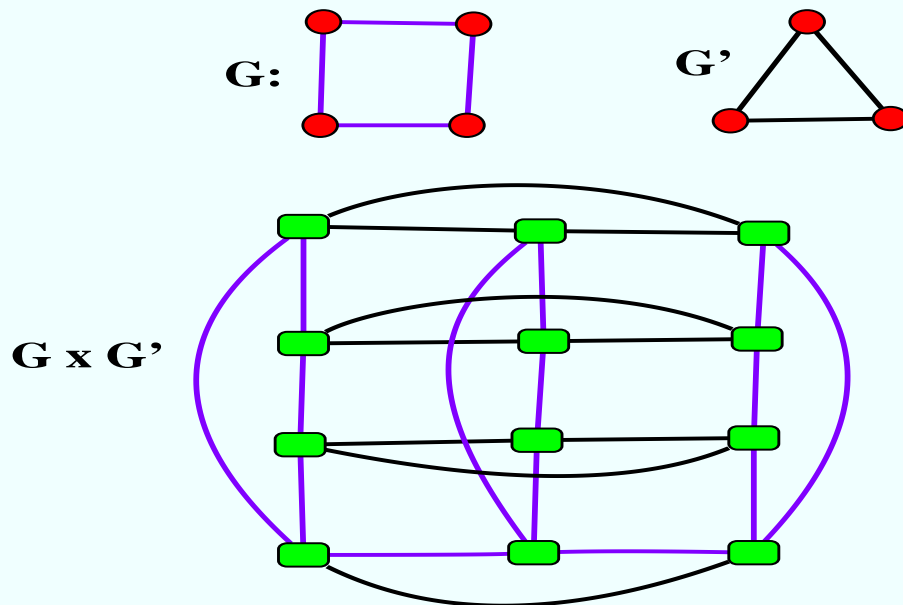
-**Suma** a două grafuri G și G' este graful

$$G + G' = \overline{G \cup G'}.$$



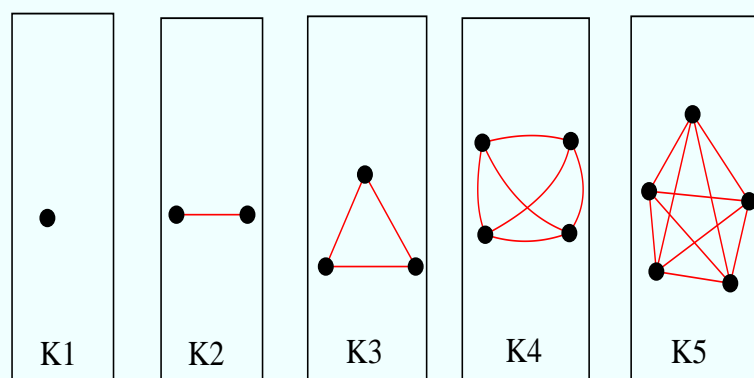
-**Produsul cartezian** al grafurilor G și G' este graful $G \times G'$ cu $V(G \times G') = V(G) \times V(G')$ și

$$E(G \times G') = \{(v, w)(v', w') \mid v, v' \in V(G), w, w' \in V(G') \text{ și } w = w' \text{ și } ww' \in E(G') \text{ sau } w = w' \text{ și } vv' \in E(G)\}$$

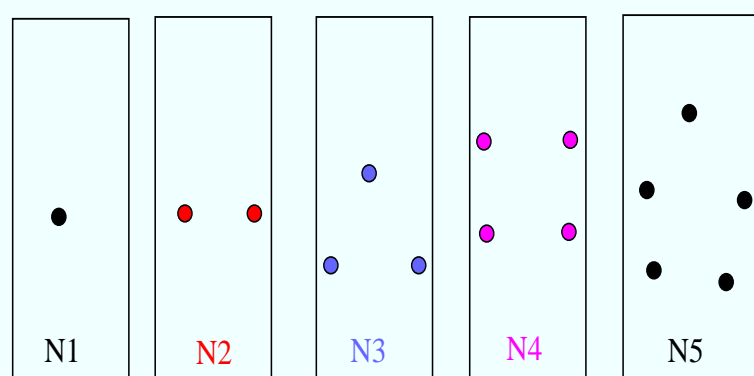


6. Clase de grafuri

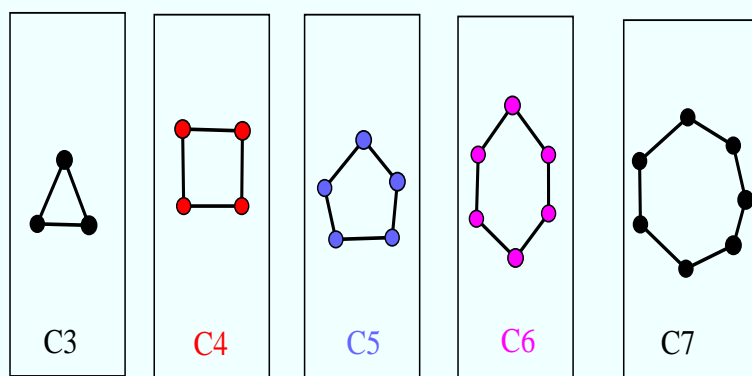
Graful complet de ordin n : K_n
 cu $|V(K_n)| = n$ și $E(K_n) = \mathcal{P}_2(V(K_n))$.



Graful nul de ordin n : $N_n = \overline{K_n}$.



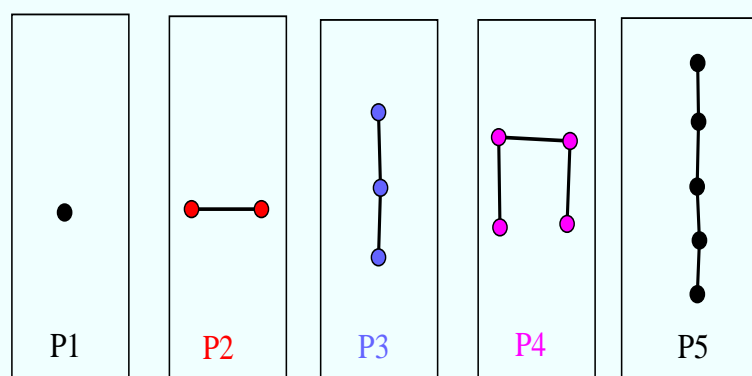
Circuitul de ordin n ($n \geq 3$) : C_n
 cu $V(C_n) = \{1, \dots, n\}$ și
 $E(C_n) = \{12, 23, \dots, n-1n, n1\}$.



Drumul de ordin n : P_n

$P_1 = K_1, P_2 = K_2;$

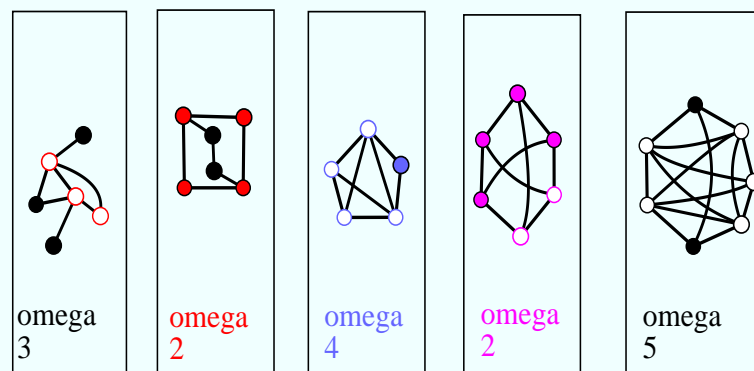
$n \geq 3 : P_n = C_n - e \quad (e \in E(C_n)).$



Un subgraf complet (de ordin q) al unui graf G se numește **clică** (q -**clică**) a lui G .

Cardinalul maxim al unei clici a lui G se numește **numărul de clică** sau **numărul de densitate** al lui G și se notează $\omega(G)$. Cum, evident $\omega(G) = \alpha(\overline{G})$, rezultă că determinarea numărului de clică al unui graf și a unei clici de cardinal maxim este problema **P1** cu intrarea \overline{G} .

Exemple:

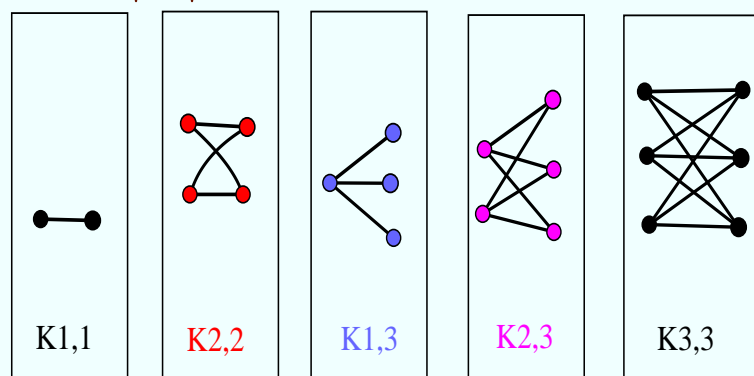


Un **graf bipartit** este un graf G cu proprietatea că $V(G)$ se poate partiționa în două mulțimi independente în G .

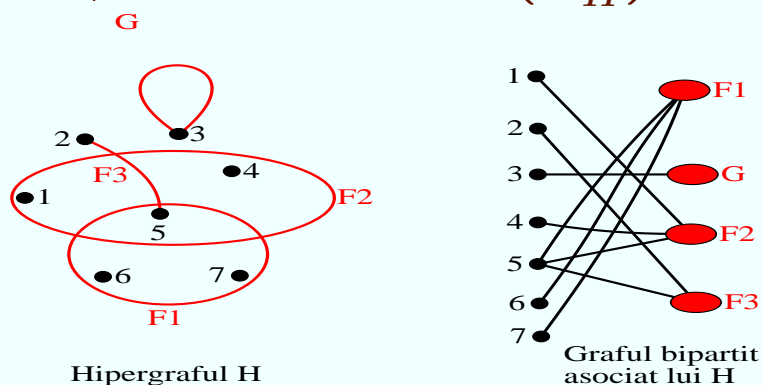
Dacă S și T satisfac $S \cup T = V(G)$, $S \cap T = \emptyset$ și S, T sînt independente și nevide în G , atunci graful bipartit G se notează $G = (S, T; E(G))$.

Deci, dacă $G = (S, T; E(G))$ este un graf bipartit, atunci $\forall e \in E(G)$ are o extremitate în S și cealaltă în T .

Dacă $\forall v \in S$ și $\forall w \in T$ $vw \in E(G)$, atunci graful bipartit $G = (S, T; E(G))$ se numește **graf bipartit complet** și se notează $K_{s,t}$ unde $s = |S|$ și $t = |T|$.

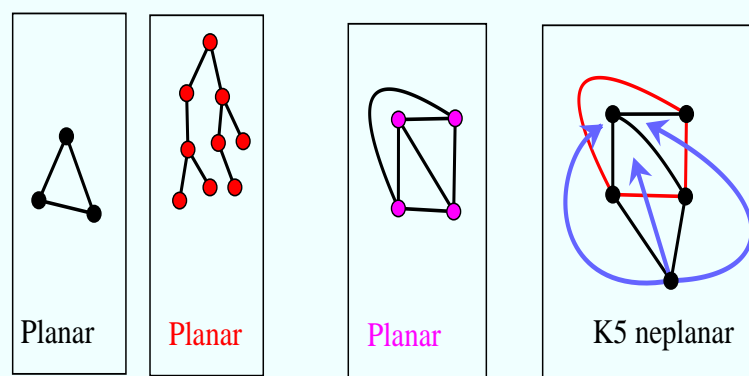


Pentru orice hipergraf $H = (V, \mathcal{E})$, se poate asocia un graf bipartit $G_H = (V, \mathcal{E}; E(G_H))$, unde $\forall v \in V, \forall F \in \mathcal{E} \quad vF \in E(G_H) \Leftrightarrow v \in F$.



O construcție inversă evidentă, ne arată că și pentru orice graf bipartit se poate asocia un hipergraf.

Un graf $G = (V(G), E(G))$ se numește **planar** dacă poate fi reprezentat în plan astfel încât fiecărui vîrf să-i corespundă un punct al planului, iar muchiilor le corespund curbe simple ce unesc punctele corespunzătoare extremităților lor și în plus **aceste curbe se interesectează (eventual) numai în vîrfuri**. Un graf care nu-i planar se numște **neplanar**. Exemple minimale de grafuri neplanare sînt grafurile K_5 și $K_{3,3}$.



Deși problema

PLAN Intrare: G un graf.
Intrebare: Este G planar ?

pare mult mai dificilă decât problema stabilei maxime (P1 din cursul trecut), ea subsumând noțiuni de topologie, s-a dovedit că este din **P** (Hopcroft & Tarjan , 1972, $O(n + m)$).

O modalitate uzuală de a defini clase de grafuri este de a **interzice** apariția unor subgrafuri induse, pentru grafurile acelei clase.

Dacă \mathcal{F} este o mulțime de grafuri, atunci un graf G se numește **\mathcal{F} -liber** (sau **\mathcal{F} -free**) dacă G nu are ca subgraf indus pe niciunul dintre grafurile lui \mathcal{F} . De exemplu, clasa grafurilor nule poate fi definită ca fiind clasa grafurilor K_2 -free; clasa grafurilor ale căror componente conexe sunt subgrafuri complete este clasa grafurilor P_3 -free; clasa grafurilor **triangulate** (sau **cordale**) este clasa grafurilor $(C_k)_{k \geq 4}$ -free.

7. Drumuri și circuite

Fie $G = (V(G), E(G))$ un graf.

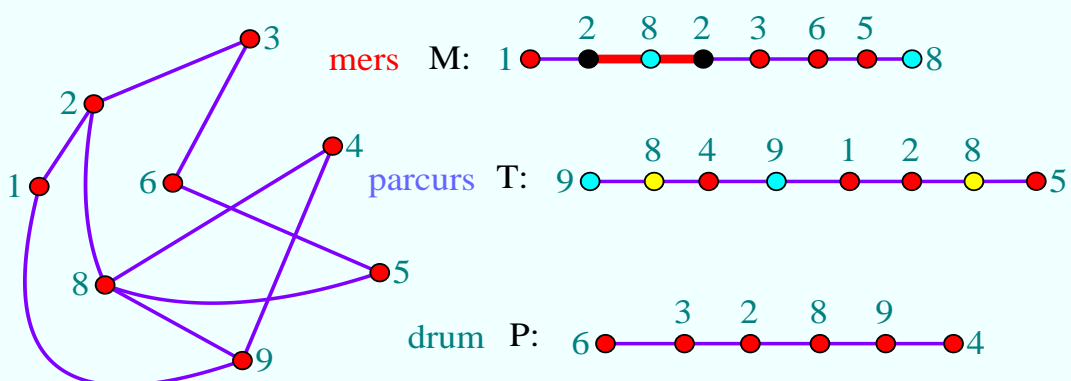
Se numește **mers** (**walk**) **de lungime** r **de la** v **la** w în G un șir de vîrfuri și muchii

$$(v =) v_0, v_0v_1, v_1, \dots, v_{r-1}, v_{r-1}v_r, v_r (= w);$$

v și w se numesc **extremitățile mersului**.

Dacă muchiile mersului sînt distincte atunci mersul se numește **parcurs** (**trail**) în G de la v la w .

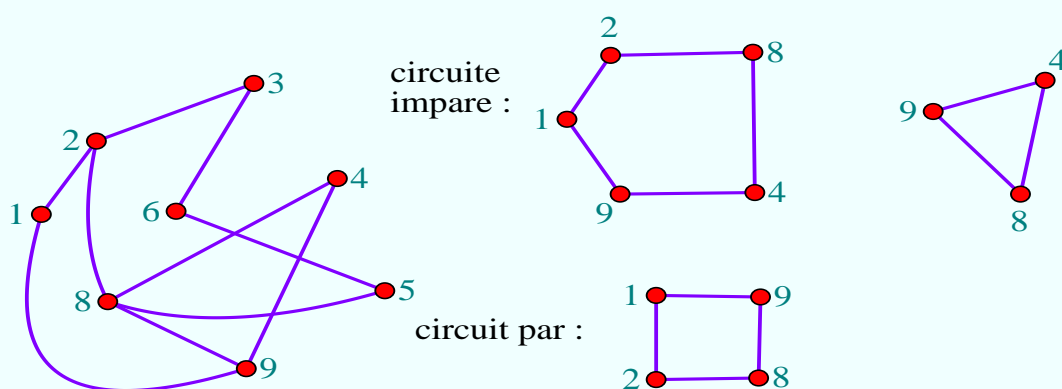
Dacă vîrfurile sînt distincte atunci mersul se numește **drum** (**path**) de la v la w .



Dacă $v = w$ atunci mersul (parcursul) se numește **închis**.

Dacă într-un mers toate vîrfurile sînt distincte, cu excepția extremităților, atunci mersul se numește **circuit** (sau **drum închis**).

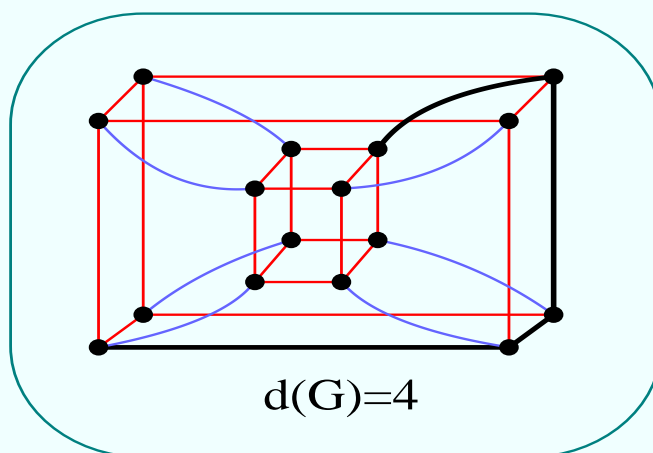
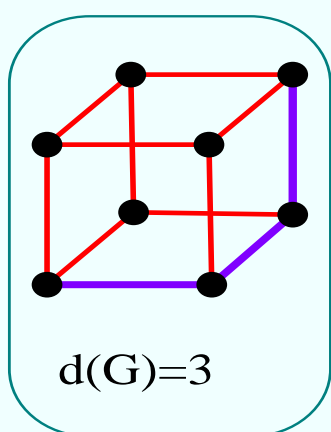
Un circuit este **par** sau **impar** după cum lungimea sa (numărul muchiilor) este pară sau impară.



Lungimea celui mai scurt circuit al grafului G (dacă G are circuite) se numește **grația** (girth) grafului G și se notează cu $g(G)$; lungimea celui mai lung circuit al lui G se numește **circumferința** lui G și se notează $c(G)$.

Dacă v și w sînt vîrfuri ale lui G , lungimea celui mai scurt drum de la v la w în G se numește **distanța** în G de la v la w și se notează $d_G(v, w)$.

Diametrul grafului G , notat $d(G)$ este $d(G) = \max\{d_G(v, w) | v, w \in V(G)\}$.

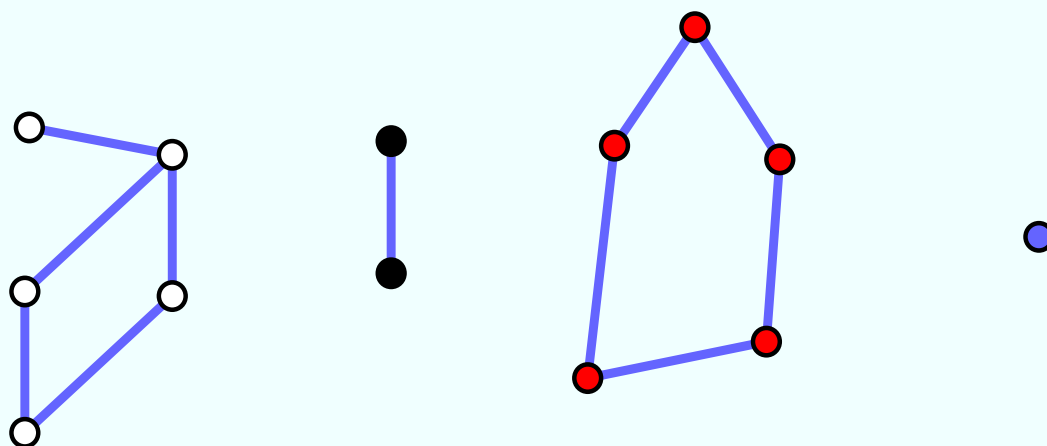


În proiectarea rețelelor de interconectare a procesoarelor este important ca graful G reprezentând rețeaua să aibă gradul maxim $\Delta(G)$ mic (restricție tehnologică) și diametrul $d(G)$ mic în raport cu numărul vîrfurilor (restricție de calitate a interconectării).

Definițiile de mai sus se extind, în mod evident, **pentru digrafuri singura modificare** fiind aceea că **se înlocuiesc muchiile cu arce**.

Un graf este **conex** dacă există (măcar) un drum între orice două vîrfuri ale sale; un graf care nu este conex se numește **neconex**.

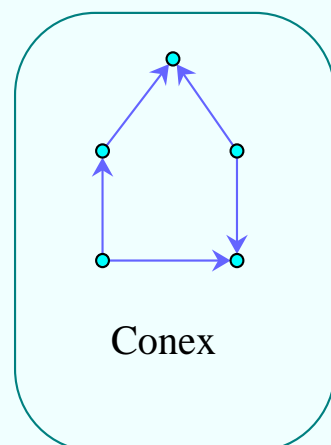
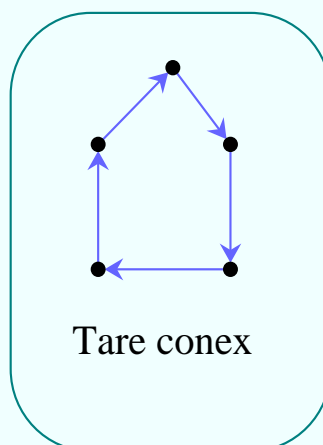
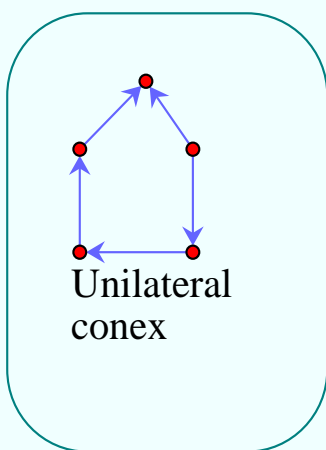
Orice graf G poate fi unic exprimat ca o reuniune disjunctă de *subgrafuri induce, conexe și maxime cu această proprietate*; aceste subgrafuri se numesc **componentele conexe** ale grafului G (mai precis, se pot defini componentele conexe ca subgrafurile induce de clasele de echivalență determinate pe $V(G)$ de relația de echivalență $\rho \subseteq V(G) \times V(G)$ definită prin : $v \rho w \Leftrightarrow$ există în G un drum de la v la w).



Graful din figura de mai sus are 4 componente conexe: una cu 1 vîrf, una cu 2 vîrfuri și două cu 5 vîrfuri.

Concepte analoage se pot defini și pentru digrafuri; dacă D este un digraf atunci :

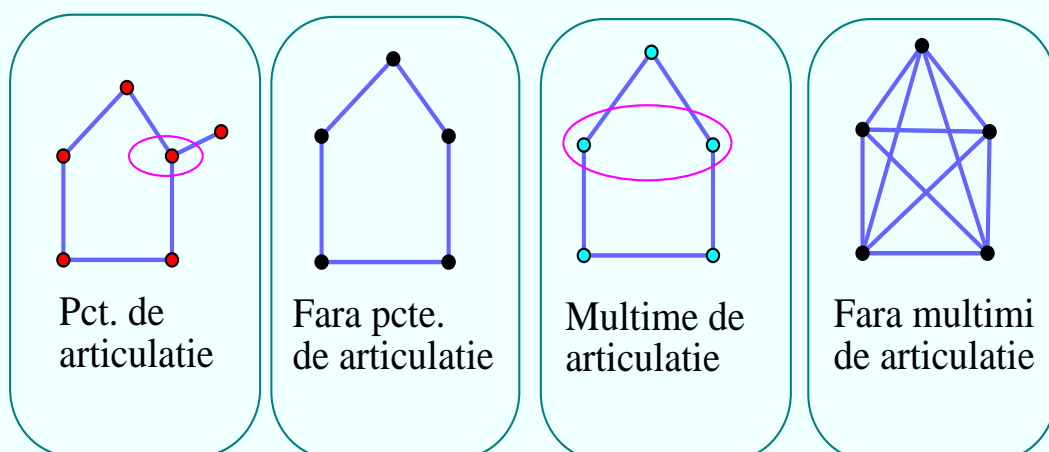
- D este **tare conex** dacă $\forall (v, w) \in V(D) \times V(D)$ există un drum în D de la v la w ;
- D este **unilateral conex** dacă $\forall (v, w) \in V(D) \times V(D)$ există în D un drum de la v la w **sau** un drum de la w la v ;
- D este **conex** dacă $G(D)$, graful suport al lui D , este conex.



Un graf conex care nu are circuite se numește **arbore**. Un graf ale cărui componente conexe sînt arbori se numește **pădure**.

Time to leave the trees !!!

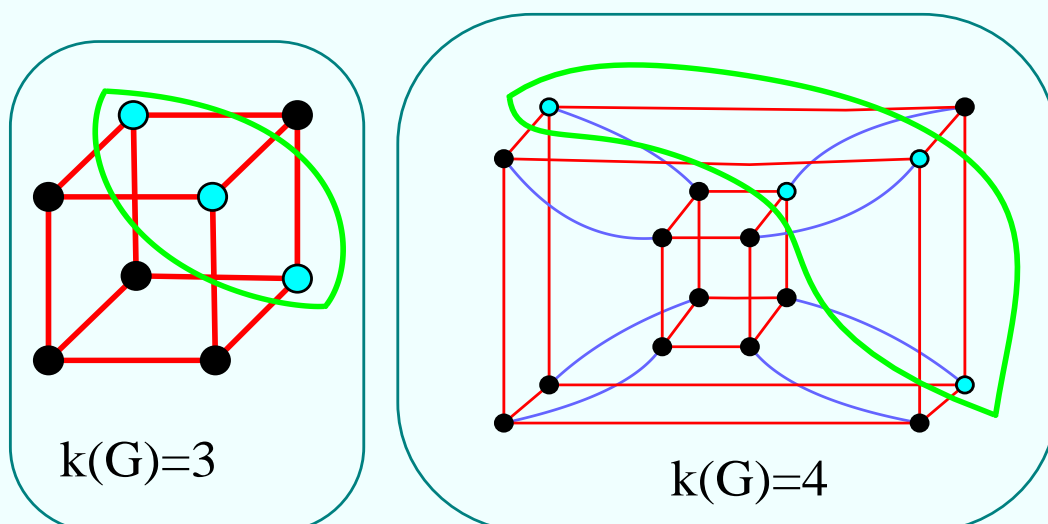
Dacă G este un graf conex, un vîrf $v \in V(G)$ cu proprietatea că $G - v$ este neconex se numește **vîrf (punct) de articulație**; mai general, o mulțime A de vîrfuri ale unui graf G se numește **mulțime separatoare de vîrfuri (mulțime de articulație)** dacă $G - A$ este neconex.



Fie p un număr întreg pozitiv;
 un graf G cu măcar p vîrfuri este **p -conex** dacă
 $G = K_p$ sau are cel puțin $p + 1$ vîrfuri și nu are
 mulțimi separatoare de vîrfuri de cardinal mai
 mic decît p .

Evident, G este 1-conex dacă și numai dacă
 este conex. Un graf 2-conex se mai numește
 și *bloc*.

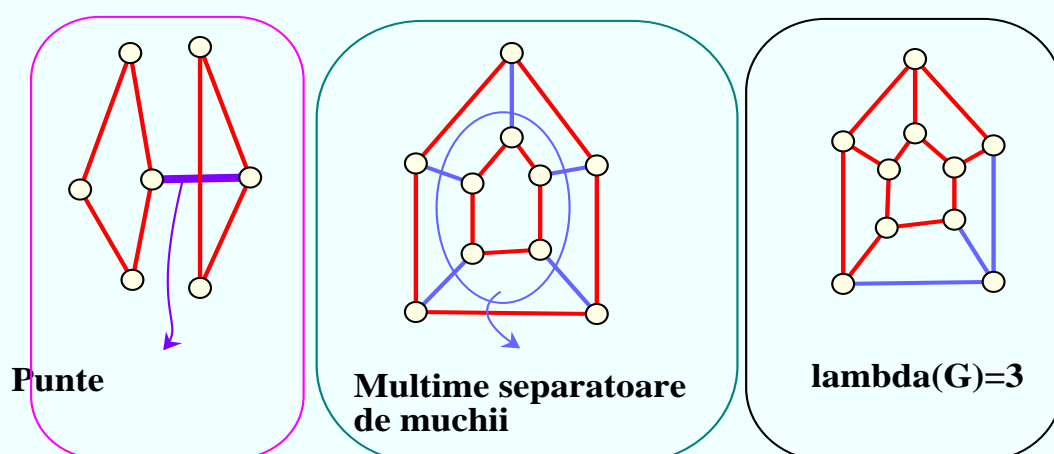
Numărul de conexiune al lui G , notat $k(G)$,
 este cel mai mare număr natural p pentru care
 G este p -conex.



Dacă G este un graf conex, o muchie $e \in E(G)$ cu proprietatea că $G - e$ este neconex se numește **punte** în graful G ; mai general, o mulțime A de muchii ale unui graf G se numește **mulțime separatoare de muchii** dacă $G - A$ este neconex.

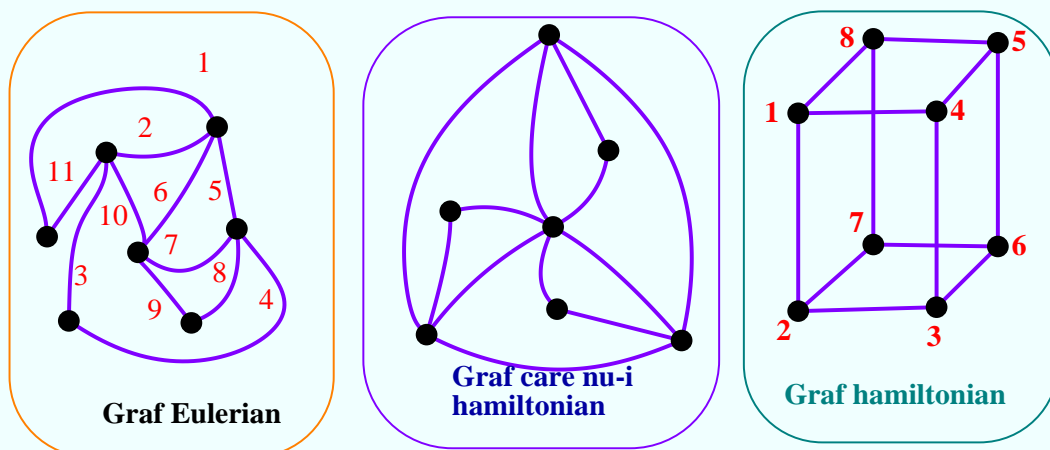
Un graf G cu cel puțin p vîrfuri este **p –muchie-conex** dacă nu admite mulțimi separatoare de muchii de cardinal mai mic decât p .

Numărul de muchie-conexiune al lui G , notat $\lambda(G)$, este cel mai mare număr natural p pentru care G este p –muchie-conex .



Un graf (sau digraf) se numește **eulerian** dacă admite un parcurs închis care folosește fiecare muchie a grafului (respectiv, fiecare arc al digrafului).

Un (di)graf G se numește **hamiltonian** dacă are un circuit care trece prin fiecare vîrf.



În timp ce problema testării dacă un graf este eulerian este foarte simplă (Euler 1736 : conex și cu toate vîrfurile de grad par), problema

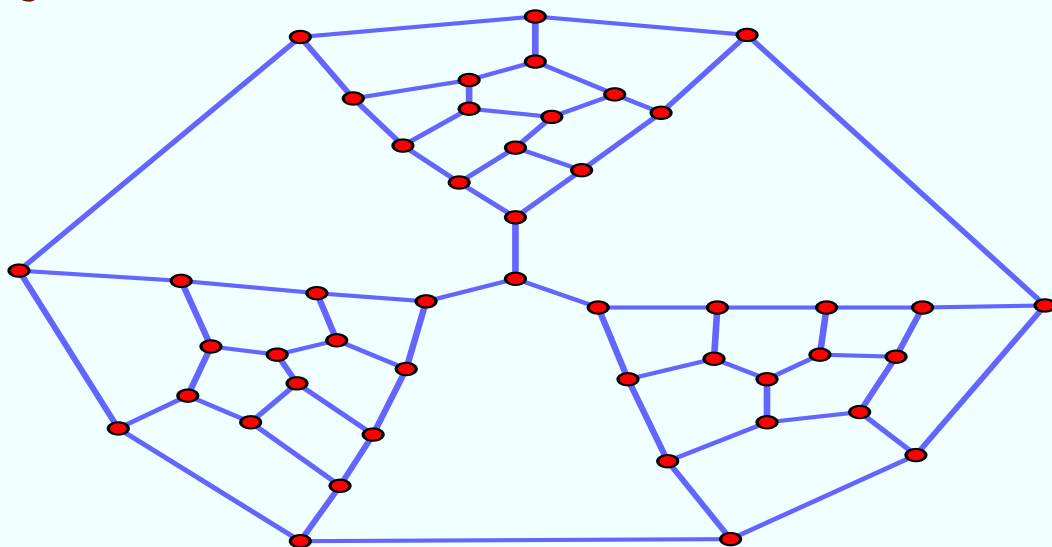
HAM Intrare: G un graf.
 Intrebare: Este G hamiltonian ?

este **NP**-completă. Apartenența la **NP** este evidentă: un circuit hamiltonian, se poate indica printr-o permutare a vârfurilor care poate fi testată în timp liniar. În schimb pentru problema

NH Intrare: G un graf.

Intrebare: Este adevărat că G nu-i hamiltonian?

nu se cunoaște o demonstrație a apartenenței la **NP** (se observă că este din **co-NP**). Nu se poate da o demonstrație succintă că graful de mai jos nu e hamiltonian:



3-conex, planar, si nehamiltonian
(Tutte)

8. Matrici asociate.

Dacă $G = (\{v_1, \dots, v_n\}, \{e_1, \dots, e_m\})$ este un graf, atunci

Matricea de adiacență a grafului G este matricea $A = (a_{ij})_{n \times n}$, unde

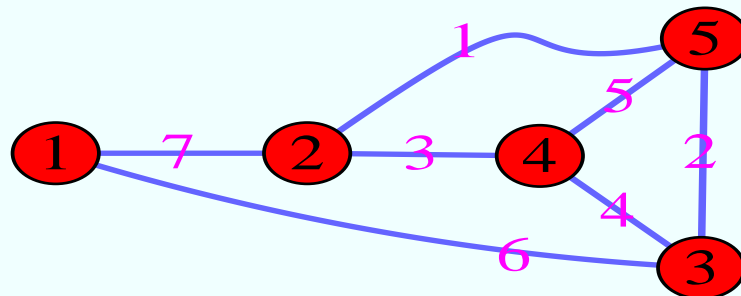
$$a_{ij} = \begin{cases} 1 & \text{dacă } v_i \text{ și } v_j \text{ sînt adiacente} \\ 0 & \text{altminteri.} \end{cases}$$

Matricea de incidență a grafului G este matricea $B = (b_{ij})_{n \times m}$, unde

$$b_{ij} = \begin{cases} 1 & \text{dacă } v_i \text{ și } e_j \text{ sînt incidente} \\ 0 & \text{altminteri.} \end{cases}$$

În cazul digrafurilor, se pot asocia similar astfel de matrici, în care, evident se poate indica și orientarea arcelor, folosind elementele 0, 1 și -1.

Pentru graful din figura de mai jos,



matricea de adiacență este:

$$A = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 \end{pmatrix},$$

iar matricea de incidență:

$$B = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}.$$

Valorile proprii și polinomul caracteristic ale matricii de adiacență se numesc **valorile proprii ale grafului**, respectiv, **polinomul caracteristic al grafului**.

9 Structuri de date utilizate în reprezentarea (di)grafurilor.

Fie $G = (V, E)$ un (di)graf cu $V = \{1, 2, \dots, n\}$ și $|E| = e$.

Cele mai uzuale structuri de date utilizate pentru reprezentarea (di)grafului G sunt:

a) matricea de adiacență

Dacă $A = (a_{ij})_{n \times n}$ este matricea de adiacență a lui G atunci, reprezentarea acesteia cu ajutorul unui tablou bidimensional va necesita $O(n^2)$ operații pentru orice inițializare, deci orice algoritm, care folosește o astfel de reprezentare, are complexitatea $\Omega(n^2)$.

Cu această structură de date testarea dacă două vârfuri sunt sau nu adiacente se face în $O(1)$, dar parcurgerea lui $N_G(v)$ (sau $N_G^+(v)$), pentru un vârf oarecare $v \in V$, necesită $\Omega(n)$ operații.

b) listele de adiacență

Pentru fiecare vârf $v \in V$ se consideră o listă $A(v)$ a vecinilor săi în G .

Dacă G este graf, atunci $A(v)$ conține

$N_G(v) = \{w \mid w \in V \text{ și } vw \in E\}$ iar dacă G este digraf atunci $A(v)$ conține

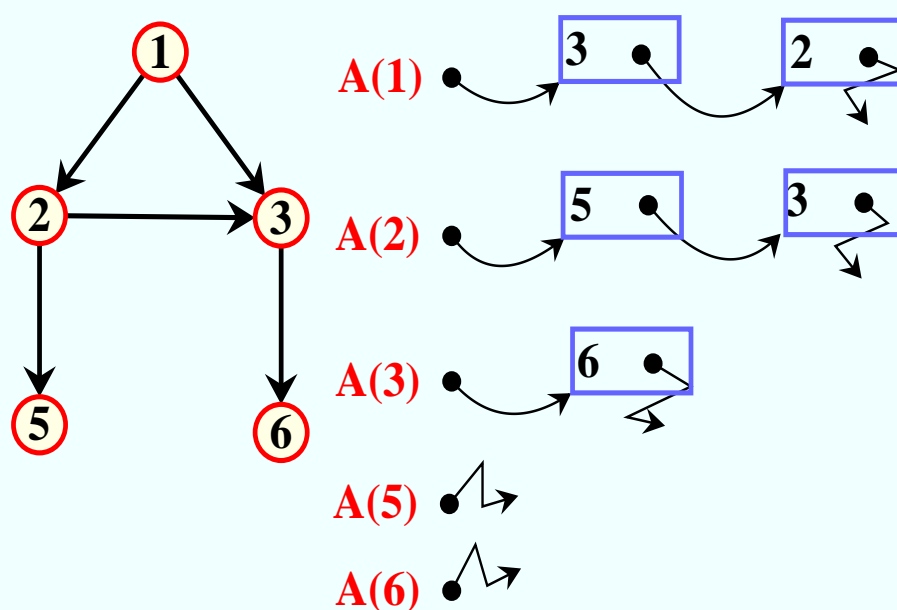
$N_G^+(v) = \{w \mid w \in V \text{ și } vw \in E\}$.

Pentru cazul în care G este graf, fiecare muchie $vw \in E$ va genera două elemente în listele de adiacență, unul în $A(v)$ și celălalt în $A(w)$. Spațiul total de memorie utilizat va fi de $O(n + 2e)$. Pentru cazul în care G este digraf, spațiul de memorie utilizat este de $O(n + e)$.

Listele de adiacență pot fi reprezentate cu ajutorul tablourilor sau ca structuri dinamice de date (liste înlănțuite).

Testarea dacă un vârf fixat v este adiacent cu un vârf oarecare w în G se face în $\Omega(d_G(v))$, dar se poate parcurge $N_G(v)$ în timpul $O(d_G(v))$ și nu $O(n)$ ca în cazul matricii de adiacență.

Pentru digraful desenat mai jos, sunt reprezentate listele de adiacență:



Cu noțiunile și terminologia minimală descrisă în secțiunile 1-9 ale acestui prim capitol, se poate trece la probleme algoritmice specifice.

II Probleme de drum în (di)grafuri

1. Parcurgeri sistematice ale (di)grafurilor.

Graph search- paradigmă algoritmică utilizată pentru a desemna o metodă sistematică de parcurgere a mulțimii vârfurilor la care se poate ajunge prin drumuri într-un (di)graf de la un vârf fixat.

Dat $G = (V, E)$ un (di)graf cu mulțimea vârfurilor $V = \{1, \dots, n\}$ și $s \in V$, se cere să se genereze "eficient" mulțimea
$$S = \{v \in V \mid \exists D \text{ drum în } G \text{ de la } s \text{ la } v\}.$$

(Di)graful G e reprezentat cu **listele de adiacență** (e nevoie de aflarea eficientă a mulțimii vecinilor nodului curent, în procesul sistematic de vizitare).

Prezentăm succint cele două tehnici principale de parcurgere.

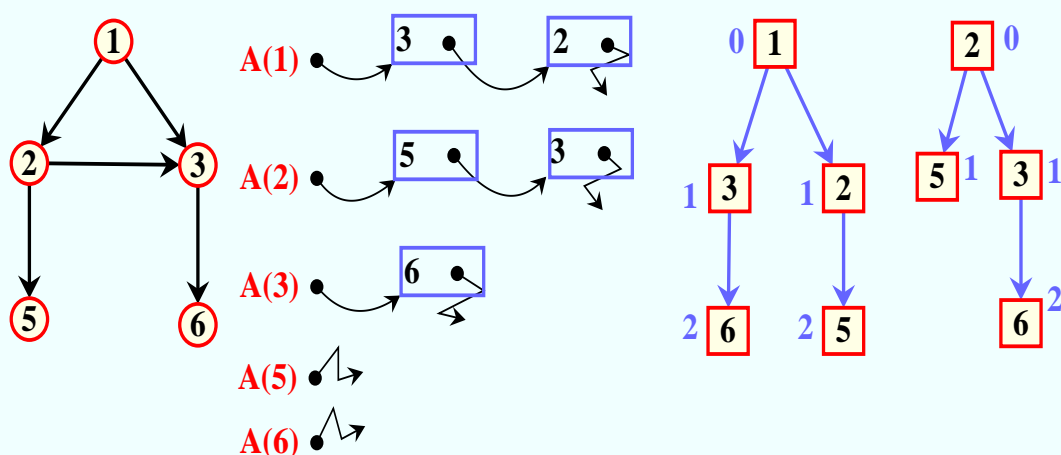
bfs - breadth first search

Inițial, $\forall v \in V$ are eticheta $label(v) < 0$.

```
 $label(s) \leftarrow 0$      $parent(s) \leftarrow 0$ ;  
creează coada  $Q$  conținând  $s$ ;  
while  $Q \neq \emptyset$  do  
{  
     $v \leftarrow$  vârful din capul cozii  $Q$ ;  
    șterge vârful din capul cozii  $Q$ ;  
    for  $w \in A(v)$  do  
        { if  $label(w) < 0$  then  
            {  $label(w) \leftarrow label(v) + 1$ ;  
               $parent(w) \leftarrow v$ ;  
              introdu  $w$  în coada  $Q$   
            }  
        }  
    }  
}
```

Evident:

- $S = \{v \in V \mid \text{label}(v) \geq 0\}$;
 - $\forall v \in V \text{ label}(v) = d_G(s, v)$;
 - variabila *parent* definește **arborele bfs** asociat căutării din s : dacă G e graf atunci acesta este arbore parțial al componente conexe a lui G la care aparține s ; dacă G este digraf atunci acesta este o arborescență (arbore orientat cu toate vârfurile accesibile prin drumuri din rădăcină);
 - deoarece fiecare listă de adiacență a unui vârf din mulțimea S este traversată exact o dată, complexitatea timp a lui $\text{bfs}(s)$ este $O(n_S + m_S)$, unde $n_S = |S|$ iar $m_S = |E([S]_G)|$;
- În figura următoare sunt desenați arborii corepunzători lui $\text{bfs}(1)$ și $\text{bfs}(2)$.



dfs - depth first search

Inițial, $\forall v \in V$ are eticheta $label(v) < 0$

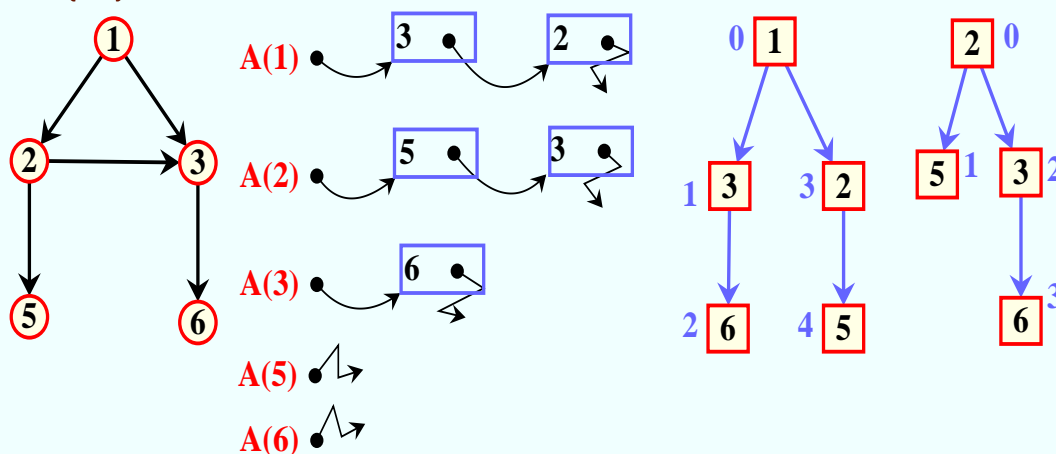
și toți pointerii de parcurgere a listelor de adiacență sunt la început.

```
label(s)  $\leftarrow$  0    parent(s)  $\leftarrow$  0;
creează stiva  $S$  conținând  $s$ ;
 $n_S \leftarrow 1$ 
while  $S \neq \emptyset$  do
{
     $v \leftarrow$  vârful din capul stivei  $S$ ;
     $w \leftarrow next[A(v)]$ 
    if  $\exists w$  then
        if  $label(w) < 0$  then
        {
             $label(w) \leftarrow n_S$ ;  $n_S++$ ;
             $parent(w) \leftarrow v$ ;
            introdu  $w$  în stiva  $S$ 
        }
        else NOP (aici; dar se poate utiliza !!)
    else șterge vârful din capul stivei  $S$ 
    // s-a terminat căutarea din  $v$ ;
}
```

Iarăși, rezultă imediat că

$S = \{v \in V | label(v) \geq 0\}$ și complexitatea timp a lui $dfs(s)$ este $O(n_S + m_S)$.

Pentru exemplul nostru de lucru, arborii dfs(1) și dfs(2) sunt ilustrați mai jos.



Parcurgerile sistematice sunt importante pentru obținerea unor algoritmi eficienți pentru determinarea componentelor conexe în grafuri, pentru determinarea componentelor tari conexe în digrafuri, componentelor 2-conexe în grafuri etc., (care sunt preprocesări uzuale pentru probleme mai complicate).

Ele sunt esențiale în problemele din Inteligența Artificială, unde spațiul stărilor de căutare poate fi văzut ca un graf . De data aceasta graful este dat implicit; în fiecare nod (stare) se dispune de un predicat *precondiție* care este utilizat de o funcție *neighbours* care întoarce lista nodurilor accesibile în contextul curent din acel nod. Graful explicit care se poate construi în principiu, este folosit pentru descrierea algoritmului și analizele de complexitate.

2. Probleme de drum minim.

Fie $G = (V, E)$ un digraf cu mulțimea vîrfurilor $V = \{1, \dots, n\}$. Considerăm dată o funcție $a : E \rightarrow \mathbf{R}$ cu interpretarea:

$\forall ij \in E \quad a(ij) = \text{costul arcului } ij \text{ (ponderea, lungimea, etc).}$

Dacă digraful G este reprezentat cu ajutorul listelor de adiacență, atunci costul arcului ij este un cîmp în nodul din lista de adiacență a lui i ce reprezintă acest arc.

Pentru ușurința notațiilor vom folosi reprezentarea digrafului G cu ajutorul matricii de cost-adiacență $A = (a_{ij})_{n \times n}$ cu

$$a_{ij} = \begin{cases} a(ij) & \text{dacă } ij \in E \\ \infty & \text{altfel} \end{cases}$$

Aici ∞ desemnează un număr real mare în raport cu celelalte costuri (de exemplu $\infty > n \times \max_{ij \in E} a(ij)$) și vom presupune în plus că

$$\infty \mp a = \infty, \quad \infty + \infty = \infty.$$

(Este posibil, de asemenea, ca ∞ să semnifice acces terminat cu insucces în structura de date în care se reprezintă matricea A).

Dacă $i, j \in V$, vom nota cu

$$\mathcal{D}_{ij} = \{D_{ij} \mid D_{ij} \text{ drum în } G \text{ de la } i \text{ la } j\}.$$

Pentru $D_{ij} \in \mathcal{D}_{ij}$

$$D_{ij} : (i =)v_0, v_0v_1, v_1, \dots, v_{r-1}, v_{r-1}v_r, v_r(=j)$$

mulțimea vîrfurilor este $V(D_{ij}) = \{v_0, v_1, \dots, v_r\}$
și mulțimea arcelor

$$E(D_{ij}) = \{v_0v_1, v_1v_2, \dots, v_{r-1}v_r\}.$$

Orice vîrf $k \neq i, j$ al lui D_{ij} , determină pe D_{ij} două drumuri $D_{ik} \in \mathcal{D}_{ik}$ și $D_{kj} \in \mathcal{D}_{kj}$. Vom nota $D_{ij} = D_{ik} \circ D_{kj}$.

Costul unui drum $D_{ij} \in \mathcal{D}_{ij}$ se definește

$$a(D_{ij}) = 0 + \sum_{ij \in E(D_{ij})} a_{ij}.$$

În particular, $a(D_{ii}) = 0$.

Principalele probleme de drum (de cost) minim care apar în aplicații practice (sau sînt utile în rezolvarea altor probleme de optimizare combinatorie) sînt:

P1 Date G digraf; $a : E(G) \rightarrow \mathbf{R}$; $s, t \in V(G), s \neq t$.
Să se determine $D_{st}^* \in \mathcal{D}_{st}$, astfel încît
 $a(D_{st}^*) = \min\{a(D_{st}) \mid D_{st} \in \mathcal{D}_{st}\}$.

P2 Date G digraf; $a : E(G) \rightarrow \mathbf{R}$; $s \in V(G)$.
Să se determine $D_{si}^* \in \mathcal{D}_{si} \ \forall i \in V(G)$, a.î.
 $a(D_{si}^*) = \min\{a(D_{si}) \mid D_{si} \in \mathcal{D}_{si}\}$.

P3 Date G digraf; $a : E(G) \rightarrow \mathbf{R}$.
Să se determine $D_{ij}^* \in \mathcal{D}_{ij} \ \forall i, j \in V(G)$, a.î.
 $a(D_{ij}^*) = \min\{a(D_{ij}) \mid D_{ij} \in \mathcal{D}_{ij}\}$.

Observații:

1. Cu convenția folosită în reprezentarea matricilor de cost adiacență, se poate considera că $\mathcal{D}_{ij} \neq \emptyset \ \forall i, j \in V$.

Dacă $a(D_{ij}) < \infty$ atunci D_{ij} este drum (adevărat) în G de la i la j , iar dacă $a(D_{ij}) = \infty$, atunci D_{ij} este drum în digraful complet simetric obținut din G prin adăugarea arcelor lipsă, cu costul ∞ .

Rezultă că toate **mulțimile pe care se consideră minimele** în problemele precedente, sînt **nevide** și, cum digrafurile considerate sînt finite, rezultă că aceste mulțimi sînt **finite** (în fiecare drum vîrfurile sînt distincte), **deci minimele considerate există**.

2. Algoritmii de rezolvare a problemei (P1) se obțin din algoritmii de rezolvare a problemei (P2) adăugîndu-li-se un test suplimentar (evident) de oprire.

3. Problema (P3) se poate rezolva iterînd un algoritm de rezolvare a problemei (P2). Sînt posibile însă soluții mai eficiente.

Aplicații. Vom schița în continuare trei aplicații practice posibile ale acestor probleme.

a) $G = (V, E)$ reprezintă o rețea de comunicație cu nodurile V și rutele directe între noduri formând mulțimea E .

Dacă $a(e)$ reprezintă lungimea arcului e , atunci cele trei probleme de mai sus reprezintă probleme naturale, care se pun în astfel de rețele: *"determinarea drumurilor celor mai scurte"*.

Dacă $p_{ij} \in (0, 1]$ este probabilitatea de funcționare a arcului $ij \in E$ atunci, presupunând că arcele funcționează independent unele de altele, probabilitatea de funcționare a drumului D este

$$p(D) = \prod_{ij \in E(D)} p_{ij}.$$

Considerând $a_{ij} = -\log p_{ij}$, problema drumului de cost minim de la s la t semnifică determinarea *drumului cel mai sigur* de la s la t .

b) Rețele *PERT* (Project Evaluation and Review Technique).

Fie $P = \{A_1, \dots, A_n\}$ mulțimea activităților atomice ale unui proiect de anvergură (n este mare). P este o mulțime parțial ordonată cu relația de ordine

$A_i < A_j \Leftrightarrow$ activitatea A_j nu poate începe decît după terminarea activității A_i .

Se cunoaște, pentru fiecare activitate A_i timpul de execuție t_i .

Se cere să se determine un plan de organizare a proiectului astfel încît timpul total de execuție să fie minim. (Notăm că problemele practice sînt mai complexe datorită restricțiilor de utilizare concurentă a resurselor - oameni, utilaje, etc. - de către diversele activități).

Ideea generală pe care se bazează pachetele soft care rezolvă astfel de probleme este de a asocia proiectului un digraf aciclic (rețeaua PERT) astfel:

Fiecărei activități A_l i se asociază arcul $i_l j_l$ de cost $a(i_l j_l) = t_l$.

Nodul i_l reprezintă evenimentul de început al activității A_l , iar nodul j_l reprezintă evenimentul de sfârșit al activității A_l .

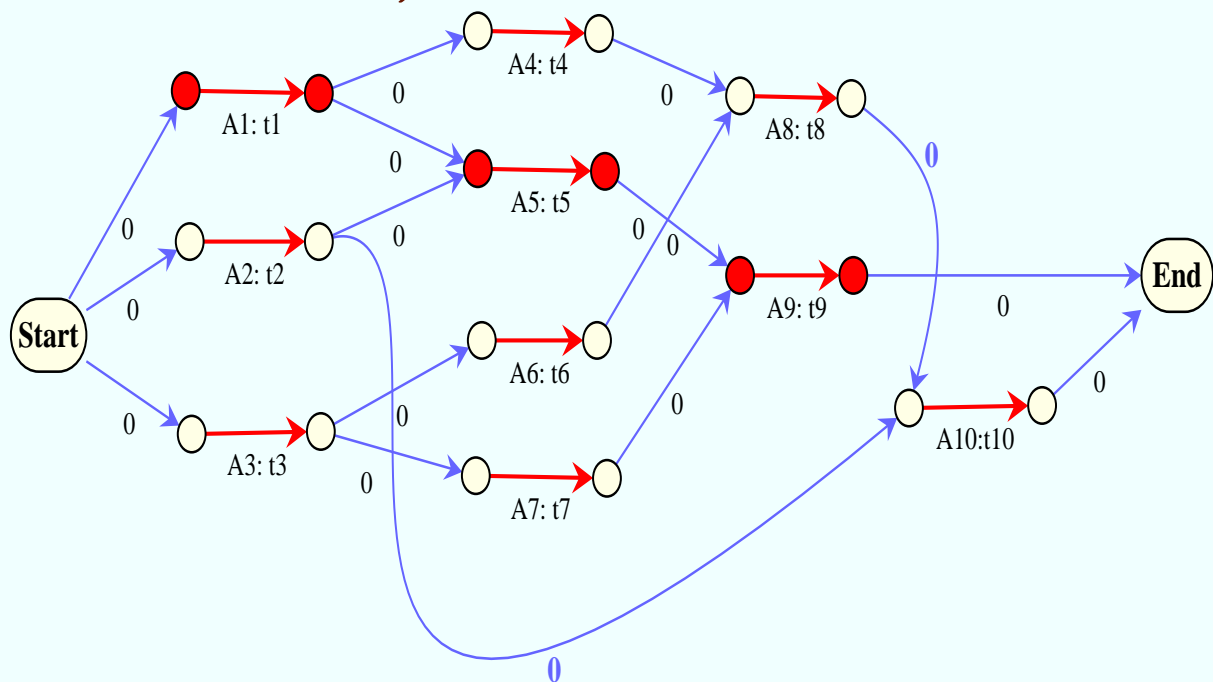
Dacă activitatea A_k poate începe imediat după terminarea activității A_l , se introduce în digraf arcul $j_l i_k$ (activitate fictivă) de cost 0.

Se asociază un eveniment s (START) unit prin arce de cost 0 cu elementele minimale ale lui $(P, <)$ și un eveniment t (END) de care vor fi unite prin câte un arc fiecare element maximal al lui P .

În digraful obținut (care este evident aciclic) costul maxim al unui drum de la s la t reprezintă *cel mai scurt timp posibil de execuție a proiectului*.

Un drum de cost maxim se numește **drum critic**, întrucât întârzierea oricărei activități corespunzătoare unui arc de pe drumul critic conduce la întârzierea întregului proiect.

Figura de mai jos ilustrează tipul de digraf (aciclic) care se formează și se evidențiază un posibil drum critic. Dificultatea majoră este în construcția digrafului (modelarea problemei reale) și problema devine extrem de interesantă dacă (dar și NP-dificilă) dacă se introduc și alte tipuri de restricții între activități (nu numai temporale).



c) *Problema rucsacului*. Dispunem de n obiecte de volume a_1, \dots, a_n și de un rucsac de volum b ($a_i \in \mathbb{Z}_+, i = \overline{1, n}, b \in \mathbb{Z}_+, a_i \leq b \ \forall i = \overline{1, n}$).

Cunoscînd "profitul" $p_i \in \mathbf{R}_+$ adus de introducerea obiectului i în rucsac, se cere **să se determine o încărcare a rucsacului de profit total maxim:**

$$\max \left\{ \sum_{i=1}^n p_i x_i \mid \sum_{i=1}^n a_i x_i \leq b, x_i \in \{0, 1\} \forall i = \overline{1, n} \right\}.$$

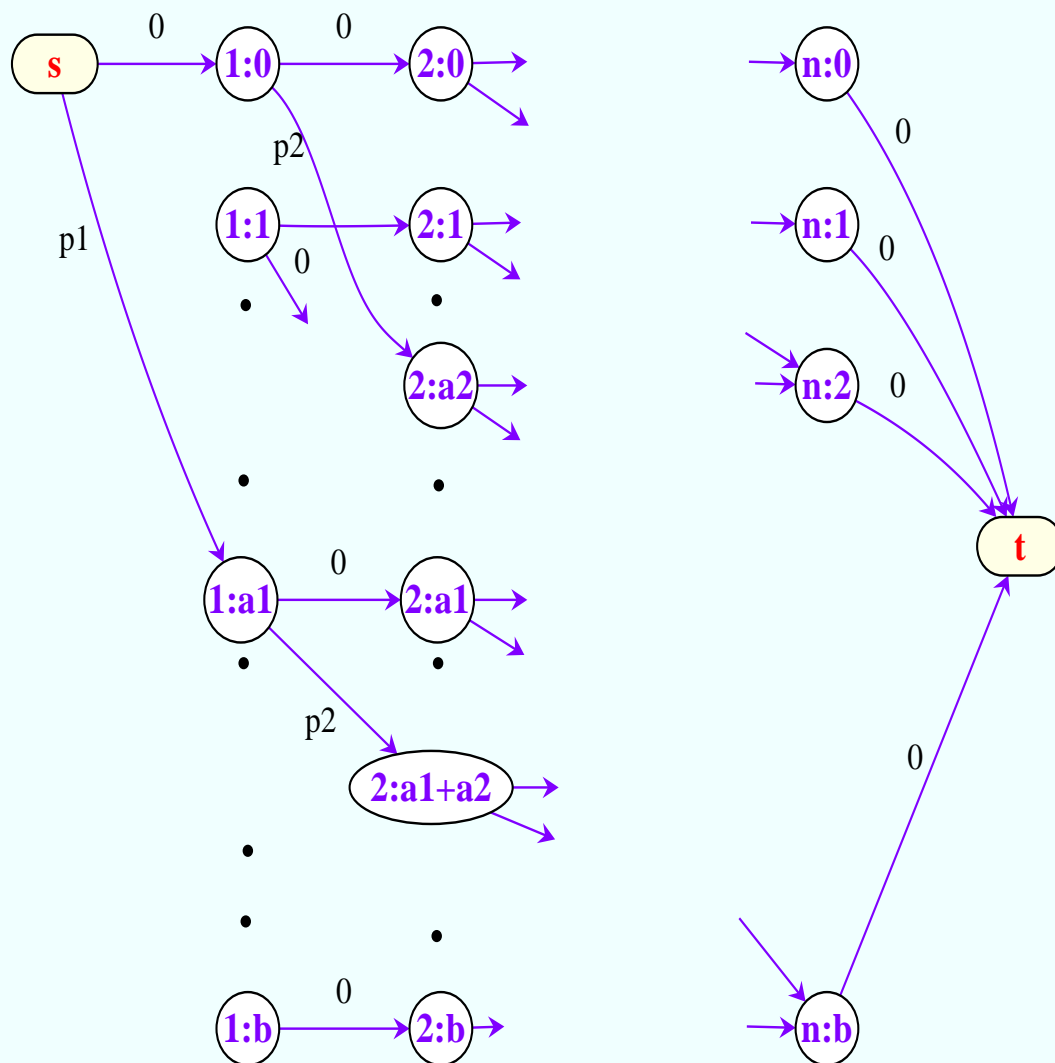
Problema, deși interesantă în unele aplicații (de exemplu la încărcarea vapoarelor într-un port) a fost aleasă pentru a pune în evidență legătura dintre metoda programării dinamice (discrete) și problemele de drum minim într-un digraf.

Considerăm $G = (V, E)$ un digraf cu $V = \{s\} \cup V_1 \cup V_2 \cup \dots \cup V_n \cup \{t\}$, unde $V_i = \{i^0, i^1, \dots, i^b\}$ este asociat obiectului i , $i = \overline{1, n}$.

Arcele lui G sînt:

- $s1^0$ și $s1^{a_1}$ cu $a(s1^0) = 0$ și $a(s1^{a_1}) = p_1$.
(se pune obiectul 1 în rucsac și se obține profitul p_1 , ajungându-se la nivelul a_1 de umplere, sau nu se pune obiectul 1 în rucsac, profitul fiind 0 și nivelul de umplere rămânând 0).
- $\forall i = \overline{2, n} \ \forall j = \overline{0, b}$:
 $(i - 1)^j i^j$ cu $a((i - 1)^j i^j) = 0$;
 (dacă decidem să nu introducem obiectul i în rucsac, atunci de la încărcarea rucsacului cu primele $i - 1$ obiecte se trece la o încărcare cu primele i obiecte în care nu este selectat obiectul i , deci se rămâne pe același nivel de încărcare j , iar profitul ce se va adăuga este 0).
 Dacă $j - a_i \geq 0$ atunci avem și arcul
 $(i - 1)^{j-a_i} i^j$ cu $a((i - 1)^{j-a_i} i^j) = p_i$.
 (se poate ajunge la o încărcare j prin introducerea obiectului i de volum a_i la o încărcare a primelor $i - 1$ obiecte de nivel $j - a_i$).
- $\forall j = \overline{0, b}$: $n^j t$ cu $a(n^j t) = 0$.

Figura următoare ilustrează construcția acestui digraf.



Se observă din construcție, că orice drum de la s la t în G corespunde unei submulțimi de obiecte cu suma volumelor mai mică sau egală cu b și de profit egal cu costul acestui drum. Reciproc, oricărei mulțimi de obiecte cu suma volumelor nedepășind b îi corespunde un drum de la s la t în G .

Rezultă că dacă în digraful G se determină un drum de cost maxim de la s la t se rezolvă problema rucsacului.

Notăm că descrierea (statică) a digrafului G poate fi ușor transformată în una procedurală astfel încât digraful să reprezinte doar ilustrarea unei metode de programare dinamică (prospectivă) pentru rezolvarea problemei rucsacului.

Atenție ! Problema rucsacului este referită uzual ca una din problemele NP-dificile. Soluția polinomială descrisă mai sus conduce la un digraf aciclic cu $O(nb)$ vârfuri, care nu-i dimensiunea intrării !!!

Rezolvarea problemei P2

Teoremă. 1. Fie $G = (V, E)$ digraf, $V = \{1, \dots, n\}$, $s \in V$ și $a : E \rightarrow \mathbb{R}$, astfel încât

$$(I) \quad \forall C \text{ circuit în } G, a(C) > 0.$$

Atunci (u_1, \dots, u_n) este o soluție a sistemului

$$(*) \quad \begin{cases} u_s = 0 \\ u_i = \min_{j \neq i} (u_j + a_{ji}) \quad \forall i \neq s. \end{cases}$$

dacă și numai dacă $\forall i \in V, \exists D_{si}^* \in \mathcal{D}_{si}$ astfel încât $a(D_{si}^*) = u_i$ și $a(D_{si}^*) = \min\{a(D) \mid D \in \mathcal{D}_{si}\}$.

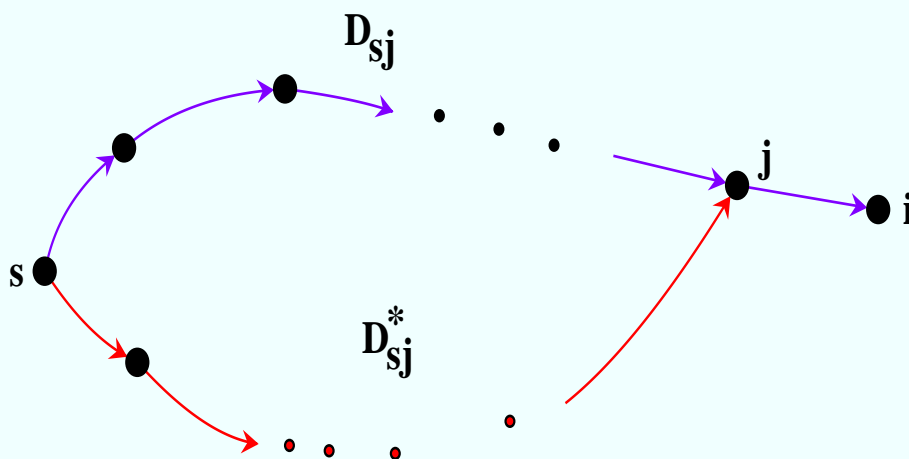
Demonstrație: " \Leftarrow " Fie $D_{si}^* (i \in V)$ soluții ale problemei (P2) cu $a(D_{si}^*) = \min\{a(D) \mid D \in \mathcal{D}_{si}\}$. Notăm cu $u_i = a(D_{si}^*) (i \in V)$.

Ipoteza (I) asigură faptul că $u_s = 0$, adică prima ecuație a sistemului (*) este verificată. Pentru $i \neq s$ drumul D_{si}^* are un penultim vîrf j . Dacă D_{sj} este drumul de la s la j determinat pe D_{si}^* de vîrf j , avem: $u_i = a(D_{si}^*) = a(D_{sj}) + a_{ji} \geq a(D_{sj}^*) + a_{ji} = u_j + a_{ji}$.

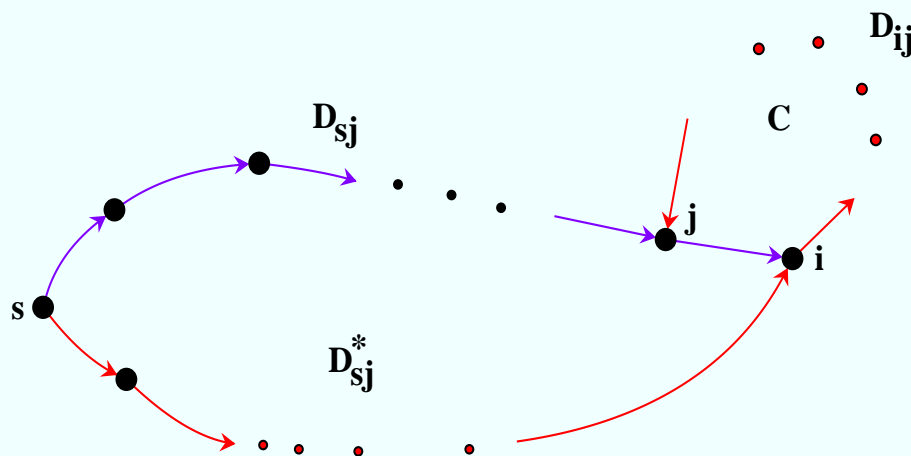
Arătăm că $u_i = u_j + a_{ji}$.

Presupunem că $u_i > u_j + a_{ji}$, adică $a(D_{sj}) > a(D_{sj}^*)$. Avem două cazuri posibile:

1. $i \notin V(D_{sj}^*)$. Atunci $D^1 = D_{sj}^* \circ (j, ji, i) \in \mathcal{D}_{si}$ și $a(D^1) = a(D_{sj}^*) + a_{ji} < a(D_{sj}) + a_{ji} = a(D_{si}^*)$, ceea ce contrazice alegerea drumului D_{si}^* (vezi figura următoare).



2. $i \in V(D_{sj}^*)$. Fie $D_{sj}^* = D_{si} \circ D_{ij}$ cele două drumuri determinate pe D_{sj}^* de vârful i . Atunci circuitul $C = D_{ij} \circ (j, ji, i)$ are costul $a(C) = a(D_{ij}) + a_{ji} = a(D_{sj}^*) - a(D_{si}) + a_{ji} = u_j + a_{ji} - a(D_{si}) \leq u_j + a_{ji} - a(D_{si}^*) = u_j + a_{ji} - u_i < 0$, contrazicînd ipoteza (I) (vezi figura următoare).



Deci am demonstrat că $\forall i \neq s \Rightarrow u_i = u_j + a_{ji}$.

Dacă u_i nu satisface (*), atunci ar exista j_1 astfel încât $u_i > u_{j_1} + a_{j_1 i}$. Atunci, ca mai sus, se poate construi un drum de cost mai mic decât u_i de la s la i .

Rezultă că suficiența teoremei este demonstrată.

Notăm că de fapt am dovedit mai sus că $a(D_{sj}) = a(D_{sj}^*)$ adică, dacă j este vârful dinaintea lui i pe un drum de cost minim de la s la i atunci și porțiunea de drum de la s la j este drum de cost minim de la s la j . Inductiv, rezultă :

(Principiul optimalității al lui Bellman) *dacă D_{si}^* este drum de cost minim de la s la i atunci $\forall j \in V(D_{si}^*),$ dacă $D_{si}^* = D_{sj} \circ D_{ji}$ atunci D_{sj} (respectiv D_{ji}) sînt drumuri de cost minim de la s la j (respectiv de la j la i).*

" \Rightarrow ". Dovedim că dacă (u_1, \dots, u_n) este o soluție a lui (*) atunci

(a) $\exists D_{si} \in \mathcal{D}_{si} : u_i = a(D_{si}), \forall i \in V.$

(b) $\forall i \in V \ u_i = \min\{a(D) \mid D \in \mathcal{D}_{si}\} (= a(D_{si}^*)).$

(a) Dacă $i = s$, atunci $u_s = 0$ și drumul D_{ss} satisface $a(D_{ss}) = 0 = u_s$.

Dacă $i \neq s$, considerăm următorul algoritm:

$v \leftarrow i; k \leftarrow 0;$

while $v \neq s$ **do**

 { determină w astfel încît $u_v = u_w + a_{vw};$

 // $\exists w$ pentru că u_v satisface (*)

$i_k \leftarrow v; k \leftarrow k + 1; v \leftarrow w$

 }

$i_{k+1} \leftarrow s$

Să observăm că algoritmul determină drumul

$$D : (s =) i_{k+1}, i_{k+1}i_k, \dots, i_1, i_1i_0, i_0(= i)$$

cu $D \in \mathcal{D}_{si}$ satisfăcînd $a(D) = a(i_{k+1}i_k) + \dots + a(i_1i_0) = (u_{i_k} - u_{i_{k+1}}) + (u_{i_{k-1}} - u_{i_k}) + \dots + (u_{i_0} - u_{i_1}) = u_{i_0} - u_{i_{k+1}} = u_i - u_s = u_i$.

Nu este posibil ca într-o iterație oarecare $w \in \{i_0, \dots, i_{k-1}\}$, căci atunci s-ar obține un circuit C de cost total 0, contrazicînd ipoteza (I).

Din construcție, se observă că $u_i = u_{i_1} + a_{i_1 i}$.

(b) Fie $\bar{u}_i = a(D_{si}^*) \forall i \in V$. Conform primei părți a demonstrației $\bar{u}_i, i = \overline{1, n}$, satisfac sistemul (*). Presupunem că $u = (u_1, \dots, u_n) \neq \bar{u} = (\bar{u}_1, \dots, \bar{u}_n)$. Cum $u_s = \bar{u}_s = 0$, rezultă că există $i \neq s$ astfel încît $u_i \neq \bar{u}_i$ și $\forall j \in V(D_{si}), j \neq i, u_j = \bar{u}_j$, unde D_{si} este drumul construit la (a) pentru \bar{u}_i . Atunci avem:

$$\begin{aligned} u_i &> \bar{u}_i = \bar{u}_{i_1} + a_{i_1 i} = u_{i_1} + a_{i_1 i} \\ &\text{(din alegerea lui } i) \\ &\geq u_i \quad \text{pentru că } u_i \text{ satisface (*).} \end{aligned}$$

Contradicția găsită arată că $u = \bar{u}$, deci că u_i reprezintă costuri de drumuri minime.

Observații 1. Din demonstrație rezultă că pentru rezolvarea problemei P2 este suficient să obținem o soluție a sistemului (*). Drumurile corespunzătoare se obțin ca la (a).

Algoritmii pe care îi vom prezenta se vor ocupa de rezolvarea sistemului (*). Totuși, dacă avem $u_i = u_k + a_{ki}$ atunci așa cum am văzut, k este vârful dinaintea lui i de pe drumul minim de la s la i de cost u_i .

Rezultă că dacă în algoritmul de rezolvare a lui (*) construim un tablou *înainte*[1.. n] cu componente din $V \cup \{0, \}$ cu interpretarea finală "*înainte*[i]=vârful dinaintea lui i de pe drumul minim de la s la i ", atunci vîrfurile acestui drum pot fi determinate în $O(n)$ construind șirul i , *înainte*[i], *înainte*[*înainte*[i]],... pînă se depistează vârful s .

2. Dacă algoritmii de rezolvare a lui (*) vor evita (prin modul de actualizare a vectorului *înainte*) apariția circuitelor de cost total 0, atunci se observă că,

deși nu mai are loc unicitatea soluției sistemului (*), problema (P2) este rezolvată. Rezultă că acești algoritmi vor rezolva problema (P2) în condiția

$$(I') \quad \forall C \text{ circuit în } G, a(C) \geq 0.$$

3. În cazul grafurilor, rezolvarea problemelor (P1)-(P3) corespunzătoare se poate face utilizând algoritmi pentru digrafuri, prin înlocuirea fiecărei muchii cu o pereche de arce simetrice de același cost ca și muchia pe care o înlocuiesc.

Dificultatea unei astfel de abordări rezultă din introducerea pentru muchii de cost negativ a unor circuite de lungime 2 de cost negativ.

Deci, în cazul grafurilor, algoritmi pentru digrafuri sînt valabili doar dacă toate costurile sînt nenegative.

4. Avînd în vedere că mulțimile \mathcal{D}_{ij} sînt finite, se pot considera probleme analoge problemelor (P1)-(P3) înlocuind min cu max.

Utilizarea ideii uzuale,

$$\max_{x \in A} x = -(\min_{x \in A} (-x))$$

prin înlocuirea costurilor a_{ij} cu $-a_{ij}$ este posibilă doar în cazul digrafurilor în care pentru orice circuit C avem $a(C) \leq 0$.

În particular, această abordare este posibilă în cazul digrafurilor fără circuite (ca în aplicațiile b) și c) prezentate).

Dacă digraful inițial are circuite, problemele de drum de cost maxim se pot dovedi ușor (prin reducerea polinomială la probleme hamiltoniene) a fi NP -dificile.

Rezolvarea problemei (P2) în cazul digrafurilor fără circuite

O **numerotare aciclică** a (vîrfurilor) digrafului $G = (V, E)$ este un vector $ord[v] \quad v \in V$, (cu interpretarea $ord[v] = \text{numărul de ordine al vîrfului } v$) astfel încît

$$\forall vw \in E \Rightarrow ord[v] < ord[w].$$

Are loc următoarea

Lemă. *G este un digraf fără circuite dacă și numai dacă admite o numerotare aciclică .*

Demonstrație. Este evident că dacă G admite o numerotare aciclică atunci G nu are circuite (dacă $v_1, v_2, \dots, v_k, v_1$ sînt vîrfurile unui circuit atunci, cum G are o numerotare aciclică, obținem $ord[v_1] < ord[v_2] < \dots < ord[v_k] < ord[v_1]$, contradicție).

Reciproc, dacă G nu are circuite atunci există un vîrf $v_0 \in V$ astfel încît $d_G^-(v_0) = 0$ (altfel, datorită finitudinii digrafului, se poate construi un circuit); punem $ord[v_0] \leftarrow 1$, considerăm $G \leftarrow G - v_0$ și repetăm raționamentul (proprietatea de a nu avea circuite se transmite la subdigrafuri induse).

Aflarea unei numerotări aciclice a unui digraf se numește și **sortare topologică** întrucît se sortează mulțimea V într-un mod compatibil cu "topologia" digrafului.

Vom presupune că digraful este reprezentat cu ajutorul listelor de adiacență. Dimensiunea problemei este $O(n + e)$.

Vom construi un algoritm care să rezolve problema în timp $O(n + e)$.

Acest lucru este posibil datorită unei utilizări judicioase a structurilor de date.

Linia algoritmului:

- determinăm gradele interioare ale vîrfurilor, parcurgând toate listele de adiacență (la întîlnirea lui w în lista de adiacență a unui vîrf oarecare v se execută $d_G^-(w)++$;
- Parcurgem vectorul d_G^- și vîrfurile de grad interior 0 le memorăm într-o stivă S_0 ;

(a)- scoatem vîrfurile din stivă și le numerotăm;

(b)- scădem 1 din gradele interioare ale vîrfurilor din lista de adiacență a vîrfurilor tocmai numerotate ("îl scoatem din digraf") ;

(c)- în modificarea anterioară, crearea unui vîrf de grad interior 0 va implica memorarea lui în stivă S_0 ;

(d)- reluăm secvența (a) – (c) pînă cînd stiva devine vidă.

Dacă nu s-au numerotat toate vîrfurile rezultă că digraful conține circuite; în cazul epuizării vîrfurilor, s-a obținut numerotarea aciclică dorită (sortarea topologică).

Revenim la rezolvarea problemei P2.

În acest caz, după sortarea topologică a digrafului, vom că considera vârfurile sunt ordonate conform numerotării aciclice (vârful 1 numerotat cu 1, vârful 2 numerotat cu 2 etc.) și în plus vârful s din enunțul problemei **P2** este vârful 1 (pentru că nu vor exista drumuri de la vârful s la vârfuri cu număr de ordine mai mic decât el). În aceste ipoteze sistemul (*) se poate rezolva prin substituție:

1. Sortează topologic G ; $\{O(n + e)$ operații $\}$
2. $u_1 \leftarrow 0$; $\hat{inainte}[1] \leftarrow 0$;
 for $i := 2$ **to** n **do**
 { $u_i \leftarrow \infty$; $\hat{inainte}[i] \leftarrow 0$;
 for $j := 1$ **to** $i - 1$ **do**
 if $u_i > u_j + a_{ji}$ **then**
 { $u_i \leftarrow u_j + a_{ji}$;
 $\hat{inainte}[i] \leftarrow j$
 }
 }
 }

Complexitatea pasului 2 este, evident $O(1 + 2 + \dots + n - 1) = O(n^2)$.

Rezolvarea problemei (P2) în cazul costurilor nenegative.



Algoritmul lui Dijkstra

Dacă $a_{ij} \geq 0 \ \forall ij \in E$, atunci condiția (I') este îndeplinită și o soluție a sistemului (*) se poate obține cu ajutorul următorului algoritm (Dijkstra, 1961).

Se consideră $S \subseteq V$ astfel încât pe parcursul algoritmului are loc

(D) :

$$\begin{cases} \forall i \in S & u_i = \min\{a(D_{si}) \mid D_{si} \in \mathcal{D}_{si}\} \\ \forall i \in V \setminus S & u_i = \min\{a(D_{si}) \mid D_{si} \in \mathcal{D}_{si}, V(D_{si}) \setminus S = \{i\}\} \end{cases}$$

Dacă se reușește construirea lui S astfel încât $S = V$, atunci problema e rezolvată.

Inițial, se va considera $S = \{s\}$ și în $n - 1$ pași se adaugă la S câte un nou vîrf din V .

Algoritmul lui Dijkstra

1. $S \leftarrow \{s\}; u_s \leftarrow 0; \text{înainte}[s] \leftarrow 0;$
 for $i \in V \setminus \{s\}$ **do**
 { $u_i \leftarrow a_{si}; \text{înainte}[i] \leftarrow s$ }
 // după aceste inițializări (D) are loc
2. **while** $S \neq V$ **do**
 {
 determină $j^* \in V \setminus S : u_{j^*} = \min\{u_j \mid j \in V \setminus S\};$
 $S \leftarrow S \cup \{j^*\};$
 for $j \in V \setminus S$ **do**
 if $u_j > u_{j^*} + a_{j^*j}$ **then**
 { $u_j \leftarrow u_{j^*} + a_{j^*j}; \text{înainte}[j] \leftarrow j^*$ }
 }

Corectitudinea algoritmului va rezulta dacă vom arăta că, dacă înaintea unei iterații din pasul 2 are loc (D), atunci, după execuția acelei iterații, (D) are loc de asemenea.

Arătăm mai întâi că în ipoteza că (D) are loc, atunci adăugarea lui j^* la S nu contrazice (D). Deci trebuie dovedit că dacă $u_{j^*} = \min\{u_j \mid j \in V \setminus S\}$ atunci $u_{j^*} = \min\{a(D_{sj^*}) \mid D_{sj^*} \in \mathcal{D}_{sj^*}\}.$

Presupunem că există $D_{sj^*}^1 \in \mathcal{D}_{sj^*}$ astfel încît $a(D_{sj^*}^1) < u_{j^*}$.

Cum S satisface (D), avem

$u_{j^*} = \min\{a(D_{sj^*}) \mid D_{sj^*} \in \mathcal{D}_{sj^*}, V(D_{sj^*}) \setminus S = \{j^*\}\}$. Rezultă că $V(D_{sj^*}^1) \setminus S \neq \{j^*\}$.

Fie k primul vîrf al drumului $D_{sj^*}^1$ (în parcurgerea sa din s) astfel încît $k \notin S$.

Atunci $a(D_{sj^*}^1) = a(D_{sk}^1) + a(D_{kj^*}^1)$.

Din alegerea lui k , avem $V(D_{sk}^1) \setminus S = \{k\}$ și cum (D) are loc, avem $a(D_{sk}^1) = u_k$. Obținem $u_{j^*} > a(D_{sj^*}^1) = u_k + a(D_{kj^*}^1) \geq u_k$ (costurile sînt nenegative), ceea ce contrazice alegerea lui j^* . Contradicția obținută arată că, după atribuirea $S := S \cup \{j^*\}$, prima parte a condiției (D) are loc.

Pentru ca și cea de-a doua parte a condiției (D) să aibă loc după această atribuire, să observăm că $\forall j \in V \setminus (S \cup \{j^*\})$ avem

$\min\{a(D_{sj}) \mid D_{sj} \in \mathcal{D}_{sj}, V(D_{sj}) \setminus (S \cup \{j^*\}) = \{j\}\} = \min\left(\min\{a(D_{sj}) \mid D_{sj} \in \mathcal{D}_{sj}, V(D_{sj}) \setminus S = \{j\}\}, \min\{a(D_{sj}) \mid D_{sj} \in \mathcal{D}_{sj}, V(D_{sj}) \setminus S = \{j, j^*\}\}\right)$.

Cum (D) are loc, primul din cele două minime de mai sus este u_j . Fie α_j valoarea celui de-al doilea minim și fie D_{sj}^1 drumul pentru care se realizează. Cum $j^* \in V(D_{sj}^1)$, avem $\alpha_j = a(D_{sj^*}^1) + a(D_{j^*j}^1)$.

Întrucît $S \cup \{j^*\}$ satisface prima parte a lui (D), avem $a(D_{sj^*}^1) = u_{j^*}$ (altfel s-ar contrazice alegerea lui D_{sj}^1 înlocuind în D_{sj}^1 porțiunea $D_{sj^*}^1$ cu un drum de cost mai mic). Deci $\alpha_j = u_{j^*} + a(D_{j^*j}^1)$.

Dacă drumul $D_{j^*j}^1$ este de lungime 1 atunci avem $\alpha_j = u_j + a_{j^*j}$.

Altfel, considerînd k vîrfurile dinaintea lui j de pe drumul D_{sj}^1 avem $k \neq j^*, k \in S$ și $\alpha_j = a(D_{sk}^1) + a_{kj}$. Cum $S \cup \{j^*\}$ satisface prima parte a lui (D), obținem $\alpha_j = u_k + a_{kj}$.

Întrucît S satisface (D), u_k este costul unui drum minim de la s la k cu vîrfurile conținute în S deci α_j este costul unui drum de la s la j cu vîrfurile conținute în S . Rezultă că $\alpha_j \geq u_j$, căci S satisface (D).

Am obținut că singurul caz în care $\alpha_j < u_j$ este atunci când $\alpha_j = u_{j^*} + a_{j^*j} < u_j$, situație testată în ciclul **for** al pasului 2.

Rezultă că (D) are loc pe tot parcursul algoritmului și deci valorile finale ale variabilelor u_i reprezintă soluția sistemului (*). Evident, tabloul *înainte* este actualizat pentru memorarea implicită a drumurilor de cost minim.

Complexitatea timp a algoritmului, în descrierea dată este $O(n^2)$ datorită selectării minimelor din pasul 2.

Este posibilă organizarea unor cozi cu prioritate (de exemplu heap-urile) pentru a memora valorile u_i , $i \in U = V \setminus S$, astfel încât extragerea minimului să se facă în $O(1)$, iar actualizările necesare în pasul 2 să se facă în timpul total de $O(m \log n)$ unde $m = |E|$ (executându-se $O(m)$ descreșteri de valori u_i , fiecare necesitând $O(\log n)$ operații; *Johnson ,1977*).

Cea mai bună implementare se obține utilizând **heap-uri Fibonacci**, ceea ce conduce la o complexitate timp de $O(m + n \log n)$ (*Fredman și Tarjan, 1984*).

Opădure cu rădăcini (*rooted forest*) este un **digraf aciclic** $D = (V, A)$ cu proprietatea că **fiecare vârf are gradul interior cel mult 1**. Vârfurile de grad interior 0 sunt *rădăcinile* lui D , iar cele cu grad exterior 0 sunt *frunzele* lui D .

Dacă $uv \in A$ atunci u este *părintele* lui v iar v este *copilul* lui u .

Dacă pădurea are o singură rădăcină, atunci ea este un *arbore cu rădăcină*.

O **pădure Fibonacci** este o pădure cu rădăcini $F = (V, A)$ în care copiii fiecărui vârf v pot fi ordonați astfel încât **copilul numărul i are la rândul său cel puțin $i - 2$ copii**.

Teoremă. Într-o pădure Fibonacci $F = (V, A)$ fiecare vârf are cel mult $1 + 2 \log |V|$ copii.

Dem. Notăm cu $\sigma(v)$ numărul vârfurilor accesibile din v în F (ordinul subarborelui cu rădăcina v).

Arătăm că $\sigma(v) \geq 2^{(d^+(v)-1)/2}$, care va implica prin logaritmare afirmația din enunțul teoremei.

Se observă că inegalitatea precedentă are loc pentru v frunză, așa că utilizăm un raționament inductiv.

Fie $k = d^+(v)$ și fie v_i copilul numărul i al lui v ($i = 1, \dots, k$).

Avem, $\sigma(v_i) \geq 2^{(d^+(v_i)-1)/2} \geq 2^{(i-1)/2}$, întrucât $d^+(v_i) \geq i - 2$.

Deci $\sigma(v) = 1 + \sum_{i=1}^k \sigma(v_i) \geq 1 + \sum_{i=1}^k 2^{(i-3)/2} \geq \dots \geq 2^{(k-1)/2}$, și teorema e demonstrată.

Un **heap Fibonacci** conținând valorile reale $(u_j; j \in U)$ este o pădure Fibonacci $F = (U, A)$ (fiecare vârf j are ordonați copii astfel încât copilul numărul i are cel puțin $i-2$ copii) în care este precizată o mulțime $T \subset U$ astfel încât:

- (i) dacă $jk \in A$ atunci $u_j \leq u_k$;
- (ii) dacă h este copilul numărul i al lui j și $h \notin T$ atunci h are cel puțin $i - 1$ copii;
- (iii) dacă j_1 și j_2 sunt două rădăcini distincte atunci $d^+(j_1) \neq d^+(j_2)$.

Teorema anterioară ne asigură că numărul rădăcinilor nu va depăși $2 + 2 \log |U|$.

Heapul Fibonacci va fi reprezentat cu ajutorul următoarei structuri de date:

- câte o listă dublu înlănțuită C_j a copiilor fiecărui $j \in U$;
- funcția $p : U \rightarrow U$, unde $p(j) =$ părintele lui j (dacă j e rădăcină $p(j) = j$);

- funcția $d^+ : U \rightarrow \mathbb{N}$;
- funcția $b : \{0, \dots, t\} \rightarrow U$ (cu $t = 1 + \lceil 2 \log |U| \rceil$) cu proprietatea că $b(d^+(j)) = j$ pentru fiecare rădăcină j ;
- funcția $l : U \rightarrow \{0, 1\}$ cu $l(j) = 1$ dacă și numai dacă $j \in T$.

Teoremă. Pentru găsirea și ștergerea de n ori a unui j care minimizează u_j și descreșterea de m ori a unei valori u_j , structura de date poate fi actualizată în timpul $O(m + p + n \log p)$, unde p este numărul de vârfuri din pădurea inițială.

Dem. Pentru găsirea unui j care minimizează u_j este suficient să parcurgem $u_{b(i)}$ pentru $i = 0, \dots, t$, deci în $O(\log p)$. Un astfel de element j (cu u_j minim) se poate șterge astfel:

- fie v_1, \dots, v_k copii lui j ;
- ștergem j și arcele ce ies din j din pădure;
- acum v_1, \dots, v_k au devenit rădăcini, iar condițiile (i) și (ii) nu-s afectate;

-pentru repararea condițiilor (iii) se execută pentru fiecare $r = v_1, \dots, v_k$:
repară(r): dacă $\exists s$ rădăcină cu $d^+(r) = d^+(s)$
 atunci: dacă $u_r \leq u_s$, adaugă s ca ultim copil al lui r și *repară(r)*, altfel ($u_r > u_s$), adaugă r ca ultim copil al lui s și *repară(s)*.

În acest fel condițiile (i) și (iii) sunt menținute, iar existența rădăcinii s de mai sus, se face cu ajutorul funcțiilor b, d și p (în timpul procesului, se actualizează structura de date).

Descrescerea unei valori u_j pentru un $j \in U$ se face astfel:

declară rădăcină(j):

dacă j are un părinte, fie acesta v , atunci se șterge arcul vj și se aplică *repară(r)*;

dacă $v \notin T$ se adaugă v la T , altfel se scoate v din T și se aplică *declară rădăcină(v)*:

Notăm cu $incr(..)$ și $decr(..)$ numărul creșterilor, respectiv descreșterilor lui $..$ în timpul operațiilor din enunțul teoremei. Avem:

numărul de apeluri ale lui *declară rădăcină* =

$$decr(u_j) + decr(T) \leq$$

$$\leq decr(u_j) + incr(T) + p \leq 2decr(u_j) + p = 2m + p,$$

deoarece creștem T cel mult o dată după ce a descrescut un u_j .

Dacă R este mulțimea rădăcinilor, avem:

numărul de apeluri ale lui *repară* =

$$decr(A) + decr(T) \leq$$

$$\leq decr(A) + incr(R) + p = 2decr(A) + p$$

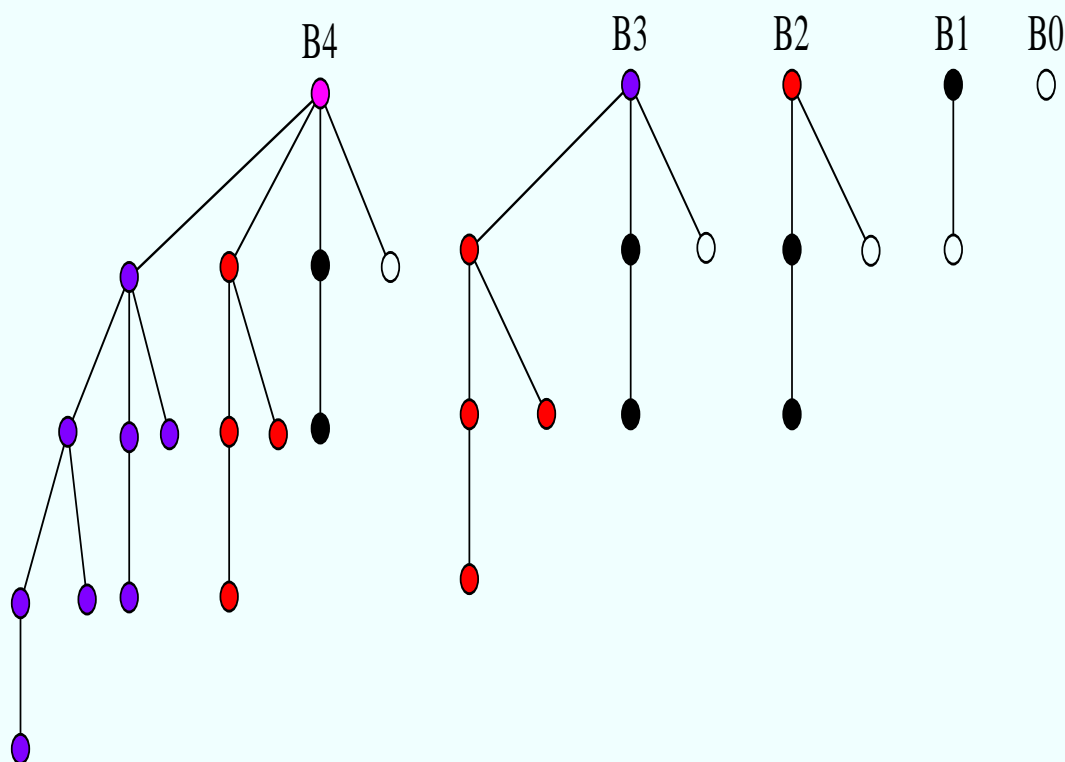
$$\leq 2(n \log p + \text{numărul de apeluri ale lui } declară\text{ rădăcină}) + p$$

$$\leq 2(n \log p + 2m + p) + p.$$

Cum decizia dacă să se apeleze una sau alta dintre cele două funcții se face în $O(1)$, rezultă că algoritmul are complexitatea $O(m + p + n \log p)$ și teorema e demonstrată.

Corolar. Algoritmul lui Dijkstra pentru rezolvarea problemei **P2** se poate implementa cu ajutorul heap-urilor Fibonacci în complexitatea timp $O(m + n \log n)$.

Demonstrația rezultă din teorema precedentă și din următoarea figură care indică un mod de construcție a heap-ului inițial (binomial):



Dacă se dorește rezolvarea problemei (P1) cu ajutorul algoritmului lui Dijkstra, atunci, la introducerea lui t în S , se poate opri algoritmul. Complexitatea, în cazul cel mai nefavorabil, rămîne aceeași. Totuși, în situații practice concrete există posibilitatea de a grăbi introducerea lui t în S utilizînd o funcție de dirijare a procesului de construcție a lui S .

O funcție $g : V \rightarrow \mathbf{R}_+$ se numește **estimator consistent** dacă

- (i) $\forall i \in V \quad u_i + g(i) \leq \min\{a(D_{st}) \mid D_{st} \in \mathcal{D}_{st} \text{ și } i \in V(D_{st})\};$
- (ii) $\forall ij \in E \quad g(i) \leq a_{ij} + g(j).$

Să observăm că $g(i) = 0 \quad \forall i$ este un estimator consistent (trivial).

Dacă însă $V(G)$ este o mulțime de puncte din plan, atunci $g(i)$ =distanța (euclidiană) de la i la t este un estimator consistent, dacă sînt satisfăcute condițiile (ii).

Dacă g este un estimator consistent atunci se poate modifica alegerea lui j^* în algoritm astfel: $u_{j^*} + g(j^*) = \min\{u_j + g(j) \mid j \in V \setminus S\}$.

Algoritmul rămîne valabil (demonstrația este identică situației $g(i) = 0 \ \forall i$ și se folosește (ii) repetat).

Avantajul este acela că se vor introduce în S vîrfuri care sã ne apropie de t .

În implementarea care rezultã din descrierea algoritmului lui Dijkstra, s-a presupus cã se dispune de matricea de cost-adiacență a digrafului. În cazul digrafurilor cu multe vîrfuri (în care, de exemplu, listele de adiacență sînt memorate în memoria secundarã), sau în cazul digrafurilor date funcțional (se dispune de o procedurã care construiește pentru un vîrf dat, lista sa de adiacență) aceastã implementare este neeficientã, respectiv neaplicabilã. O implementare care nu are aceste deficiențe este următoarea datoratã lui *Glover, Klingman și Philips (1985)*

Partition Shortest Path (PSP) algorithm:

1. $u_s \leftarrow 0$; $\hat{inainte}(s) \leftarrow 0$;
 $S \leftarrow \emptyset$; $NOW \leftarrow \{s\}$; $NEXT \leftarrow \emptyset$;
2. **while** $NOW \cup NEXT \neq \emptyset$ **do**
 { **while** $NOW \neq \emptyset$ **do**
 { Extrage i din NOW ;
 $S \leftarrow S \cup \{i\}$;
 $L \leftarrow N_G^+(i)$; // se generează în L , lista de
 adiacență și costurile corespunzătoare
 for $j \in L$ **do**
 if $j \notin NOW \cup NEXT$ **then**
 { $u_j \leftarrow u_i + a_{ij}$; $\hat{inainte}(j) \leftarrow i$;
 introdu j în $NEXT$
 }
 else if $u_j > u_i + a_{ij}$ **then**
 { $u_j \leftarrow u_i + a_{ij}$
 $\hat{inainte}(j) \leftarrow i$
 }
 }
 if $NEXT \neq \emptyset$ **then**
 { determină $d = \min\{u_i \mid i \in NEXT\}$;
 transferă $\forall i \in NEXT$ cu $u_i = d$ în NOW
 }
 }

Rezolvarea problemei (P2) în cazul general.

Dacă există $ij \in E$ astfel încât $a_{ij} < 0$, algoritmul lui Dijkstra nu mai este valabil în general (introducerea lui j^* în S poate conduce la violarea condiției (D)).

Considerând îndeplinită condiția (I') vom rezolva sistemul (*) prin aproximații succesive.

Considerăm $\forall i \in V$ și $\forall m = \overline{1, n-1}$
(BM)

$$u_i^m = \min\{a(D) \mid D \in \mathcal{D}_{si}, \text{nr arcelor lui } D \text{ este } \leq m\}.$$

Cum orice drum în G are cel mult $n-1$ arce rezultă că dacă reușim construcția lui

$u^1 = (u_1^1, \dots, u_n^1)$, $u^2 = (u_1^2, \dots, u_n^2)$, ..., $u^{n-1} = (u_1^{n-1}, u_2^{n-1}, \dots, u_n^{n-1})$, atunci u^{n-1} este soluția sistemului (*).

Algoritmul care rezultă este următorul:

Algoritmul lui *Bellman, Ford, Moore* (~ 1960)

1. $u_s^1 \leftarrow 0$; **for** $i \in V \setminus \{s\}$ **do** $u_i^1 \leftarrow a_{si}$;
 // evident (BM) are loc
2. **for** $m := 1$ **to** $n - 2$ **do**
 for $i := 1$ **to** n **do**
 $u_i^{m+1} \leftarrow \min(u_i^m, \min_{j \neq i}(u_j^m + a_{ji}))$

Pentru a demonstra corectitudinea algoritmului, arătăm că dacă $u^m (m \geq 1)$ satisface (BM) atunci și u^{m+1} o satisface. Fie $i \in V$ și considerăm mulțimile de drumuri:

$$A = \{D \mid D \in \mathcal{D}_{si}, \text{numărul arcelor lui } D \leq m+1\}.$$

$$B = \{D \mid D \in \mathcal{D}_{si}, \text{numărul arcelor lui } D \leq m\}.$$

$$C = \{D \mid D \in \mathcal{D}_{si}, \text{numărul arcelor lui } D = m+1\}.$$

Atunci $A = B \cup C$ și

$$\min\{a(D) \mid D \in A\} = \min(\min\{a(D) \mid D \in B\}, \min\{a(D) \mid D \in C\})$$

Cum u^m satisfac (BM), rezultă că $\min\{a(D) \mid D \in A\} = \min(u_i^m, \min\{a(D) \mid D \in C\})$.

Fie $\min\{a(D) \mid D \in C\} = a(D^0)$, $D^0 \in C$.

Dacă j este vârful ce-l precede pe i în D^0 (există, întrucât D^0 are măcar 2 arce) atunci

$$a(D^0) = a(D_{sj}^0) + a_{ji} \geq u_j^m + a_{ji} ,$$

întrucât D_{sj}^0 are m arce și u^m satisface (BM).

Rezultă că

$$\min\{a(D) \mid D \in A\} = \min\{u_i^m, \min_{j \neq i}(u_j^m + a_{ji})\} \text{ valoare care în algoritm se atribuie lui } u_i^{m+1} .$$

Observăm că algoritmul are complexitatea de $O(n^3)$ dacă determinarea minimului din pasul 2 necesită $O(n)$ operații.

Determinarea drumurilor minime se face menținând vectorul *înainte*, inițializat în mod evident în pasul 1 și actualizat corespunzător, la stabilirea minimului din pasul 2.

Observații:

1. Dacă la algoritm se adaugă și pasul 3:

3. **if** ($\exists i \in V$ a.î. $u_i^{n-1} > \min_{j \neq i}(u_j^{n-1} + a_{ji})$)
then "există circuit de cost negativ".

se obține posibilitatea testării în $O(n^3)$ a existenței unui circuit C de cost negativ în digraful G (altfel, din demonstrația corectitudinii algoritmului ar trebui să nu se poată micșora u_i^{n-1}).

Depistarea circuitului C se face simplu ($O(n)$) utilizând vectorul *înainte*.

2. Dacă există $k < n - 1$ astfel încât $u^k = u^{k+1}$ atunci algoritmul se poate opri. Mai mult, se poate obține o implementare a acestui algoritm, care să aibă complexitatea $O(nm)$, folosind o coadă UQ în care se vor păstra vîrfurile i cărora li se modifică u_i curent (se va renunța, evident, la memorarea tuturor aproximațiilor succesive).

Rezolvarea problemei (P3).

Considerăm

$$u_{ij} = \min\{a(D_{ij}) \mid D_{ij} \in \mathcal{D}_{ij}\} \quad \forall i, j \in V.$$

Problema se reduce la determinarea matricii $U = (u_{ij})_{n \times n}$, atunci cînd se cunoaște A matricea de cost-adiacentă.

Drumurile de cost minim vor fi obținute în $O(n)$ dacă odată cu determinarea matricii U se va construi matricea

$\hat{Inainte} = (\hat{inainte}(i, j))_{n \times n}$ cu elementele avînd semnificația

$\hat{inainte}(i, j) = \text{vîrful dinaintea lui } j \text{ de pe drumul de cost minim de la } i \text{ la } j \text{ în } G.$

Să observăm că dacă $a_{ij} \geq 0 \quad \forall ij$, atunci, iterînd algoritmul lui Dijkstra pentru $s \in \{1, \dots, n\}$, se obține un algoritm de complexitate $O(n^3)$.

Dacă G nu conține circuite de cost negativ, dar există și arce de cost negativ, iterînd algoritmul lui Bellman Ford pentru $s = \overline{1, n}$ se obține un algoritm de complexitate $O(n^4)$.

Arătăm în continuare că se poate proceda și mai eficient.

Soluția I^a .

Fie $\alpha : V \rightarrow \mathbf{R}$ a. î. $\forall ij \in E \ \alpha(i) + a_{ij} \geq \alpha(j)$.

Considerăm $\bar{a} : E \rightarrow \mathbf{R}_+$ dată de

$$\bar{a}_{ij} = a_{ij} + \alpha(i) - \alpha(j), \ \forall ij \in E.$$

Avem $\bar{a}_{ij} \geq 0$ și, în plus, oricare ar fi $D_{ij} \in \mathcal{D}_{ij}$,

$$(2) \quad \bar{a}(D_{ij}) = a(D) + \alpha(i) - \alpha(j).$$

Rezultă că se poate itera algoritmul lui Dijkstra pentru obținerea drumurilor de cost \bar{a} minim și din relația (2) se observă că un drum este de cost \bar{a} minim dacă și numai dacă este drum de cost a minim. Rezultă următorul algoritm:

1. Determină α și construiește \overline{A} .
2. Rezolvă (P3) pt. \overline{A} construind \overline{U} și *Înainte*.
3. Determină U ($u_{ij} := \overline{u}_{ij} - \alpha(i) + \alpha(j) \forall ij$).

Pasul 2 al algoritmului necesită $O(n)^3$) operații prin iterarea algoritmului lui Dijkstra.

Pasul 1 se poate realiza în timpul $O(n^3)$, fixând $s \in V$ și rezolvând (P2) cu alg. Bellman-Ford.

În adevăr, dacă $(u_i, i \in V)$ este soluție a lui (P2), atunci $(u_j, j \in V)$ este soluție a sistemului $(*) \Rightarrow u_j = \min_{i \neq j} \{u_i + a_{ij}\}$, adică $\forall ij \in E$ $u_j \geq u_i + a_{ij}$, sau, $a_{ij} + u_i - u_j \geq 0$.

Deci, se poate considera $\alpha(i) = u_i \forall i \in V$.

Soluția a II^a .

Fie

$$u_{ij}^m = \min\{a(D_{ij}) \mid D_{ij} \in \mathcal{D}_{ij}, V(D_{ij}) \setminus \{i, j\} \subseteq \{1, 2, \dots, m-1\}\} \forall i, j \in \{1, 2, \dots, n\}, m = \overline{1, n+1}.$$

Atunci, evident $u_{ij}^1 = a_{ij} \forall i, j \in V$ (presupunem matricea A având elementele diagonale egale cu 0). În plus,

$$u_{ij}^{m+1} = \min\{u_{ij}^m, u_{im}^m + u_{mj}^m\} \forall i, j \in V, m = 1, \dots, n.$$

Această ultimă relație se poate justifica inductiv: un drum de cost minim de la i la j care nu are vîrfuri interioare $\geq m$ poate să nu conțină vîrfurile m , și atunci are costul u_{ij}^m , sau poate conține vîrfurile m , și atunci, din principiul optimalității al lui Bellman și ipoteza inductivă, este $u_{im}^m + u_{mj}^m$.

Evident, dacă se obține $u_{ii}^m < 0$ atunci digraficul conține un circuit de cost negativ C care trece prin vîrfurile i , cu $V(C) \setminus \{i\} \subseteq \{1, \dots, m-1\}$.

Această soluție a problemei (P3) este cunoscută ca algoritmul lui **Floyd-Warshall** și poate fi descris astfel:

```

1:  for  $i := 1$  to  $n$  do
    for  $j := 1$  to  $n$  do
        {    $\hat{inainte}(i, j) \leftarrow i$ ;
            if  $i = j$  then {  $a_{ii} \leftarrow 0$ ;  $\hat{inainte}(i, i) \leftarrow 0$  }
        }

2:  for  $m := 1$  to  $n$  do
    for  $i := 1$  to  $n$  do
        for  $j := 1$  to  $n$  do
            if  $a_{ij} > a_{im} + a_{mj}$  then
                {    $a_{ij} \leftarrow a_{im} + a_{mj}$ ;
                     $\hat{inainte}(i, j) \leftarrow \hat{inainte}(m, j)$ 
                    if  $(i = j \wedge a_{ij} < 0)$  then
                        return "circuit negativ"
                }

```

Evident, complexitatea algoritmului este de $O(n^3)$.

Observație. Dacă digraful nu conține circuite de cost negativ, atunci inițializând $a_{ii} \leftarrow \infty$, valorile finale ale elementelor diagonale dau costul minim al unui circuit ce trece prin vârful corespunzător.

Soluția a III^a. Considerăm $a_{ii} = 0 \ \forall i \in V$ (G nu conține circuite de cost < 0).

Iterarea algoritmului lui Bellman Ford corespunde următoarei abordări. Fie

$$u_{ij}^m = \min\{a(D_{ij}) \mid D_{ij} \in \mathcal{D}_{ij}, D_{ij} \text{ are cel mult } m \text{ arce}\} \\ \forall i, j \in V, \forall m = 1, 2, \dots, n - 1.$$

Dacă notăm $U^m = (u_{ij}^m)$ cu $m \in \{0, 1, 2, \dots, n - 1\}$, unde U^0 are toate elementele ∞ cu excepția celor de pe diagonală care-s egale cu 0 atunci, iterarea algoritmului lui Bellman Ford revine la:

1. **for** $i, j \in V$ **do** $u_{ij}^0 \leftarrow \infty$ **if** $i \neq j$ **else** 0;
2. **for** $m := 0$ **to** $n - 2$ **do**
 for $i, j \in V$ **do** $u_{ij}^{m+1} = \min_k (u_{ik}^m + a_{kj})$;

(în minimul anterior, comparația cu u_{ij}^m din algoritmul Bellman Ford, se realizează pentru $k = j$, și utilizând ipoteza că $a_{jj} = 0$). Întregul proces de calcul se poate rescrie matricial dacă se consideră următorul produs pe mulțimea matricilor pătrate cu elemente reale:

$$\forall B, C \in \mathcal{M}_{n \times n} \quad B \otimes C = P = (p_{ij})$$

unde, $p_{ij} = \min_{k=\overline{1,n}}(a_{ik} + b_{kj})$.

Se observă că, dacă se folosește determinarea uzuală a minimului, atunci calculul matricii P este similar înmulțirii uzuale a matricilor. În plus operația \otimes este asociativă.

Cu aceste notații avem

$U^{m+1} = U^m \otimes A$ și inductiv rezultă că

$$U^1 = A, U^2 = A^{(2)}, \dots, U^{n-1} = A^{(n-1)}$$

unde $A^{(k)} = A^{(k-1)} \otimes A$ și $A^{(1)} = A$.

În ipoteza că graful nu are circuite de cost negativ, atunci $A^{(2^k)} = A^{(n-1)} \quad \forall k : 2^k \geq n - 1$.

Rezultă că determinarea succesivă a matricilor $A, A^{(2)}, A^{(4)} = A^{(2)} \otimes A^{(2)}, \dots$ conduce la un algoritm de complexitate $O(n^3 \log n)$ pentru rezolvarea problemei (P3)

(desigur, matricea *Înainte* se va obține în $O(n^3)$ operații ca în demonstrația teoremei 1, după determinarea lui U^{n-1} .)

Dacă "produsul" matricial considerat se face cu algoritmi mai performanți atunci se obține o rezolvare eficientă a problemei (n^3 din evaluarea precedentă se poate înlocui cu $n^{\log_2 7} = n^{2,81}$ (*Strassen* 1969); sau chiar cu $n^{2,38}$ (*Coopersmith, Winograd* 1987)).

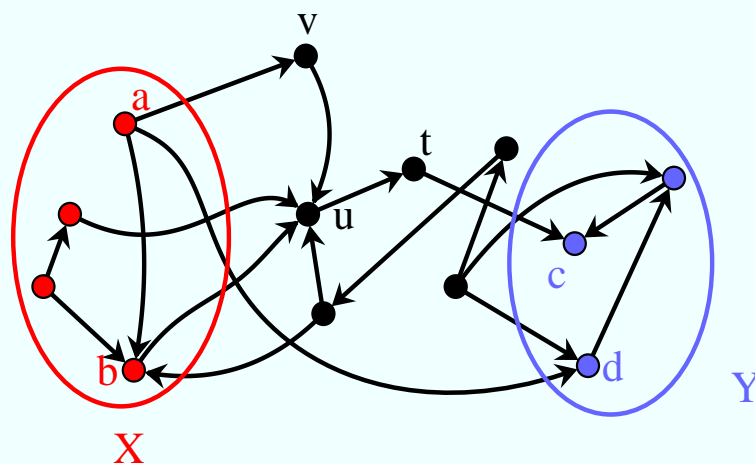
.....

3. Probleme de conexiune.

Teorema lui Menger și aplicații.

Definiție. Fie $G = (V, E)$ (di)graf și $X, Y \subseteq V$. Numim **XY -drum** în G orice drum D în G de la un vîrf $x \in X$ la un vîrf $y \in Y$, astfel încît $V(D) \cap X = \{x\}$ și $V(D) \cap Y = \{y\}$.

În figura alăturată, $D_1 : a, v, u, t, c$ și $D_2 : a, d$ sunt singurele XY -drumuri ce pornesc din a :



Vom nota cu $\mathcal{D}(X, Y; G)$ mulțimea tuturor XY -drumurilor în G .

Să observăm că dacă $x \in X \cap Y$ atunci drumul de lungime 0 $D = \{x\}$ este XY -drum.

Vom spune că drumurile D_1 și D_2 sînt disjuncte dacă $V(D_1) \cap V(D_2) = \emptyset$.

Probleme practice evidente, din rețelele de comunicație, dar și unele probleme legate de conexiunea grafurilor și digrafurilor, necesită determinarea unor mulțimi de XY -drumuri disjuncte și cu număr maxim de elemente.

Vom nota cu $p(X, Y; G)$ **numărul maxim de XY -drumuri disjuncte în (di)graful G .**

Teorema care precizează acest număr a fost stabilită de Menger în 1927 și constituie unul din rezultatele fundamentale din teoria grafurilor.

Definiție. Fie $G = (V, E)$ un digraf și $X, Y \subseteq V$. Numim **mulțime XY -separatoare** în G o mulțime $Z \subseteq V$ astfel încît $\forall D \in \mathcal{D}(X, Y; G) \Rightarrow V(D) \cap Z \neq \emptyset$.

Notăm cu

$$\mathcal{S}(X, Y; G) = \{Z \mid Z \text{ } XY\text{-separatoare în } G\}$$

și cu

$$k(X, Y; G) = \min\{|Z|; Z \in \mathcal{S}(X, Y; G)\}.$$

Din definiție, rezultă următoarele proprietăți imediate ale mulțimilor XY -separatoare:

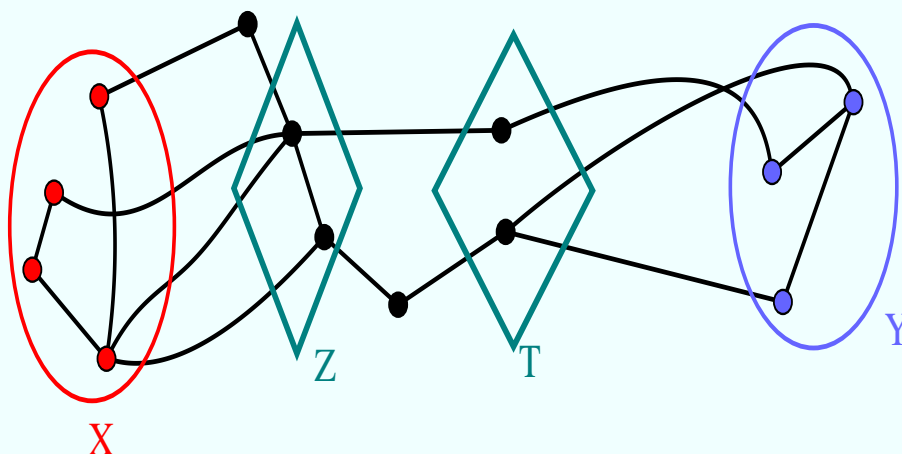
(a) Dacă $Z \in \mathcal{S}(X, Y; G)$ atunci $\forall D \in \mathcal{D}(X, Y; G)$ D nu este drum în $G - Z$.

(b) $X, Y \in \mathcal{S}(X, Y; G)$.

(c) Dacă $Z \in \mathcal{S}(X, Y; G)$ atunci $\forall A$ astfel încât $Z \subseteq A \subseteq V$ avem $A \in \mathcal{S}(X, Y; G)$.

(d) Dacă $Z \in \mathcal{S}(X, Y; G)$ și $T \in \mathcal{S}(X, Z; G)$ sau $T \in \mathcal{S}(Z, Y; G)$ atunci $T \in \mathcal{D}(X, Y; G)$.

Proprietatea (d) este esențială pentru obținerea teoremei următoare și este evidențiată mai jos



Teoremă. 1. Fie $G = (V, E)$ (di)graf și $X, Y \subseteq V$. Atunci

$$p(X, Y; G) = k(X, Y; G).$$

Demonstrație: 1^0 . Dacă $p = p(X, Y; G)$ și D_1, D_2, \dots, D_p sînt XY -drumuri disjuncte în G , atunci $\forall Z \in \mathcal{S}(X, Y; G)$ avem $Z \cap V(D_i) \neq \emptyset$ și cum D_i sînt disjuncte ($i = 1, p$):

$$|Z| \geq |Z \cap \cup_{i=1}^p V(D_i)| = \sum_{i=1, p} |Z \cap V(D_i)| \geq \sum_{i=1, p} 1 = p.$$

Deci $\forall Z \in \mathcal{S}(X, Y; G) \quad |Z| \geq p$; în particular $k(X, Y; G) \geq p(X, Y, G)$.

2^0 . Arătăm prin inducție după $a(G) = |V| + |E|$ că $\forall G = (V, E) \forall X, Y \subseteq V$

(*) $\exists k(X, Y; G)$ XY -drumuri disjuncte în G .

(Evident, din (*) rezultă că $p(X, Y; G) \geq k(X, Y; G)$ și deci, împreună cu 1^0 , teorema e demonstrată).

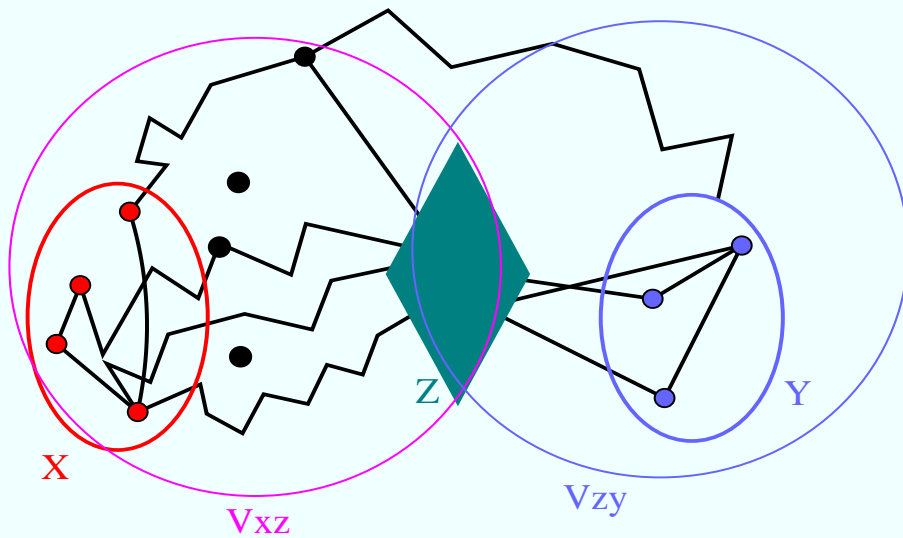
Cum (*) se verifică pentru (di)grafuri G cu $a(G) = 1, 2$, considerăm în pasul inductiv că (*) are loc pentru orice (di)graf G' și orice $X', Y' \subseteq V(G')$, cu $a(G') < a(G)$. Pentru a exclude cazurile banale, vom presupune că $X \not\subseteq Y$, $Y \not\subseteq X$ și $k = k(X, Y; G) > 0$.

Cazul 1. Există $Z \in \mathcal{S}(X, Y; G)$ astfel încât $|Z| = k$, $Z \neq X, Y$.

Considerăm $V_{XZ} = \{v \mid \exists D \in \mathcal{D}(X, Z; G) : v \in V(D)\}$ și $V_{ZY} = \{v \mid \exists D \in \mathcal{D}(Z, Y; G) : v \in V(D)\}$.

Să observăm că $V_{XZ} \cap V_{ZY} = Z$

(dacă există $v \in V_{XZ} \cap V_{ZY} - Z$, atunci se obține că Z nu este XY -separatoare; dacă există $z \in Z$ astfel încât $z \notin V_{XZ} \cap V_{ZY}$ atunci $Z - \{z\}$ este XY -separatoare, contrazicând $|Z| = k(X, Y; G)$).

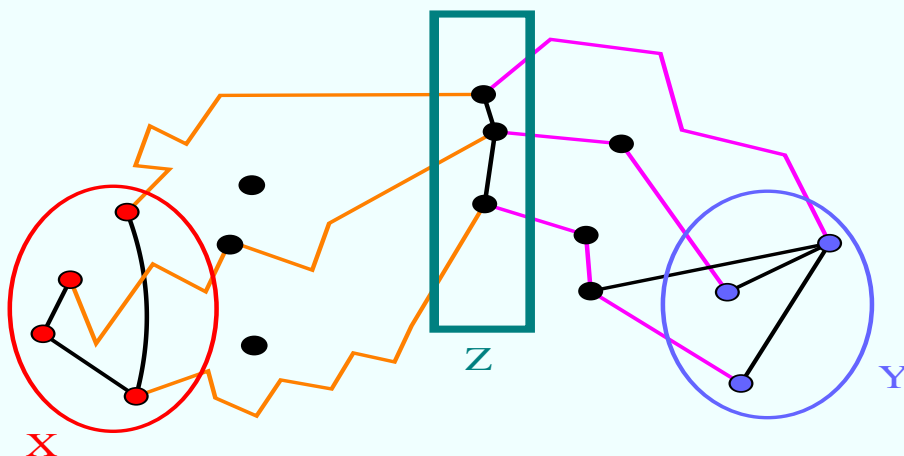


Pe de altă parte, există $x \in X - Z$ (dacă $X \subseteq Z$, atunci cum $X \in \mathcal{S}(X, Y; G)$ și $|Z| = k(X, Y; G)$ rezultă $X = Z$, contrazicând ipoteza cazului 1) și evident $x \notin V_{ZY}$ (altfel, Z nu ar fi XY -separatoare). Rezultă $|V_{ZY}| < |V|$. În mod similar $|V_{XZ}| < |V|$.

Fie $G_{XZ} = [V_{XZ}]_G$ și $G_{ZY} = [V_{ZY}]_G$. Din observațiile precedente: $a(G_{XZ}), a(G_{ZY}) < a(G)$.

Avem $k(X, Z; G_{XZ}) = k$ și $k(Z, Y; G_{ZY}) = k$ (Z este XZ -separatoare în G_{XZ} , respectiv ZY -separatoare în G_{ZY} și are cardinalul k ; dacă în unul din cele două grafuri, ar exista o mulțime T separatoare de cardinal $< k$, atunci, utilizând observația (d), se contrazice definiția lui k pentru G, X și Y).

Din ipoteza inductivă, rezultă că există k XZ -drumuri disjuncte în G_{XZ} și k ZY -drumuri disjuncte în G_{ZY} . Cum $V_{XZ} \cap V_{ZY} = Z$ și $|Z| = k$, rezultă că aceste $2k$ drumuri se pot concatena două câte două în G (vezi figura de mai jos) și deci (*) are loc.



Cazul 2. Oricare ar fi Z XY -separatoare astfel încît $|Z| = k$ avem $Z = X$ sau $Z = Y$.

Presupunem, pentru precizarea notațiilor, $Z = X$. Cum $X \not\subseteq Y$, există $x \in X - Y$. $X - \{x\}$ nu este XY -separatoare (are mai puțin de k elemente). Există deci un XY -drum în G . Fie $e = xy$ prima muchie (arc) a acestui drum (există!). Să observăm că $y \notin X$. Considerăm $G' = G - e$. Avem $a(G') < a(G)$, deci (*) are loc pentru G' , X și Y .

Dacă $k(X, Y; G') = k$, atunci cele k XY -drumuri disjuncte din G' sînt XY -drumuri disjuncte și în G deci (*) are loc pentru G , X și Y .

Dacă $k(X, Y; G') < k$, atunci în G' există Z' XY -separatoare cu $|Z'| = k - 1$ (se aplică, eventual, proprietatea (c)).

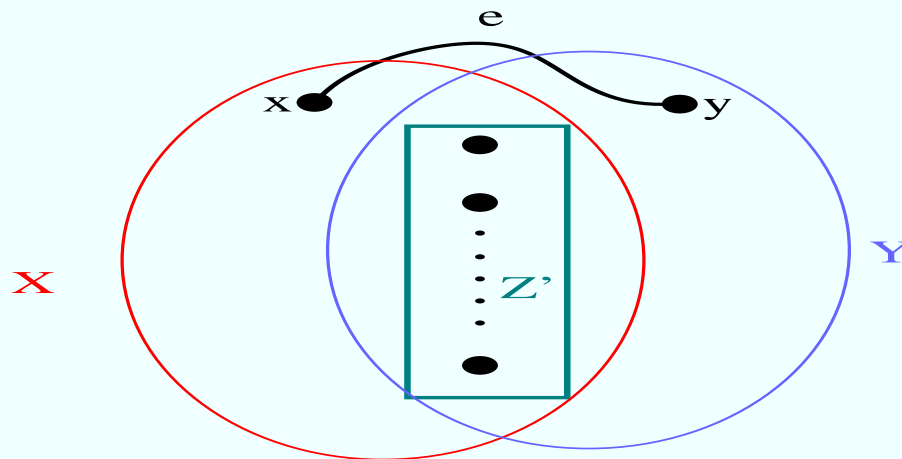
Deci Z' nu este XY -separatoare în G ($|Z'| < k$). Singurele XY -drumuri pe care Z nu le intersectează sînt cele care au drept primă muchie (arc) pe e .

Din definiția lui k , rezultă că $x \notin Z'$, $y \notin Z'$ și $|Z' \cup \{x\}| = |Z' \cup \{y\}| = k$.

Din alegerea lui x și y avem $Z' \cup \{x\} \neq Y$ și $Z' \cup \{y\} \neq X$.

Din ipoteza cazului 2, rezultă atunci că $Z' \cup \{x\} = X$ și $Z' \cup \{y\} = Y$.

Cele k drumuri din (*) sînt în acest caz $\{z\}_{z \in Z'}$ și (x, xy, y) .



Cu acestea, teorema este demonstrată.

Observații: 1⁰. Egalitatea min-max din enunțul teoremei este interesantă și conduce, așa cum vom vedea, la rezultate importante, în cazuri particulare.

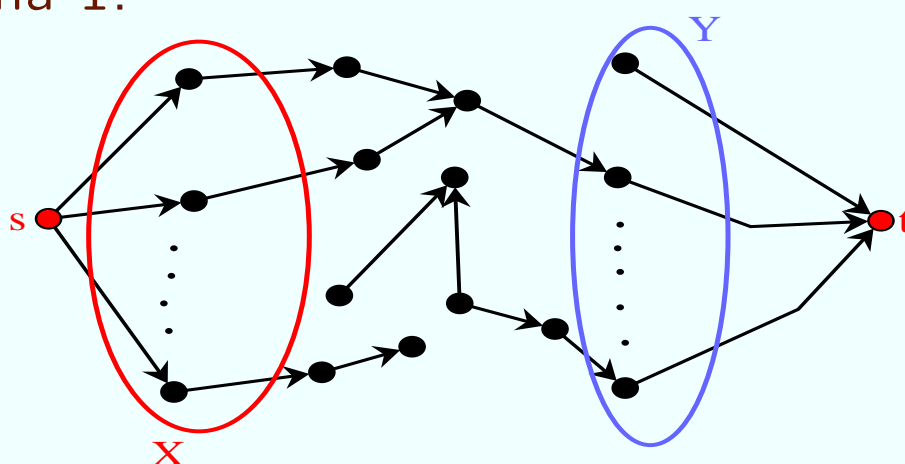
2⁰ Teorema se poate demonstra și algoritmic ca o consecință a teoremei fluxului maxim-secțiunii minime, așa cum vom arăta în capitolul relativ la probleme de flux.

Forma echivalentă în care a fost enunțată și demonstrată inițial de Menger (1927) teorema 1 este:

Teoremă. 1'. *Fie $G = (V, E)$ un (di)graf și $s, t \in V$, astfel încât $s \neq t$, $st \notin E$. Există k drumuri intern disjuncte de la s la t în graful G dacă și numai dacă îndepărtând mai puțin de k vârfuri diferite de s și t , în graful rămas există un drum de la s la t .*

Notăm că două drumuri sînt intern disjuncte dacă nu au vîrfuri comune cu excepția extremităților.

Se observă că dacă se consideră $X = N_G(s)$ și $Y = N_G(t)$ (respectiv, $N_G^+(s)$ și $N_G^-(t)$ în cazul digrafurilor) teorema 1' se obține imediat din teorema 1.



Reciproc, o construcție inversă celei de mai sus asupra tripletului G, X, Y din teorema 1, arată că teorema 1 se obține din teorema 1'.

Am definit un graf G p -conex ($p \in N^*$) dacă $G = K_p$ sau dacă $|G| > p$ și G nu poate fi deconectat prin îndepărtarea a mai puțin de p vîrfuri. Utilizînd teorema 2' obținem

Corolar. *Un graf G este p -conex dacă $G = K_p$ sau $\forall st \in E(\overline{G})$ există p drumuri intern disjuncte de la s la t în G .*

Determinarea numărului $k(G)$ de conexiune a grafului G (cea mai mare valoare a lui p pentru care G este p -conex) se reduce deci la determinarea lui

$$\min_{st \in E(\overline{G})} p(\{s\}, \{t\}; G)$$

problemă care vom dovedi că se poate rezolva în timp polinomial.

Un caz particular interesant al teoremei 1, se obține atunci cînd G este un graf bipartit iar X și Y sînt cele două clase ale bipartiției:

Teoremă. 2. (Konig, 1931) Dacă $G = (S, R; E)$ este un graf bipartit, atunci cardinalul maxim al unui cuplaj este egal cu cardinalul minim al unei mulțimi de vîrfuri incidente cu toate muchiile grafului.

Demonstrație: Evident, cardinalul maxim al unui cuplaj în G este $p(S, R; G)$, care este egal, conform teoremei 1, cu $k(S, R; G)$.

Teorema rezultă imediat dacă observăm că o mulțime de vîrfuri este SR -separatoare dacă și numai dacă este incidentă cu orice muchie a grafului.

O aplicație, fundamentală în numeroase raționamente combinatorii, a acestei teoreme este teorema lui **Hall** (1935).

Definiție: Fie I și S mulțimi finite nevide. Numim familie de submulțimi ale lui S (indexată după I) orice aplicație $\mathcal{A} : I \rightarrow 2^S$. Vom nota familia $\mathcal{A} = (A_i; i \in I)$ și vom folosi notația funcțională uzuală $\mathcal{A}(J) = \cup_{j \in J} A_j$ (pentru $J \subseteq I$).

Dacă $\mathcal{A} = (A_i; i \in I)$ este o familie de submulțimi ale lui S , o funcție $r_{\mathcal{A}} : I \rightarrow S$ cu proprietatea că $r_{\mathcal{A}}(i) \in A_i, \forall i \in I$ se numește *funcție de reprezentare pentru familia \mathcal{A}* .

În acest caz, $(r_{\mathcal{A}}(i); i \in I)$ formează un sistem de reprezentanți ai familiei \mathcal{A} .

Dacă funcția de reprezentare $r_{\mathcal{A}}$ este injectivă atunci $r_{\mathcal{A}}(I) \subseteq S$ se numește *sistem de reprezentanți distincți ai familiei \mathcal{A} , sau transversală*.

Problema centrală în teoria transversalelor este aceea de a caracteriza familiile \mathcal{A} care admit transversale (eventual cu anumite proprietăți). Prima teoremă de acest tip a fost stabilită de *Hall* în 1935:

Teoremă. 3. Familia $\mathcal{A} = (A_i; i \in I)$ de submulțimi ale lui S admite o transversală dacă și numai dacă

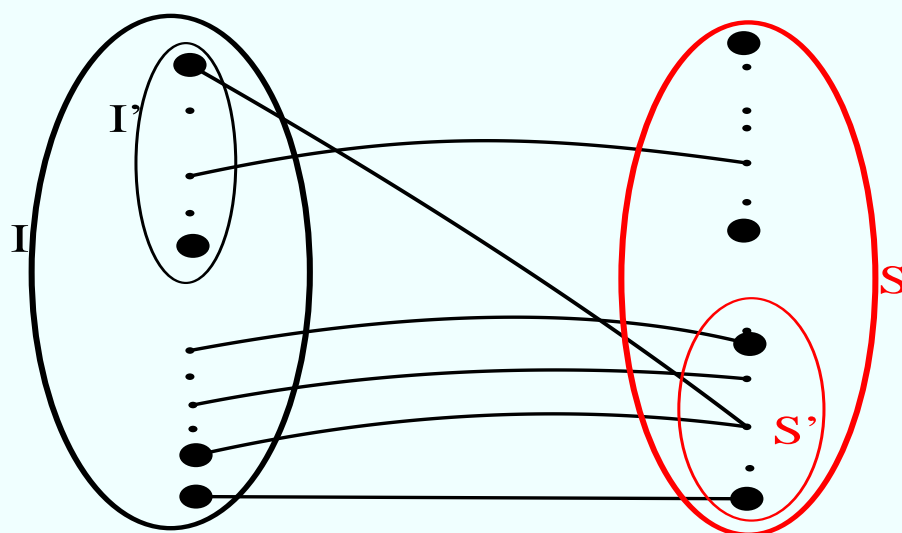
$$(H) \quad |\mathcal{A}(J)| \geq |J| \quad \forall J \subseteq I.$$

Demonstrație: Necesitatea este evidentă: dacă \mathcal{A} admite o funcție $r_{\mathcal{A}}$ de reprezentare injectivă atunci $\forall J \subseteq I \quad r_{\mathcal{A}}(J) \subseteq \mathcal{A}(J)$ și deci $|\mathcal{A}(J)| \geq |r_{\mathcal{A}}(J)| \geq |J|$ (întrucât $r_{\mathcal{A}}$ este injectivă).

Suficiența. Considerăm graful bipartit $G_{\mathcal{A}} = (I, S; E)$ unde am presupus $I \cap S = \emptyset$ (altfel, se consideră copii izomorfe disjuncte) iar $E = \{is \mid i \in I, s \in S \wedge s \in A_i\}$. Se observă că $N_{G_{\mathcal{A}}}(i) = A_i$ și că \mathcal{A} are o transversală dacă și numai dacă $G_{\mathcal{A}}$ are un cuplaj de cardinal $|I|$. În ipoteza că (H) are loc, arătăm că orice mulțime de vîrfuri incidentă cu toate muchiile lui $G_{\mathcal{A}}$ are măcar $|I|$ elemente,

ceea ce dovedește existența cuplajului de cardinal $|I|$ (utilizând teorema 2).

Fie $X = I' \cup S' \subset I \cup S$ o mulțime de vârfuri incidentă cu toate muchiile. Rezultă că $N_{G_{\mathcal{A}}}(I - I') \subseteq S'$, adică $\mathcal{A}(I - I') \subseteq S'$. Atunci, $|X| = |I'| + |S'| \geq |I'| + |\mathcal{A}(I - I')|$. Folosind condiția (H) obținem în continuare: $|X| \geq |I'| + |\mathcal{A}(I - I')| \geq |I'| + |I - I'| = |I|$.



O altă teoremă celebră care poate fi obținută ca o consecință imediată a teoremei 2 este teorema lui *Dilworth, 1950*.

Preferăm totuși, o demonstrație directă, pentru a evidenția asemănarea cu cea a teoremei 1. Fie (P, \leq) o mulțime finită parțial ordonată (\leq este o relație de ordine pe P).

Dacă $x, y \in P$, spunem că x și y sînt comparabile dacă $x \leq y$ sau $y \leq x$.

Un lanț în (P, \leq) este o submulțime L a lui P cu proprietatea că orice două elemente ale sale sînt comparabile.

Un antilanț în (P, \leq) este o submulțime A a lui P cu proprietatea că $\forall x, y \in A \ x \leq y \Rightarrow x = y$.

Teoremă. 4. (Dilworth, 1950) Dacă (P, \leq) este o mulțime parțial ordonată finită, atunci numărul minim de lanțuri a căror reuniune (disjunctă) este P este egal cu cardinalul maxim al unui antilanț.

Demonstrație. Fie $a(P, \leq)$ cardinalul maxim al unui antilanț al lui (P, \leq) .

Arătăm prin inducție după $|P|$, că există $a(P, \leq)$ lanțuri a căror reuniune este P (inegalitatea inversă este imediată).

Dacă $|P| = 1$, afirmația este trivială, deci presupunem, în pasul inductiv, că teorema are loc pentru orice mulțime parțial ordonată cu mai puțin de $|P| \geq 2$ elemente.

Fie L un lanț maximal (în raport cu incluziunea) al lui P .

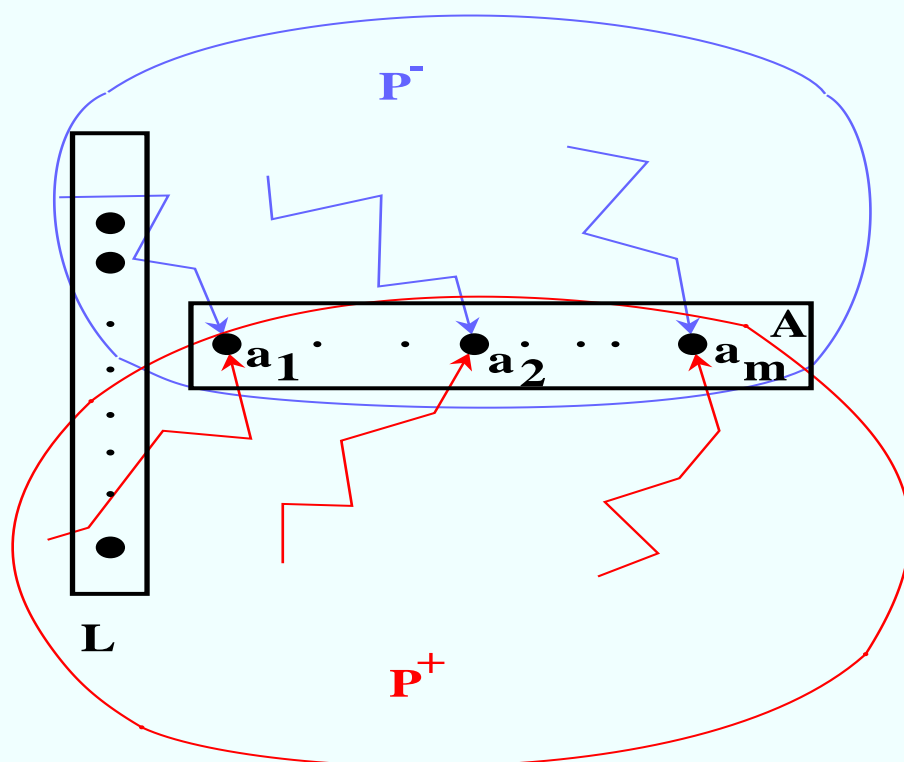
Dacă $a(P - L, \leq) = a(P, \leq) - 1$, atunci teorema are loc din ipoteza inductivă.

Deci, presupunem că în $P - L$ există un antilanț $A = \{a_1, a_2, \dots, a_m\}$ cu $m = a(P, \leq)$. Fie

$$P^- = \{x \mid x \in P, \exists a_i \in A : x \leq a_i\}$$

$$P^+ = \{x \mid x \in P, \exists a_j \in A : a_j \leq x\}.$$

Se observă că $P = P^- \cup P^+$, $P^- \cap P^+ = A$ și că $|P^-|, |P^+| < |P|$ [elementul maximal (minimal) al lui L nu aparține lui P^- (respectiv, P^+)].



Se poate, deci, aplica ipoteza inductivă pentru a scrie

$$P^- = \cup_{i=1,m} L_i^- \quad L_i^- \text{ lanț} \wedge a_i \in L_i^- \quad \forall i = 1, m$$

$$P^+ = \cup_{i=1,m} L_i^+ \quad L_i^+ \text{ lanț} \wedge a_i \in L_i^+ \quad \forall i = 1, m.$$

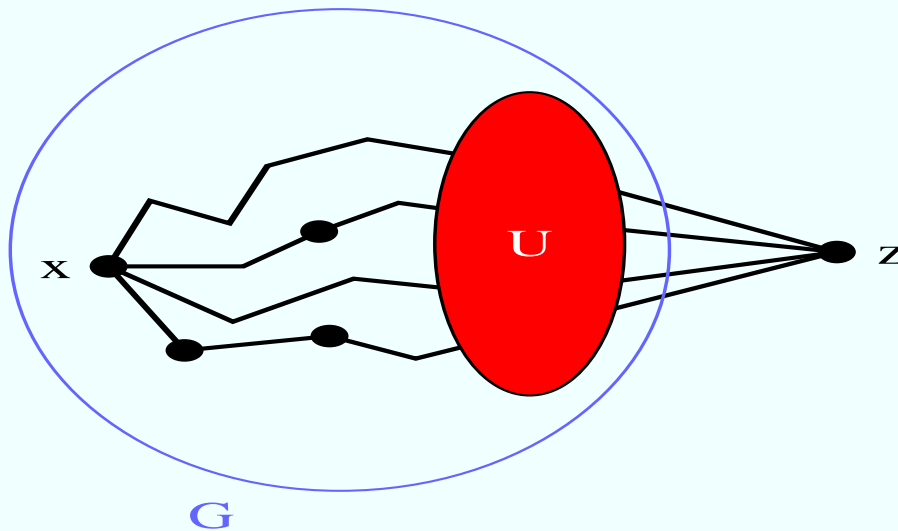
În plus, a_i este element maximal în L_i^- și element minimal în L_i^+ .

Rezultă că $(L_i^- \cup L_i^+)_{i=1,m}$ sînt cele $a(P, \leq)$ lanțuri a căror reuniune este P .

Structura grafurilor p -conexe.

Lemă. 1. Fie $G = (V, E)$ p -conex, $|V| \geq p+1$, $U \subseteq V$ $|U| = p$ și $x \in V - U$. Există în G p xU -drumuri cu singurul vîrf comun x .

Demonstrație: Considerăm graful $G' = (V \cup \{z\}, E')$, unde $E' = E \cup \{zy \mid y \in U\}$.

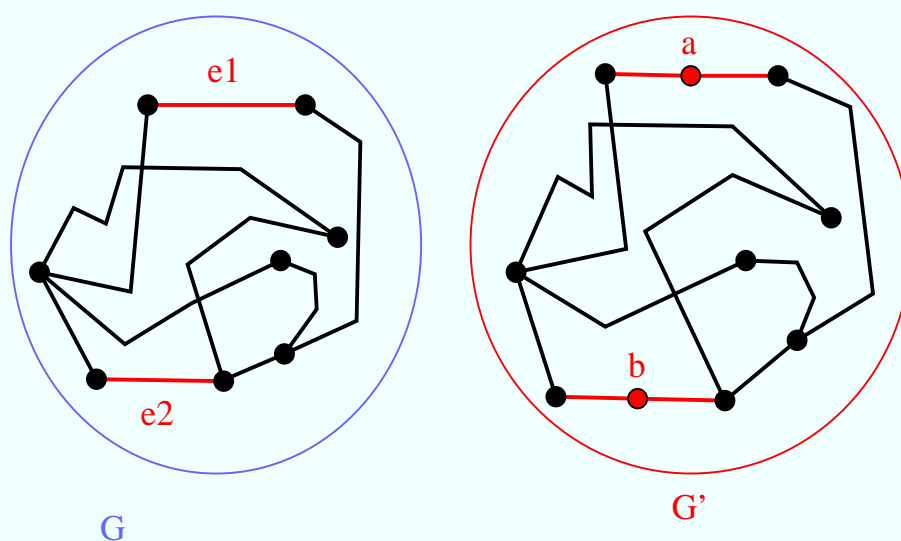


G' este p -conex. În adevăr, $\forall A$ cu $|A| \leq p-1$ $G' - A$ este conex (dacă $z \in A$, acest lucru este evident din p -conexiunea lui G ; dacă $A \subseteq V$, atunci $G' - A$ este conex întrucît $G - A$ este conex și $\exists y \in U$ cu $zy \in E(G' - A)$). Lema rezultă, aplicînd teorema 1' grafului G' și perechii x, z .

Lemă. 2. Dacă $G = (V, E)$ este un graf p -conex $p \geq 2$, atunci oricare ar fi două muchii e_1 și e_2 și $p - 2$ vîrfuri x_1, x_2, \dots, x_{p-2} există un circuit în G care le conține.

Demonstrție: Inducție după p .

Dacă $p = 2$, trebuie să dovedim că în orice graf 2-conex, prin orice două muchii trece un circuit. Considerăm G' obținut din G prin inserția cîte unui vîrf pe muchiile e_1 (a) și e_2 (b).



Noul graf este tot 2-conex, deoarece orice vîrf am scoate, nu se pierde conexiunea. Există deci în G' două ab -drumuri disjuncte, care în G oferă un circuit ce conține e_1 și e_2 .

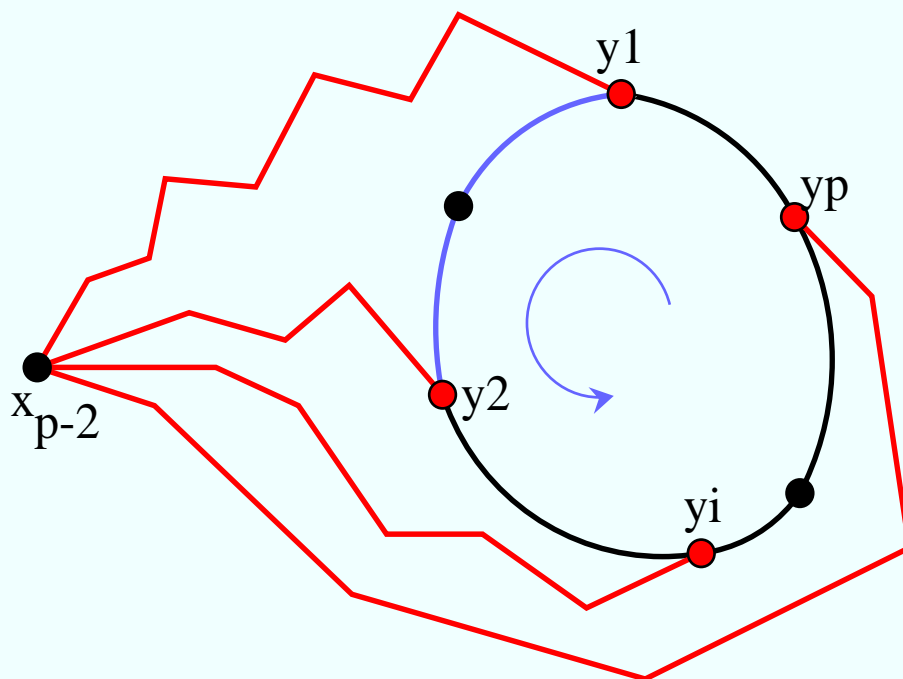
Fie $p > 2$ și presupunem afirmația adevărată pentru orice graf k -conex $k < p$. Fie G p -conex.

Putem presupune că extremitățile muchiilor e_1 și e_2 nu sînt printre x_1, x_2, \dots, x_{p-2} , deoarece altfel, afirmația ar rezulta prin inducție.

Graful $G - x_{p-2}$ este $(p - 1)$ -conex. Conform ipotezei inductive exista un circuit μ ce conține x_1, x_2, \dots, x_{p-3} și e_1, e_2 . Fie Y mulțimea vîrfurilor lui μ , $|Y| \geq p$.

Folosind lema 1, există în G p drumuri cu $y \in Y$, disjuncte. Putem presupune că pentru orice astfel de drum, y este primul vîrf din Y întîlnit, așa că aceste drumuri au cîte un singur vîrf comun cu Y . Dăm o orientare circuitului μ și numerotăm vîrfurile sale conform acestei orientări. Avem deci drumurile

$$D_{x_{p-2}y_1}, D_{x_{p-2}y_2}, \dots, D_{x_{p-2}y_p}.$$



Vîrfurile y_1, y_2, \dots, y_p descompun circuitul în drumurile $D_{y_1 y_2}, D_{y_2 y_3}, \dots, D_{y_{p-1} y_p}, D_{y_p y_1}$.

Există un drum dintre acestea, în care nu e conținut nici unul din elementele $x_1, \dots, x_{p-3}, e_1, e_2$.

Fie acest drum $D_{y_1 y_2}$; atunci $D_{x_{p-2} y_2}, D_{y_2 y_3}, \dots, D_{y_p y_1}, D_{y_1 x_{p-2}}$ este un circuit ce conține $x_1, x_2, \dots, x_{p-2}, e_1$ și e_2 , și lema este demonstrată.

Teoremă. 5. (Dirac 1953) *Dacă $G = (V, E)$ este un graf p -conex $p \geq 2$, atunci prin orice p vîrfuri ale sale trece un circuit.*

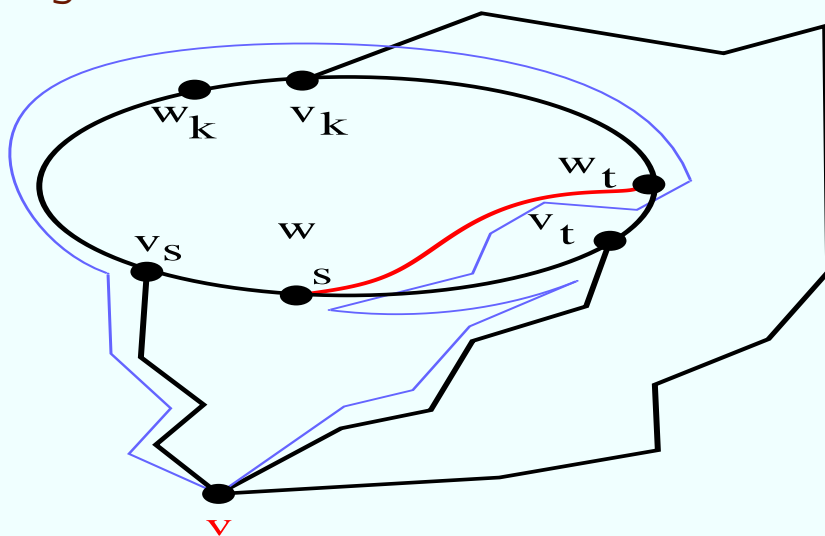
Demonstrație. Fie $x_1, x_2, \dots, x_{p-2}, x_{p-1}, x_p$ p vîrfuri oarecare ale lui G . Deoarece graful G este conex, există $e_1 = xx_{p-1}$ și $e_2 = yx_p$ și aplicăm lema 2.

Aplicăm această teoremă, precum și ideea utilizată în demonstrația lemei 2, pentru a demonstra o condiție suficientă de hamiltonietate interesantă, datorată lui *Erdős și Chvatal (1972)*.

Teoremă. 6. *Fie G p -conex. Dacă $\alpha(G) \leq p$ atunci G este hamiltonian.*

Demonstrație: Presupunem că G nu e hamiltonian. Vom obține o contradicție. Cum G este p -conex, există un circuit de lungime cel puțin p (conform teoremei lui Dirac de mai sus).

Fie C un circuit de lungime maximă în G . Dacă G nu e hamiltonian, există $v \notin C$. Întrucît $|C| \geq p$, conform lemei 2, există p vC -drumuri disjuncte (cu excepția lui v) fie ele $D_{vv_1}, D_{vv_2}, \dots, D_{vv_p}$ (numerotarea vîrfurilor este în ordinea întîlnirii lor într-o parcurgere fixată a circuitului). Notăm, pentru fiecare v_i , cu w_i vîrfurile succesori al lui v_i în parcurgerea lui C .



Atunci, $vw_i \notin E$ (altfel am avea circuitul $vw_i, w_i, C - w_i v_i, D_{v_i v}$ de lungime mai mare decît C). Cum $\alpha(G) \leq p$, mulțimea $\{v, w_1, w_2, \dots, w_p\}$ nu este stabilă. Deci, există $w_s w_t \in E$.

Dar atunci:

D_{vv_s} , drumul (invers) pe C de la v_s la w_t , muchia $w_t w_s$, drumul (invers) pe C de la w_s la v_t , și $D_{v_t v}$ este un circuit de lungime mai mare decît C , contrazicînd ipoteza că C este de lungime maximă.

III. ARBORI

1. Proprietăți elementare ale arborilor

Definiție: Un **arbore** este un graf conex și fără circuite.

Teoremă. 1. *Fie $G = (V, E)$ un graf.*

Următoarele afirmații sînt echivalente:

(i) *G este arbore.*

(ii) *G este conex și este minimal cu această proprietate.*

(iii) *G este fără circuite și este maximal cu această proprietate.*

*Observație:*Maximalitatea și minimalitatea din condițiile (ii) și (iii) se referă la mulțimea muchiilor grafului G și se consideră în raport cu relația de ordine dată de incluziune. Mai precis, cele două afirmații se pot formula echivalent astfel:

(ii') *G este conex și $\forall e \in E(G)$, $G - e$ este neconex.*

(iii') *G este fără circuite și $\forall e \in E(\overline{G})$, $G + e$ are un circuit.*

Definiție: Fie $G = (V, E)$ un (multi)graf. Se numește **arbore parțial** al lui G , un graf parțial $T = (V, E')$ ($E' \subseteq E$) care este arbore.

Vom nota cu \mathcal{T}_G **mulțimea arborilor parțiali** ai lui G .

Obs. $\mathcal{T}_G \neq \emptyset$ dacă și numai dacă G este conex.

În adevăr, dacă $\mathcal{T}_G \neq \emptyset$, atunci există un arbore parțial $T = (V, E')$ al lui G . T este conex, deci între orice două vîrfuri ale lui G există un drum cu muchii din $E' \subseteq E$. Prin urmare G este conex.

Reciproc, dacă G este conex, atunci considerăm următorul algoritm:

1. $T \leftarrow G$
2. **while** $(\exists e \in E(T)$ astfel încît $T \setminus \{e\}$ este conex) **do**
 $T \leftarrow T \setminus \{e\}$

Graful T obținut este graf parțial al lui G , este conex (din ipoteză, după atribuirea din 1, așa este și din condiția lui **while**, T este conex după fiecare iterație) și în plus la oprirea algoritmului, T satisface condiția *ii*) din teorema 1, deci este arbore.

O altă demonstrație a reciprocei anterioare se bazează pe observația că $G = (V, E)$ este conex dacă și numai dacă oricare ar fi o partiție (V_1, V_2) a lui V există $e = v_1v_2 \in E$ cu $v_i \in V_i$ $i = \overline{1, 2}$.

Dacă $|V| = n > 0$ atunci următorul algoritm construiește un arbore parțial al lui G :

1. $T_1 \leftarrow (\{v\}, \emptyset)$ ($v \in V$, oarecare); $k \leftarrow 1$;
2. **while** $k < n$ **do**
 - { Fie $v_1v_2 \in E$ cu $v_1 \in V(T_k), v_2 \in V \setminus V(T_k)$;
 - // \exists o astfel de muchie din conexiunea lui G
 - $V(T_{k+1}) \leftarrow V(T_k) \cup \{v_2\}$;
 - $E(T_{k+1}) \leftarrow E(T_k) \cup \{v_1v_2\}$;
 - $k := k + 1$
 - }

Se observă că T_k este arbore $\forall k = \overline{1, n}$

(inductiv, dacă T_k este arbore, atunci din construcție T_{k+1} este conex și nu are circuite)

și, în plus, se verifică imediat că:

$$|V(T_k)| = k \text{ iar } |E(T_k)| = k - 1 \quad \forall k = 1, 2, \dots, n.$$

Această demonstrație aplicată unui arbore G cu n vîrfuri dovedește ca G are $n - 1$ muchii.

Proprietatea obținută poate fi folosită pentru completarea teoremei 1 cu alte caracterizări ale arborilor:

Teoremă. 1'. *Următoarele afirmații sînt echivalente pentru un graf $G = (V, E)$ cu n vîrfuri:*

(i) G este arbore.

(ii) G este conex și are $n - 1$ muchii.

(iii) G este fără circuite și are $n - 1$ muchii.

(iv) $G = K_n$ pentru $n = 1, 2$ și $G \neq K_n$ pentru $n \geq 3$ și adăugarea unei muchii la G produce exact un circuit.

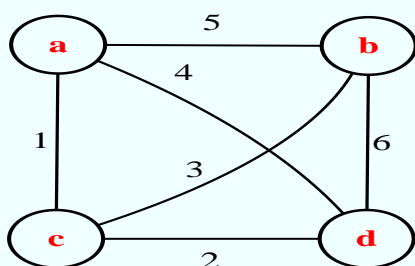
2. Numărarea și enumerarea arborilor parțiali

Familia \mathcal{T}_G a arborilor parțiali ai unui (multi)graf are proprietăți interesante. Vom prezenta o metodă (tip backtrack) de **generare a elementelor lui \mathcal{T}_G** , problemă de interes practic în multe aplicații (de exemplu, în chimie).

Fie $G = (V, E)$, $V = \{1, 2, \dots, n\}$, $|E| = m$.

Reprezentăm E printr-un tablou $E[1..m, 1..2]$ cu componente din V cu semnificația că dacă $v = E[i, 1]$ și $w = E[i, 2]$, atunci vw este muchia i a grafului G ($i = \overline{1, m}$).

Vom presupune în plus, că primele $d_G(v_0)$ muchii din tabloul E satisfac $E[i, 1] = v_0$ unde $v_0 \in V$ este un vîrf oarecare. Exemplu:



| | | |
|---|----------|----------|
| 1 | c | a |
| 2 | c | d |
| 3 | c | b |
| 4 | a | d |
| 5 | a | b |
| 6 | b | d |

E

Un arbore parțial $T \in \mathcal{T}_G$ va fi identificat cu mulțimea indicilor ce reprezintă muchiile sale în tabloul E (submulțime a lui $\{1, \dots, m\}$ de cardinal $n - 1$).

Pe tot parcursul generării dispunem de un vector global $T[1..n-1]$ cu componente din mulțimea $1..m$ și de un indicator i avînd semnificația: *în arborele curent care se construiește, primele $i - 1$ muchii sînt*

$T[1] < T[2] < \dots < T[i - 1] \quad (i \in \{1, \dots, n\})$.

generare-arbori-parțiali(*int* i);

// se generează toți arborii parțiali ai lui G
avînd drept prime $i - 1$ muchii, elementele
 $T(1), \dots, T(i - 1)$
ale tabloului E (ordonate crescător).

variabile locale:

$j \in \{1, \dots, m\}$; S , listă de vîrfuri; $x \in V$;

if $i = n$ **then**

// $\{T(1), \dots, T(n - 1)\}$ formează un
arbore parțial ;

prelucrează T (listează, memorează etc.)

```

else
  if  $i = 1$  then
    for  $j := 1$  to  $d_G(v_0)$  do
      {  $T[i] \leftarrow j$ ;
        A:
        generare-arbori-parțiali( $i + 1$ );
        B:
      }
  else
    for  $j := T[i - 1] + 1$  to  $m - (n - 1) + i$  do
      if  $\langle \{T[1], \dots, T[i - 1]\} \cup \{j\} \rangle_G$  nu are circuite
      then
        {  $T[i] \leftarrow j$ ;
          A:
          generare-arbori-parțiali( $i + 1$ );
          B:
        }

```

Apelul *generare-arbori-parțiali*(1) rezolvă problema enumerării elementelor lui \mathcal{T}_G .

Pentru exemplul considerat mai sus ($G = K_4$ și numerotarea precizată pentru muchii) arborii generați sunt următorii 16 :

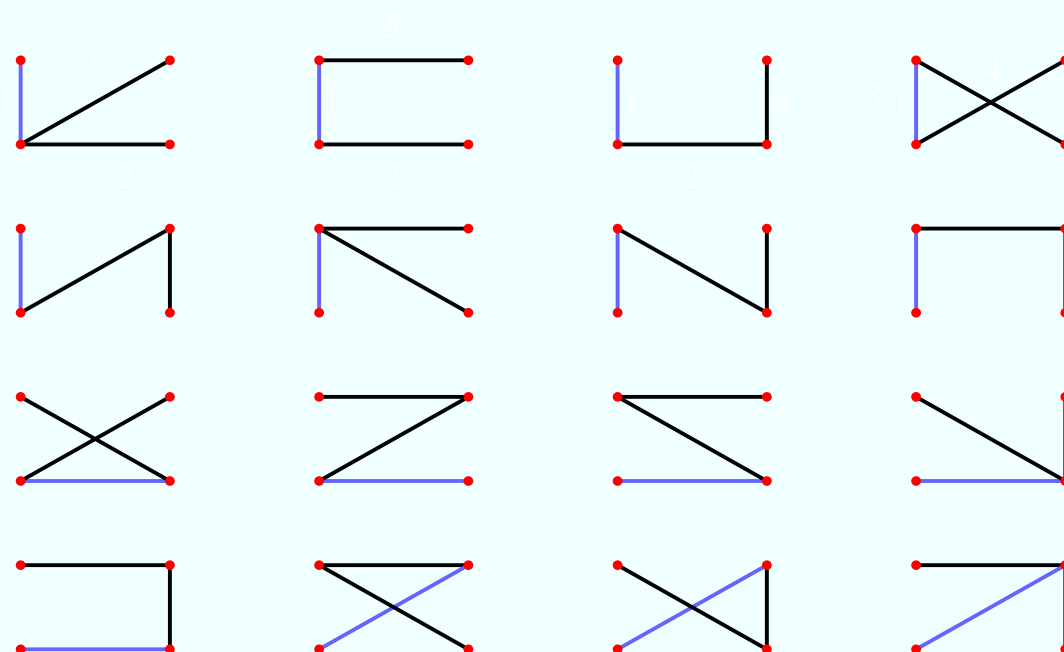


Figura poate fi interpretată și ca lista tuturor arborilor **etichetați** distincți care se pot forma cu cele 4 noduri (nu s-au mai trecut etichetele nodurilor și muchiilor, pentru decongestionarea figurii; totuși s-a marcat muchia $T(1)$ pentru a se evidenția modul de funcționare a algoritmului).

Pentru implementarea eficientă a testului dacă graful parțial

$\langle \{T[1], \dots, T[i-1]\} \cup \{j\} \rangle_G$ nu are circuite să observăm că din construcție,

$$\langle \{T[1], \dots, T[i-1]\} \rangle_G$$

nu are circuite , deci componentele sale conexe sînt arbori.

Vom considera un vector global $rad[1..n]$ cu componente din V cu semnificația

$rad[v] =$ rădăcina arborelui la care aparține vîrfurile v (unul din vîrfurile acestui arbore).

Înaintea apelului *generare-arbori-parțiali(1)* se inițializează $rad[v] \leftarrow v (\forall v \in V)$, ceea ce core-spunde faptului că $\{T[1], \dots, T[i-1]\} = \emptyset$.

Pe parcursul apelurilor (recursive) se încearcă plasarea muchiei j în mulțimea curentă $T[1], \dots, T[i-1]$.

Fie $v = E[j, 1]$ și $w = E[j, 2]$.

Atunci $\langle \{T[1], \dots, T[i-1]\} \cup \{j\} \rangle_G$ nu are circuite **dacă și numai dacă**

muchia vw nu are extremitățile în aceeași componentă a lui

$$\langle \{T[1], \dots, T[i-1]\} \rangle_G,$$

adică **dacă și numai dacă** $rad[v] \neq rad[w]$.

Vectorul rad trebuie întreținut pentru a avea semnificația dorită.

Acest lucru se obține înlocuind în algoritmul descris, instrucțiunile (vide) etichetate A și B. Astfel, A: se va înlocui cu secvența :

```
 $S \leftarrow \emptyset; x \leftarrow rad[v];$   
for  $u \in V$  do  
  if  $rad[u] = x$  then  
  {  $S \leftarrow S \cup \{u\}$  ;  
     $rad[u] \leftarrow rad[w]$   
  }
```


(în cuvinte, arborele cu rădăcina x se "unește" cu arborele cu rădăcina $rad[w]$; se salvează în S vîrfurile arborelui cu rădăcina x).

După apelul lui *generare-arbori-parțiali*($i + 1$) trebuie refăcut vectorul rad la valoarea dinainte de apel, deci se va înlocui B: cu

for $u \in S$ **do** $rad[u] := x$;

Numărul elementelor lui \mathcal{T}_G , problemă interesantă chiar și numai pentru analiza algoritmului precedent, se poate determina eficient. Prezentăm în continuare una din soluțiile posibile.

Fie $G = (V, E)$ un multigraf cu $V = \{1, 2, \dots, n\}$. Considerăm $A = (a_{ij})_{n \times n}$ matricea de adiacență a lui G (a_{ij} = multiplicitatea muchiei ij dacă $ij \in E$, altfel 0). Fie

$$D = \text{diag}(d_G(1), d_G(2), \dots, d_G(n)).$$

Matricea $L[G] = D - A$ se numește **matricea de admitanță a multigrafului G** sau **matricea Laplace a lui G** .

Să observăm că în $L[G]$ suma elementelor de pe fiecare linie și fiecare coloană este 0.

Teoremă. 2. (Kirchoff-Trent) (*Matrix Tree Theorem*) Dacă G este un multigraf cu mulțimea de vîrfuri $\{1, \dots, n\}$ și $L[G]$ matricea Laplace, atunci

$$|\mathcal{T}_G| = \det(L[G]_{ii}) \quad \forall i \in \{1, \dots, n\}.$$

$L[G]_{ij}$ notează minorul lui $L[G]$ obținut prin îndepărtarea liniei i și coloanei j .

Demonstrația (pe care o omitem) se bazează pe regula clasică de dezvoltare a unui determinat după o linie, după descompunerea lui \mathcal{T}_G în arborii care conțin o muchie fixată și cei care nu conțin aceeași muchie.

Corolar. $|\mathcal{T}_{K_n}| = n^{n-2}$ (Cayley).

În adevăr,

$$L[K_n] = \begin{pmatrix} n-1 & -1 & \dots & -1 \\ -1 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & -1 \\ -1 & \dots & -1 & n-1 \end{pmatrix}$$

și (după un simplu calcul):

$$\det(L[K_n]_{11}) = \begin{vmatrix} n-1 & -1 & \dots & -1 \\ -1 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & -1 \\ -1 & \dots & -1 & n-1 \end{vmatrix} = n^{n-2}.$$

(Dacă $n = 4$, caz ilustrat în figura precedentă, se obține $|\mathcal{T}_{K_4}| = 4^2 = 16$, adică au fost generați toți arborii !)

Observație. Teorema oferă un algoritm polynomial de determinare a lui $|\mathcal{T}_G|$.

3. Arbori parțiali de cost minim.

Considerăm următoarea problemă:

(P1) Date $G = (V, E)$ graf și $c : E \rightarrow \mathbf{R}$ ($c(e)$ – costul muchiei e), să se determine $T^* \in \mathcal{T}_G$ astfel încât

$$c(T^*) = \min\{c(T) \mid T \in \mathcal{T}_G\},$$

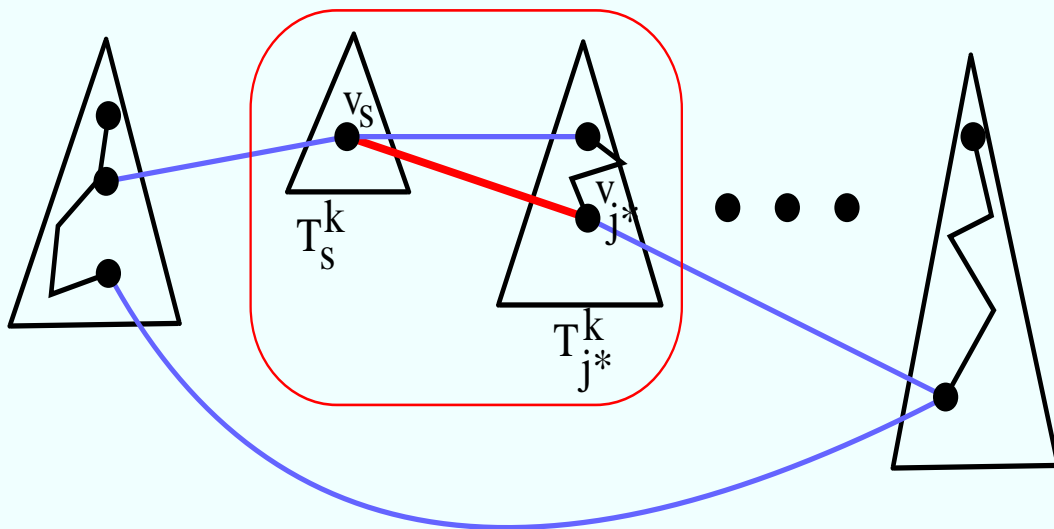
unde $c(T) = \sum_{e \in E(T)} c(e)$.

Algoritmii cunoscuți pentru rezolvarea problemei (P1) au la bază următoarea idee:

Se consideră inițial, familia $\mathcal{T}^0 = (T_1^0, T_2^0, \dots, T_n^0)$ de arbori disjuncți $T_i^0 = (\{i\}, \emptyset)$ $i = \overline{1, n}$ (am presupus, ca de obicei că $V = \{1, 2, \dots, n\}$) .

În pasul general k ($k = \overline{0, n-2}$) al algoritmului se dispune de familia $\mathcal{T}^k = (T_1^k, T_2^k, \dots, T_{n-k}^k)$ de $n-k$ arbori disjuncți astfel încât $V(T_i^k)_{i=\overline{1, n-k}}$ constituie o partiție a lui V și se construiește \mathcal{T}^{k+1} astfel:

- se alege T_s^k unul din arborii familiei \mathcal{T}^k .
- dintre toate muchiile lui G cu o extremitate în T_s^k și cealaltă în $V - V(T_s^k)$ se alege una de cost minim, $e^* = v_s v_{j^*}$ unde $v_{j^*} \in V(T_{j^*}^k)$ $j^* \neq s$.
- $\mathcal{T}^{k+1} = (\mathcal{T}^k \setminus \{T_s^k, T_{j^*}^k\}) \cup T$ unde T este arborele obținut din T_s^k și $T_{j^*}^k$ la care adăugăm muchia e^* .



Se verifică imediat că noua familie este formată din arbori disjuncți care partiționează mulțimea de vîrfuri ale grafului G . Dacă alegerea muchiei e^* nu este posibilă, atunci rezultă că G nu este conex și deci problema (P1) nu are soluție.

Evident, familia \mathcal{T}^{n-1} construită în pasul $n - 2$ este formată dintr-un singur arbore T_1^{n-1} .

Teoremă. 3. *Dacă $G = (V, E)$ este un graf conex cu $V = \{1, 2, \dots, n\}$ atunci T_1^{n-1} construit de algoritmul descris mai sus este arbore parțial de cost minim.*

Demonstrație. Vom arăta că $\forall k \in \{0, 1, \dots, n - 1\} \exists T^*$ arbore parțial de cost minim al lui G , astfel încât $E(\mathcal{T}^k) = \cup_{i=1}^{n-k} E(T_i^k) \subseteq E(T^*)$.

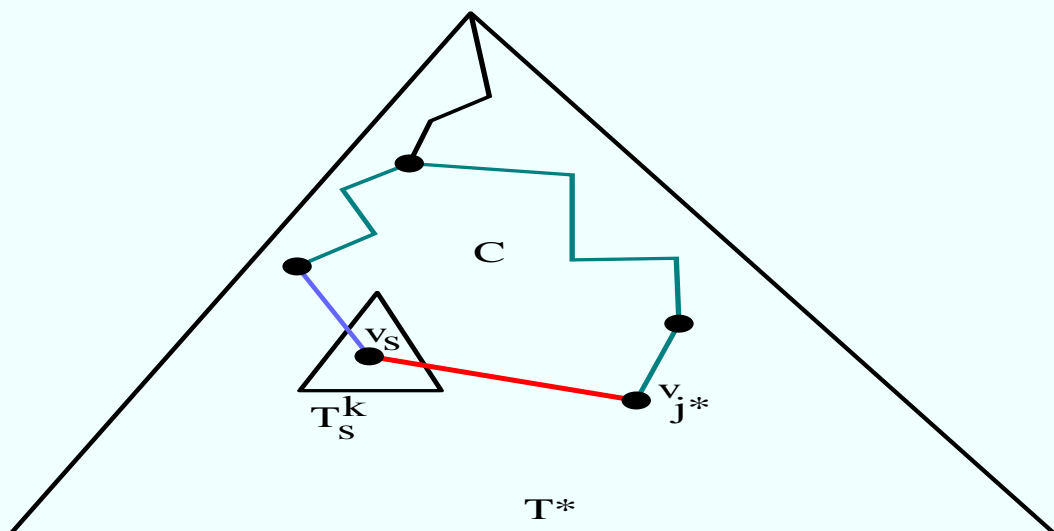
În particular, pentru $k = n - 1$, $E(\mathcal{T}^{n-1}) = E(T_1^{n-1}) \subseteq E(T^*)$ va implica $T_1^{n-1} = T^*$ (cei doi arbori avînd același număr de muchii, rezultă că incluziunea are loc cu egalitate) și teorema e demonstrată.

Pentru $k = 0$, afirmația este trivial adevărată: $E(\mathcal{T}^0) = \emptyset$ și din conexiunea grafului G , \mathcal{T}_G este nevidă deci există T^* soluție a problemei $P1$.

Dacă afirmația este adevărată pentru $0 \leq k \leq n - 2$, atunci avem $E(T^k) \subseteq E(T^*)$ (T^* arbore parțial de cost minim) și $E(T^{k+1}) = E(T^k) \cup \{e^*\}$.

Dacă $e^* \in E(T^*)$, atunci, evident, $E(T^{k+1}) \subseteq E(T^*)$ și deci afirmația are loc pentru $k + 1$.

Presupunem, deci, că $e^* \notin E(T^*)$. Atunci $T^* + \{e^*\}$ conține exact un circuit C ce trece prin muchia $e^* = v_s v_{j^*}$. Cum $v_{j^*} \notin V(T_s^k)$ rezultă că C va conține o muchie $e_1 \neq e^*$ cu o extremitate în $V(T_s^k)$ și cealaltă în $V \setminus V(T_s^k)$. Din alegerea muchiei e^* , avem $c(e^*) \leq c(e_1)$ și $e_1 \in E(T^*) \setminus E(T^k)$.



Fie $T^1 = (T^* + \{e^*\}) - \{e_1\}$.

$T^1 \in \mathcal{T}_G$ (este conex și are $n-1$ muchii).

Din construcție, avem că $E(\mathcal{T}^{k+1}) \subseteq E(T^1)$.

În plus, $c(T^1) = c(T^*) + c(e^*) - c(e_1) \leq c(T^*)$
deci $c(T^1) = c(T^*)$, adică T^1 este de cost minim
și teorema e demonstrată.

Observații:

1⁰ Demonstrația anterioară rămîne valabilă pentru funcții de cost $c : \mathcal{T}_G \rightarrow \mathbf{R}$ astfel încît:

$\forall T \in \mathcal{T}_G, \forall e \in E(T), \forall e' \notin E(T)$

$$c(e') \leq c(e) \Rightarrow c((T + e') - e) \leq c(T).$$

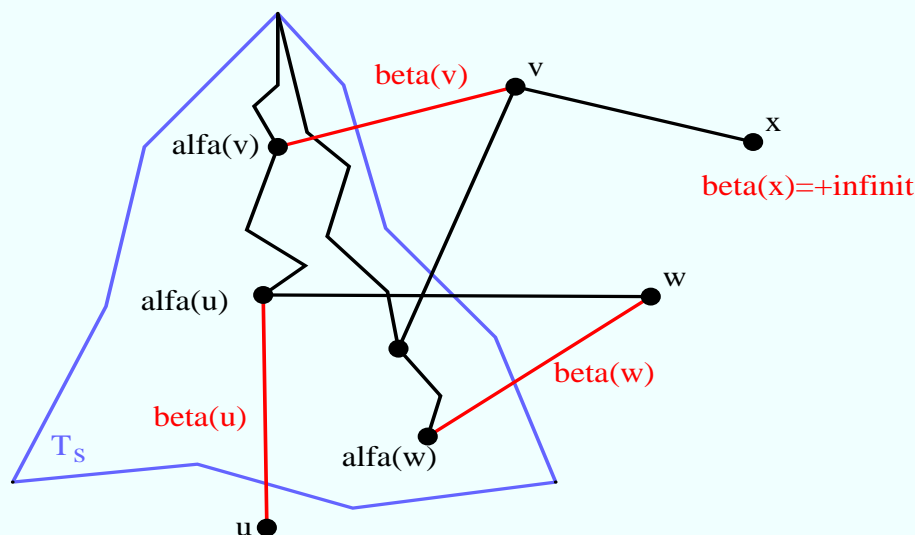
2⁰ În algoritmul descris nu s-a precizat modul de alegere al arborelui T^k . Vom considera, în continuare două strategii de alegere a acestui arbore.

Algoritmul lui Prim(1957) (implementarea este datorată lui **Dijkstra, 1961**).

Arborele T_s^k va fi întotdeauna arborele cu cele mai multe vîrfuri dintre arborii familiei curente.

Rezultă deci, că la fiecare pas $k > 0$, vom avea un arbore cu $k + 1$ vîrfuri, ceilalți $n - k - 1$ avînd cîte un singur vîrf. Notăm $T_s = (V_s, E_s)$ arborele curent. Considerăm vectorii $\alpha[1..n]$ cu componente din V și $\beta[1..n]$ cu componente reale, cu următoarea semnificație:

(S)
 $\forall j \in V - V_s, \quad \beta[j] = c(\alpha[j]j) = \min\{c(ij) \mid i \in V_s, ij \in E\}.$



Descrierea algoritmului:

```

1.  $V_s \leftarrow \{s\};$  ( $s \in V$ , oarecare )
    $E_s \leftarrow \emptyset;$ 
   for  $v \in V \setminus \{s\}$  do {  $\alpha[v] := s; \beta[v] := c(sv)$  };
           // dacă  $ij \notin E$  atunci  $c(ij) = \infty$ ).
2. while  $V_s \neq V$  do
   { determină  $j^* \in V \setminus V_s$  a.î.
      $\beta[j^*] = \min\{\beta[j] \mid j \in V - V_s\}$  ;
      $V_s \leftarrow V_s \cup \{j^*\};$ 
      $E_s := E_s \cup \{\alpha[j^*]j^*\};$ 
     for  $j \in V - V_s$  do
       if  $\beta[j] > c[j^*j]$  then
       {  $\beta[j] \leftarrow c[j^*j];$ 
          $\alpha[j] \leftarrow j^*$ 
       }
   }

```

Se observă că (S) este satisfăcută de inițializările pasului 1, iar în pasul 2, se respectă, pe de o parte, strategia generală de alegere a muchiei de cost minim cu exact o extremitate în V_s (alegerea lui j^*) și pe de altă parte se menține valabilitatea condiției (S) pentru iterația următoare (testul asupra valorii curente a lui $\beta[j]$).

Complexitatea algoritmului este $O(n-1) + O(n-2) + \dots + O(1) = O(n^2)$ dată de operațiile din pasul 2 necesare determinării minimului și actualizării tabloului β . Se poate introduce testul de conexiune a grafului, după determinarea lui $\beta[j^*]$. Algoritmul este recomandat în cazul grafurilor cu multe muchii, $m = O(n^2)$.

Algoritmul lui Kruskal (1956)

În metoda generală prezentată, se va alege la fiecare pas drept arbore T_s^k **unul din cei doi arbori cu proprietatea că sînt "uniți" printr-o muchie de cost minim printre toate muchiile cu extremitățile pe arbori diferiți.**

Această alegere, complicată la prima vedere, se realizează simplu prin sortarea muchiilor grafului nedescrescător în raport cu costurile și apoi prin examinarea în mod "greedy" a listei obținute.

Dacă notăm cu $T = E(\mathcal{T}^k)$, atunci algoritmul poate fi descris, astfel:

1. Sortează $E = (e_1, e_2, \dots, e_m)$ astfel încît:

$$c(e_1) \leq c(e_2) \leq \dots \leq c(e_m).$$
 - 1.2 $T \leftarrow \emptyset; i \leftarrow 1;$
2. **while** $i \leq m$ **do**
{ **if** $\langle T \cup \{e_i\} \rangle_G$ nu are circuite **then**

$$T \leftarrow T \cup \{e_i\} ;$$

$$i++$$
}

Evident, pasul 1 necesită $O(m \log n)$ operații. Pentru realizarea eficientă a testului din pasul 2 va fi necesar să reprezentăm la fiecare pas k (din metoda generală) $V(T_1^k), V(T_2^k), \dots, V(T_n^k)$ și să testăm dacă muchia e_i curentă are ambele extremități în aceeași mulțime. Se vor folosi pentru reprezentarea acestor mulțimi, arbori (care nu sînt în general subarbori ai lui G).

Fiecare astfel de arbore va avea un vîrf, numit *rădăcină*, care va desemna mulțimea de vîrfuri ale lui G pe care o memorează.

Vom folosi o funcție $find(v)$ care determină în ce mulțime este vârful v , adică întoarce rădăcina arborelui care memorează mulțimea de vârfuri la care aparține v .

Pentru realizarea reuniunilor (disjuncte) de mulțimi de vârfuri care apar în transformarea familiilor \mathcal{T}^k (din metoda generală) vom folosi o procedură $union(v, w)$ cu semnificația:

"mulțimile de vârfuri (diferite și disjuncte) la care aparțin v și w se reunesc în una singură".

Cu aceste proceduri, pasul 2 al algoritmului se scrie:

```
2. while  $i \leq m$  do
    {   fie  $e_i = vw$ ;
        if  $find(v) \neq find(w)$  then
            {    $union(v, w)$ ;
                 $T \leftarrow T \cup \{e_i\}$ 
            };
         $i++$ 
    }
```

Complexitatea algoritmului va depinde de modul de implementare a funcției *find* și procedurii *union*.

Soluția I^a . Considerăm tabloul $rad[1..n]$ cu componente din V cu semnificația $rad[v] = \text{rădăcina arborelui ce memorează mulțimea la care aparține vârful } v$.

Adăugăm pasului 1, inițializarea

```
1.3 for  $v \in V$  do  $rad[v] \leftarrow v$ ;
```

care corespunde familiei \mathcal{T}^0 . Funcția *find* are în acest caz complexitatea $O(1)$ și este:

```
function find( $v : V$ );  
    return  $rad[v]$ 
```

Procedura *union* necesită $O(n)$ operații:

```
procedure union( $v, w : V$ );  
    variabilă locală  $i : V$ ;  
    for  $i \in V$  do  
        if  $rad[i] = rad[v]$  then  $rad[i] := rad[w]$ 
```

Pasul 2 al algoritmului necesită $O(m)$ apeluri ale funcției *find* (exact m așa cum l-am descris, sau $O(m)$ dacă se introduce și un test asupra cardinalului mulțimii T curente).

În secvența acestor $O(m)$ apeluri ale funcției *find* se vor intercala exact $n - 1$ apeluri ale procedurii *union*.

Rezultă că în total pasul 2 necesită $O(m + (n - 1)O(m)) = O(n^2)$ operații.

Deci, complexitatea algoritmului este $O(\max(m \log n, n^2))$.

Dacă graful este plin, $m = O(n^2)$, se observă că acest algoritm este mai puțin eficient decât cel al lui Prim.

Soluția a II^a. Considerăm $pred[1..n]$ un tablou întreg cu interpretarea :

" $pred[v]$ = vârful dinaintea lui v de pe drumul unic la v , de la rădăcina arborelui care memorează mulțimea la care aparține v ;
 $pred[v] = 0 \Leftrightarrow v$ este rădăcina arborelui".

Adăugăm în pasul 1, inițializarea

1.3 **for** $v \in V$ **do** $pred[v] \leftarrow 0$;

Modificăm și pasul 2 al algoritmului astfel:

```
2. while  $i \leq m$  do
  {   fie  $e_i = vw$ ;
       $x \leftarrow find(v)$ ;  $y \leftarrow find(w)$ ;
      if  $x \neq y$  then
        {    $union(x, y)$ ;
             $T \leftarrow T \cup \{e_i\}$ 
        };
       $i++$ 
  }
```


Deci, procedura *union* va fi apelată numai pentru argumente reprezentând rădăcini de arbori diferiți:

```
procedure union( $v, w : V$ );  
     $pred[v] \leftarrow w$ 
```

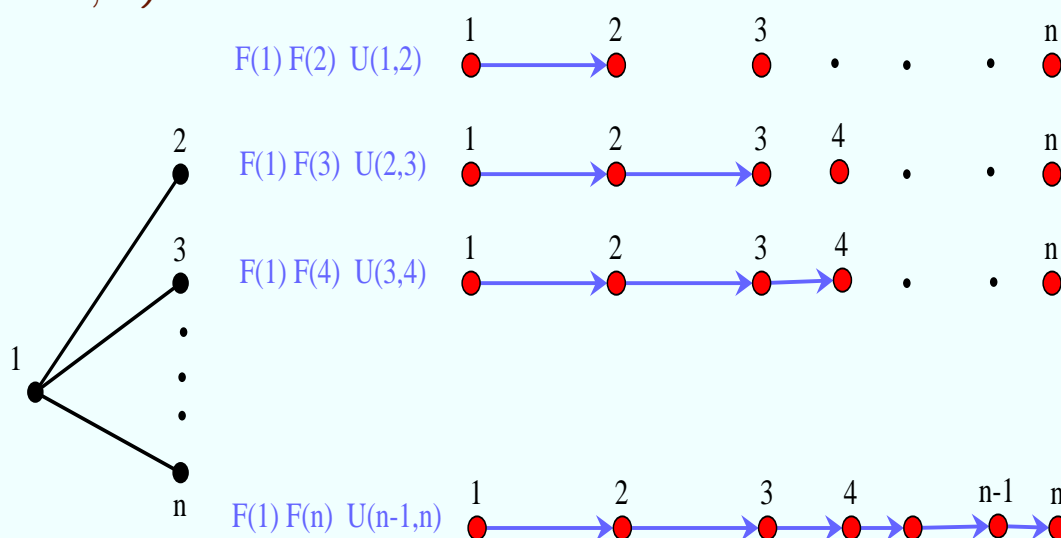
Evident, complexitatea procedurii *union* este $O(1)$. Funcția, *find* este în acest caz mai complicată:

```
function find( $v : V$ );  
    variabilă locală  $i : V$ ;  
     $i \leftarrow v$ ;  
    while  $pred[i] > 0$  do  $i \leftarrow pred[i]$ ;  
    return  $i$ 
```

Complexitatea lui $find(v)$ este $O(h(v))$ unde $h(v)$ este lungimea drumului (din arbore) de la vârful v la rădăcina arborelui care-l conține pe v .

Dacă graful G este $K_{1,n-1}$ desenat mai jos, și lista ordonată a muchiilor $E = \{12, 13, \dots, 1n\}$, atunci execuția algoritmului provoacă următorul șir de apeluri ale procedurii $union(U)$ și funcției $find(F)$:

$F(1), F(2), U(1, 2), F(1), F(3), U(2, 3), \dots, F(1), F(n), U(n - 1, n)$.



Apelurile $F(i) (i > 1)$ și $U(i, i + 1) \ i \geq 1$ necesită în total $O(n)$ operații. Șirul de $F(1)$ necesită însă $O(1) + O(2) + \dots + O(n - 1) = O(n^2)$ operații.

Este deci posibil ca pasul 2 al algoritmului în această implementare să fie de complexitate $\Omega(n^2)$ chiar dacă graful este rar.

Deficiența acestei implementări este datorată posibilității ca în procedura *union* să declarăm rădăcină nouă pentru cei doi arbori pe cea a celui cu mai puține vîrfuri, ceea ce are ca efect posibilitatea ca $h(v)$ să devină mare ($O(n)$) pe parcursul algoritmului.

Acest defect poate fi evitat dacă, la execuția lui *union* ținem seama de cardinalul celor două mulțimi.

Se poate memora cardinalul unei mulțimi în componenta tabloului *pred* corespunzătoare rădăcinii arborelui care memorează acea mulțime. Mai precis, considerăm inițializarea

1.3 for $v \in V$ **do** $pred[v] \leftarrow -1;$

și modificăm procedura *union* astfel încît să asigurăm îndeplinirea condiției

$pred[v] < 0 \Leftrightarrow v$ **este rădăcină a unui arbore și** $-pred[v]$ **este cardinalul mulțimii memorate în el.**

Procedura *union* are, în acest caz, tot complexitatea $O(1)$, dar selectează drept nouă rădăcină pe cea care corespunde arborelui cu mai multe vîrfuri:

```
procedure union( $v, w : V$ );
    //  $v$  și  $w$  sunt rădăcini
    variabila locală întreagă  $t$ 

     $t \leftarrow \text{pred}[v] + \text{pred}[w]$ ;
    if  $\text{pred}[v] > \text{pred}[w]$  then
        {  $\text{pred}[v] \leftarrow w; \text{pred}[w] \leftarrow t$  }
    else {  $\text{pred}[w] \leftarrow v; \text{pred}[v] \leftarrow t$  }
```

Cu această implementare a funcției *find* și procedurii *union*, pe tot parcursul algoritmului are loc:

$$(*) \quad \forall v \in V \quad - \text{pred}[\text{find}(v)] \geq 2^{h(v)}$$

(reamintim că $h(v)$ notează lungimea drumului de la v la rădăcina $\text{find}(v)$ a arborelui ce memorează v).

După inițializarea 1.3, $\forall v \in V \ h(v) = 0$ și $find(v) = v$ iar $-pred[v] = 1$, deci (*) are loc cu egalitate.

Dacă, înaintea unei iterații din pasul 2, (*) are loc, atunci, dacă în acea iterație nu se execută *union*, nu se modifică tabloul *pred* și deci (*) rămîne valabilă și după execuție.

Presupunem prin urmare că se apelează *union*(*x*, *y*) și că se execută $pred[y] := x$.

Aceasta înseamnă că înaintea acestei iterații avem $-pred[x] \geq -pred[y]$. Să observăm că singurele vîrfuri *v* cărora li se modifică $h(v)$ după execuția iterației curente sînt cele care înaintea iterației satisfăceau $find(v) = y$, pentru care aveam $-pred[y] \geq 2^{h(v)}$.

Dupa execuția iterației avem $h'(v) = h(v) + 1$ iar $find'(v) = x$, și deci trebuie să verificăm că $-pred'[x] \geq 2^{h'(v)}$. Avem $-pred'[x] = -pred[x] - pred[y] \geq 2 \cdot (-pred[y]) \geq 2 \cdot 2^{h(v)} = 2^{h(v)+1} = 2^{h'(v)}$.

Rezulta că (*) are loc pe tot parcursul algoritmului, deci, prin logaritmare obținem

$$\forall v \in V \quad h(v) \leq \log(-pred[find[v]]) \leq \log n.$$

Complexitatea pasului 2 va fi deci $O(n - 1 + 2m \log n) = O(m \log n)$ și deci tot algoritmul are complexitatea $O(m \log n)$ ceea ce-l face superior algoritmului lui Prim pentru grafuri rare.

Soluția a III^a. Complexitatea pasului 2, cu implementarea precedentă, este datorată apelurilor succesive ale lui *find*.

Tarjan (1976) a propus ca fiecare apel al lui *find* care necesită parcurgerea unui drum de lungime mai mare decât 1, să "distrugă" acest drum, aducându-i vîrfurile drept descendenți imediați ai rădăcinii, cu scopul ca apelurile viitoare ale lui *find* pentru aceste vîrfuri să nu mai consume timp. Mai precis, avem

```

function find( $v : V$ );
    variabile întregi locale  $i, j, k$ ;
     $i \leftarrow v$ ;
    while  $pred[i] > 0$  do  $i \leftarrow pred[i]$ ;
     $j \leftarrow v$ ;
    while  $pred[j] > 0$  do
    {  $k \leftarrow pred[j]$ ;  $pred[j] \leftarrow i$ ;  $j \leftarrow k$ ; }
    return  $i$ 

```

Dacă $A : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$ este funcția lui **Ackermann** dată de:

$A(i, 0) = 0 \quad \forall i \geq 0$;
 $A(i, 1) = 2 \quad \forall i \geq 1$;
 $A(0, x) = 2x \quad \forall x \geq 0$;
 $A(i + 1, x + 1) = A(i, A(i + 1, x)) \quad \forall i \geq 0 \quad \forall x \geq 1$,
 atunci, considerînd $\forall m \geq n > 0$

$$\alpha(m, n) = \min\{z \mid A(z, 4^{\lceil m/n \rceil}) \geq \log n, z \geq 1\}$$

avem:

Complexitatea pasului 2, utilizînd *union* din soluția a II-a și *find* descris mai sus, este $O(m \cdot \alpha(m, n))$.

Notăm că $\alpha(m, n)$ crește extrem de încet (pentru valorile practice ale lui n , $\alpha(m, n) \leq 3$) și deci se poate presupune ca această ultimă implementare este practic liniară (în raport cu m).

IV. Cuplaje (de cardinal maxim).

Fie $G = (V, E)$ un (multi)graf. Dacă $A \subseteq E$ și $v \in V$, vom nota cu

$$d_A(v) = |\{e \mid e \in A, e \text{ incidentă cu } v\}|,$$

adică gradul vârfului v în graful parțial $\langle A \rangle_G$.

Definiție. Se numește **cuplaj** (sau **mulțime independentă de muchii**) al grafului G , orice mulțime M de muchii cu proprietatea că $d_M(v) \leq 1, \forall v \in V$.

Vom nota cu \mathcal{M}_G familia cuplajelor grafului G :

$$\mathcal{M}_G = \{M \mid M \subseteq E, M \text{ cuplaj în } G\}.$$

Se observă că \mathcal{M}_G satisface proprietățile:

i) $\emptyset \in \mathcal{M}_G$.

ii) $M \in \mathcal{M}_G, M' \subseteq M \Rightarrow M' \in \mathcal{M}_G$.

Dacă $M \in \mathcal{M}_G$ atunci un vîrf $v \in V$ cu $d_M(v) = 1$ se numește **saturat** de cuplajul M . Mulțimea $S(M)$ a **vîrfurilor saturate de cuplajul M în graful G** , satisface $|S(M)| = 2|M|$.

Dacă $d_M(v) = 0$, atunci v se numește **expus** față de cuplajul M . Mulțimea $E(M)$ a vîrfurilor expuse față de cuplajul M satisface $E(M) = V - S(M)$ și $|E(M)| = |V| - 2|M|$.

Problema **cuplajului maxim** :

P1 *Dat $G = (V, E)$ un graf, să se determine $M^* \in \mathcal{M}_G$ astfel încît*

$$|M^*| = \max\{|M| \mid M \in \mathcal{M}_G\}.$$

(Vom nota cu $\nu(G) = \max\{|M| \mid M \in \mathcal{M}_G\}$).

Problema cuplajului maxim este strîns legată de **problema acoperirii minime**.

Definiție. Se numește **acoperire** (a vîrfurilor cu muchii) în graful G orice mulțime $F \subseteq E$ de muchii cu proprietatea că $d_F(v) \geq 1 \ \forall v \in V$.

$\mathcal{F}_G = \{F \mid F \subseteq E, F \text{ acoperire în } G\}$ notează familia acoperirilor grafului G .

$\mathcal{F}_G \neq \emptyset \Leftrightarrow G$ nu are vîrfuri izolate (atunci, măcar E este o acoperire).

Problema acoperirii minime este:

P2 *Dat $G = (V, E)$ un graf, să se determine $F^* \in \mathcal{F}_G$ astfel încât*

$$|F^*| = \min\{|F| \mid F \in \mathcal{F}_G\}.$$

Teoremă. 1. (Norman-Rabin 1959) *Fie $G = (V, E)$ un graf fără vîrfuri izolate, de ordin n . Dacă M^* este un cuplaj de cardinal maxim în G , iar F^* o acoperire de cardinal minim în G , atunci*

$$|M^*| + |F^*| = n.$$

Demonstrație: a) Fie M^* un cuplaj de cardinal maxim. Considerăm următorul algoritm:

```
 $F \leftarrow M^*;$   
for  $\forall v \in E(M)$  do  
  { determină  $v' \in S(M^*)$  astfel încît  $vv' \in E$ ;  
     $F \leftarrow F \cup \{vv'\}$   
  }
```

Să observăm că pentru $\forall v \in E(M^*)$, cum G nu are vîrfuri izolate, există o muchie incidentă cu v , și cum M^* este maximal în raport cu incluziunea, această muchie are cealaltă extremitate din $S(M^*)$. Mulțimea de muchii F astfel construită este o acoperire și în plus $|F| = |M^*| + |E(M^*)| = |M^*| + n - 2|M^*| = n - |M^*|$.

Rezultă că $|F^*| \leq |F| = n - |M^*|$.

b) Fie F^* o acoperire de cardinal minim. Considerăm următorul algoritm:

```

 $M \leftarrow F^*$ 
while  $\exists v \in V : d_M(v) > 1$  do
{   determină  $e \in M$  incidentă cu  $v$ ;
     $M \leftarrow M - e$ 
}

```

Algoritmul construiește un cuplaj M în G .

Dacă muchia e incidentă cu v , care se înlătură din M într-o iterație while, este $e = vv'$,

atunci $d_M(v') = 1$ și deci în pasul următor $d_M(v')$ va fi zero, adică la orice îndepărtare a unei muchii din mulțimea M curentă de muchii se obține un vîrf expus față de cuplajul final M (dacă vîrfurile v' ar fi incident cu încă o muchie, atunci din acoperirea inițială F^* se poate înlătura muchia e și să obținem tot o acoperire, contrazicînd alegerea lui F^*).

Deci dacă M este cuplajul construit de algoritm avem: $|F^*| - |M| = |E(M)| = n - 2|M|$, adică

$$|F^*| = n - |M| \geq n - |M^*|.$$

Din (a) și (b) rezultă concluzia teoremei.

Demonstrația făcută arată, chiar mai mult, că problemele (P1) și (P2) sînt polinomial echivalente, cuplajul M și acoperirea F construite fiind și ele soluții optime respectiv pentru cele două probleme.

Dacă vom considera matricea de incidență (vîrf - muchie) a grafului G cu n vîrfuri și m muchii $B = (b_{ij})_{n \times m}$ cu $b_{ij} = 1$ dacă vîrfurile i și muchia j sînt incidente și $b_{ij} = 0$ altminteri (într-o ordonare fixată a vîrfurilor și muchiilor), și dacă notăm cu e_p vectorul p -dimensional cu toate componentele 1, atunci cele două probleme se scriu analitic astfel

$$\mathbf{P1'} \quad \max\{e_m^T x \mid Bx \leq e_n, x \geq 0, x_i \in \{0, 1\} \ i = \overline{1, m}\}$$

$$\mathbf{P2'} \quad \min\{e_m^T x \mid Bx \geq e_n, x \geq 0, x_i \in \{0, 1\} \ i = \overline{1, m}\}$$

și teorema 1 oferă o egalitate min-max interesantă.

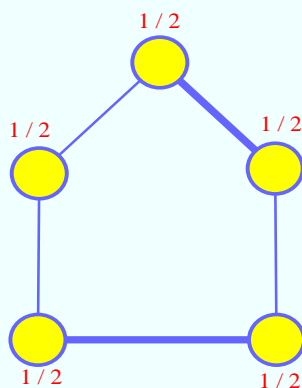
În cele ce urmează ne vom ocupa numai de problema (P1); în plus vom presupune că G nu are vîrfuri izolate.

Dacă s-ar încerca rezolvarea problemei (P1') apelând la problema de programare liniară asociată

$$(LP1') \quad \max\{e'_m x \mid Bx \leq e_n, x \geq 0\}$$

se constată că soluțiile optime pot să nu fie cu componente întregi și, chiar mai mult, valoarea maximă determinată de (LP1') să fie superioară lui $\nu(G)$.

Cel mai simplu exemplu în acest sens este $G = C_{2n+1}$. Evident, $\nu(C_{2n+1}) = n$ și totuși $x_i = \frac{1}{2} \forall i = 1, 2n+1$ este o soluție optimă a lui (LP1') corespunzătoare, cu valoarea optimă $n + \frac{1}{2} > n$.



Rezultă că existența circuitelor impare în graful G poate provoca dificultăți în rezolvarea problemei (P1). Mai precis, avem următoarea teoremă:

Teoremă. 2. (Balinski 1971) *Vîrfurile polipodului $Bx \leq e_n, x \geq 0, x \in \mathbb{R}^m$, au coordonatele 0, 1 și $\frac{1}{2}$. Coordonatele $\frac{1}{2}$ apar dacă și numai dacă G are circuite impare.*

Rezultă de aici că, în cazul grafurilor bipartite, problema (P1) este ușor de rezolvat: se apelează la problema de programare (LP1') și soluția găsită este soluție optimă pentru problema P1 (reprezentînd vectorul caracteristic al unui cuplaj). Adaptarea combinatorie a algoritmului simplex din programarea liniară, direct pe graful bipartit considerat (în scopul unei economii de memorie, tablourile simplex sînt reprezentate implicit) a condus la așa numita "metodă ungară" de rezolvare a problemei (P1) pentru grafurile bipartite. Nu vom prezenta acest algoritm, preferînd descrierea unuia mai performant datorat lui *Hocroft și Karp (1973)*.

Totuși teorema de dualitate din programarea liniară, precum și integritatea soluțiilor optime, pot oferi demonstrații instantanee pentru teoreme de caracterizare a soluțiilor optime ale problemei (P1) în cazul grafurilor bipartite:

Teoremă. 3. (Hall, 1935) *Fie $G = (R, S; E)$ un graf bipartit. Există un cuplaj care saturează vîrfurile lui R dacă și numai dacă*

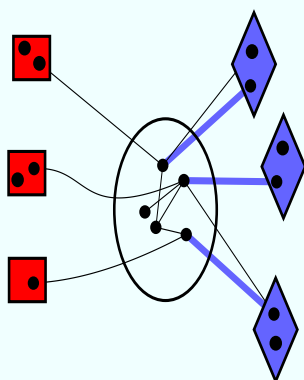
$$|N_G(A)| \geq |A| \quad \forall A \subseteq R.$$

Teoremă. 4. (Konig, 1930) *Fie $G = (R, S; E)$ un graf bipartit. Cardinalul maxim al unui cuplaj este egal cu numărul minim de vîrfuri prin îndepărtarea cărora se obține graful nul:*

$$\nu(G) = n - \alpha(G)$$

Revenind la problema (P1) cu G un graf oarecare și observînd că $\nu(G) \leq \frac{1}{2}|V(G)|$, rezultă că este interesant de caracterizat grafurile cu proprietatea că admit un cuplaj M astfel încît $S(M) = V(G)$. Un astfel de cuplaj se numește **cuplaj perfect** sau **1-factor**.

Este evident că un graf care are un cuplaj perfect are în orice componentă conexă un număr par de vîrfuri. Mai mult, dacă $S \subseteq V(G)$ atunci, în ipoteza că G are un cuplaj perfect, va trebui ca pentru fiecare componentă conexă cu un număr impar de vîrfuri a grafului $G - S$ să existe o muchie în cuplajul perfect cu o extremitate în S și cealaltă în componenta conexă impară. Rezultă că numărul componentelor conexe impare ale grafului $G - S$ nu poate depăși $|S|$.



Dacă pentru un graf oarecare H notăm cu $q(H)$ numărul componentelor conexe impare ale lui H , atunci observația anterioară arată că o condiție necesară pentru ca G să aibă un cuplaj perfect este ca $q(G-S) \leq |S| \quad \forall S \subseteq V(G)$.

Să observăm că atunci când $S = \emptyset$ condiția anterioară cere ca orice componentă conexă a lui G să aibă un număr par de vîrfuri. Condiția este și suficientă, așa cum rezultă din următoarea teoremă.

Teoremă. 5. (Tutte, 1947) *Un graf $G = (V, E)$ are un cuplaj perfect dacă și numai dacă*

$$(T) \quad q(G - S) \leq |S| \quad \forall S \subseteq V.$$

Demonstrație. Arătăm, prin inducție după $n = |V|$, că, dacă $G = (V, E)$ satisface (T), atunci G are un cuplaj perfect.

Cum teorema se verifică imediat pentru $n = 1, 2$, vom presupune în pasul inductiv că orice graf G' cu $|G'| < n$ și care satisface (T) are un cuplaj perfect.

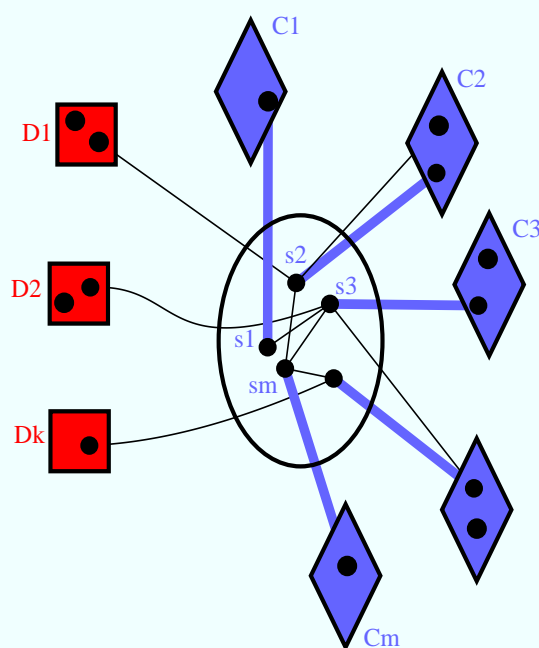
Fie $G = (V, E)$ cu $|V| = n$ și care satisface (T).

Alegem $S_0 \subseteq V(G)$ astfel încât în (T) avem egalitate: $q(G - S_0) = |S_0|$ și oricare ar fi S astfel încât $S_0 \subseteq S$, $S_0 \neq S$ avem $q(G - S) < |S|$.

Cum G satisface T , rezultă că G are numai componente conexe de cardinal par. Într-o astfel de componentă conexă, există întotdeauna un vîrf v_0 care nu este vîrf de articulație (orice vîrf pendent al unui arbore parțial) și se verifică imediat că $q(G - \{v_0\}) = 1 = |\{v_0\}|$.

Prin urmare familia $\{S \mid S \subseteq V, q(G - S) = |S|\}$ este nevidă și orice element maximal al ei poate fi considerat S_0 .

Fie $m = |S_0| > 0$ și C_1, \dots, C_m componentele conexe de cardinal impar ale lui $G - S_0$, iar D_1, \dots, D_k componentele conexe de cardinal par ($k \geq 0$) ale lui $G - S_0$.



Vom construi un cuplaj perfect al grafului G compus din :

- a)** - câte un cuplaj perfect în fiecare componentă conexă pară D_i ;
- b)** - un cuplaj format din m muchii, muchia e_i ($i = 1, m$) avînd o extremitate $s_i \in S_0$ și cealaltă $v_i \in C_i$;
- c)** - câte un cuplaj perfect în fiecare subgraf $[C_i - v_i]_G$.

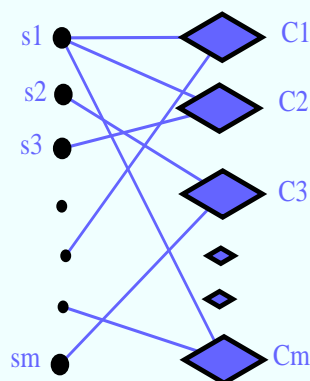
a) Pentru orice $i = 1, k$ graful $[D_i]_G$ are un cuplaj perfect.

În adevăr, cum $m > 0$, rezultă că $|D_i| < n$ și deci pentru a dovedi existența cuplajului perfect în $[D_i]_G$ este suficient să dovedim (din ipoteza inductivă) că $G' = [D_i]_G$ satisface (T).

Fie $S \subseteq D_i$. Dacă $q(G' - S) > |S|$ atunci $q(G - S_0 \cup S) = q(G - S_0) + q(G' - S) = |S_0| + q(G' - S) > |S_0| + |S| = |S_0 \cup S|$, contrazicînd faptul că G satisface (T).

Avem $q(G' - S) \leq |S|$ și deci G' satisface T.

b) Fie $H = (S_0, \{C_1, \dots, C_m\}; E')$ graful bipartit avînd o clasă a bipartiției S_0 , cealaltă clasă, mulțimea componentelor conexe impare ale lui $G - S_0$, iar mulțimea muchiilor E' formată din perechile sC_i ($s \in S$; $i = 1, m$) cu proprietatea că există $v \in C_i$ cu $sv \in E(G)$.



Acest graf are un cuplaj perfect. În adevăr, este suficient să arătăm că H satisface condiția lui Hall de existență a unui cuplaj M_0 ce saturează vîrfurile lui $\{C_1, \dots, C_m\}$:

$$\forall A \subseteq \{C_1, \dots, C_m\}, |N_H(A)| \geq |A|$$

și cum $|S_0| = m$ va rezulta că M_0 este cuplaj perfect în H .

Fie $A \subseteq \{C_1, \dots, C_m\}$.

Observăm că $B = N_H(A) \subseteq S_0$ și din definiția lui H , în graful G nu avem muchii de la un vîrf $v \in S_0 - B$ la un vîrf $v \in C_i$ cu $C_i \in A$. Deci componentele conexe impare C_i din A ale grafului $G - S_0$ vor rămîne componente conexe impare și în $G - B$.

Rezultă că $q(G - B) \geq |A|$. Pe de altă parte G satisface condiția lui Tutte (T) deci $|B| \geq q(G - B)$. Am obținut deci $|B| \geq |A|$ adică $|N_H(A)| \geq |A|$.

Cum A a fost aleasă arbitrar, rezultă că H are un cuplaj perfect $M_0 = \{s_1v_1, s_2v_2, \dots, s_mv_m\}$ cu $S_0 = \{s_1, \dots, s_m\}$ și $v_i \in C_i$ $i = 1, m$.

c) Pentru orice $i \in \{1, \dots, m\}$ graful $G' = [C_i - v_i]_G$ are un cuplaj perfect.

Folosind ipoteza inductivă, afirmația va rezulta dacă dovedim că G' satisface (T).

Fie $S \subseteq C_i - v_i$. Dacă $q(G' - S) > |S|$ atunci cum $q(G' - S) + |S| \equiv 0 \pmod{2}$, rezultă că $q(G' - S) \geq |S| + 2$ și atunci considerînd $S' = S_0 \cup \{v_i\} \cup S$, avem $|S'| \geq q(G - S') = q(G - S_0) - 1 + q(G' - S) = |S_0| - 1 + q(G' - S) \geq |S_0| - 1 + |S| + 2 = |S'|$, adică $q(G - S') = |S'|$ ceea ce contrazice alegerea lui S_0 căci $S_0 \subset S'$. Rezultă că $\forall S' \subseteq C_i - v_i$ $q(G' - S) \leq |S|$ deci G' are un cuplaj perfect.

Evident, cuplajul lui G obținut prin reuniunea cuplajelor puse în evidență în a), b), și c) de mai sus este un cuplaj perfect și cu aceasta teorema este demonstrată.

Notăm că *Berge* (1958) a generalizat această teoremă stabilind că

$$\nu(G) = \frac{1}{2}(|V(G)| - \max_{S \subseteq V(G)} [q(G - S) - |S|]).$$

Totuși, algoritmi care rezolvă problema (P1) se bazează pe o caracterizare mai simplă a cuplajelor de cardinal maxim.

Fie $G = (V, E)$ un graf și $M \in \mathcal{M}_G$ un cuplaj al său.

Definiție: Se numește **drum alternat al lui G relativ la cuplajul M** orice drum

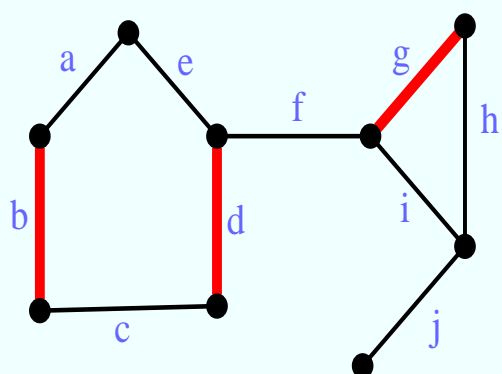
$$P : v_0, v_0v_1, v_1, \dots, v_{k-1}, v_{k-1}v_k, v_k$$

$$\text{a. } \hat{1}. \quad \forall i = \overline{1, k-1} \quad \{v_{i-1}v_i, v_iv_{i+1}\} \cap M \neq \emptyset.$$

Să observăm că, întrucât M este cuplaj, din definiția dată rezultă că **dintre orice două muchii consecutive ale drumului P exact una aparține cuplajului** (muchiiile lui P aparțin alternativ la M și $E - M$).

Vom desemna, în cele ce urmează, prin P mulțimea muchiilor drumului P (pentru simplificarea notațiilor).

Definiție: Se numește **drum de creștere al lui G relativ la cuplajul M** un drum alternat cu extremitățile vîrfuri distincte, expuse relativ la cuplajul M .



a, b, c, d - drum alternat par

f - drum alternat impar

j - drum de crestere

g, f, d - drum alternat impar

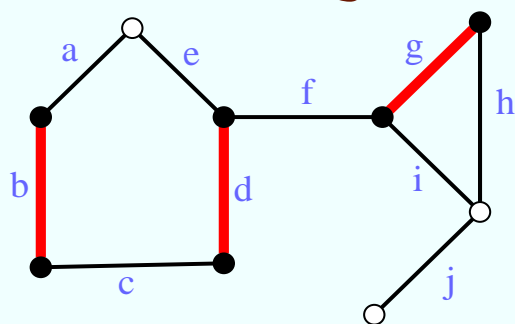
a, b, c, d, e - drum alternat inchis

a, b, c, d, f, g, h - drum de crestere

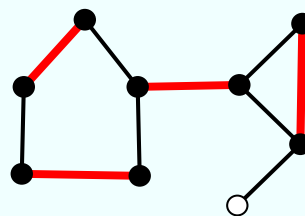
Observație: Din definiție, rezultă că dacă P este un drum de creștere relativ la cuplajul M atunci $|P - M| = |P \cap M| + 1$.

Teoremă. 6. (Berge 1959) *Un cuplaj M este de cardinal maxim în graful G dacă și numai dacă nu există în G drumuri de creștere relativ la M .*

Demonstrație: Dacă M este un cuplaj de cardinal maxim și P ar fi un drum de creștere în G relativ la M atunci $M' = P \Delta M = (P - M) \cup (M - P)$ este un cuplaj în G . (Construcția lui $P \Delta M$ revine la interschimbarea muchiilor lui $M - P$ și $P - M$ pe drumul P). În plus, $|M'| = |P \cap M| + 1 + |M - P| = |M| + 1$, contrazicând alegerea lui M .



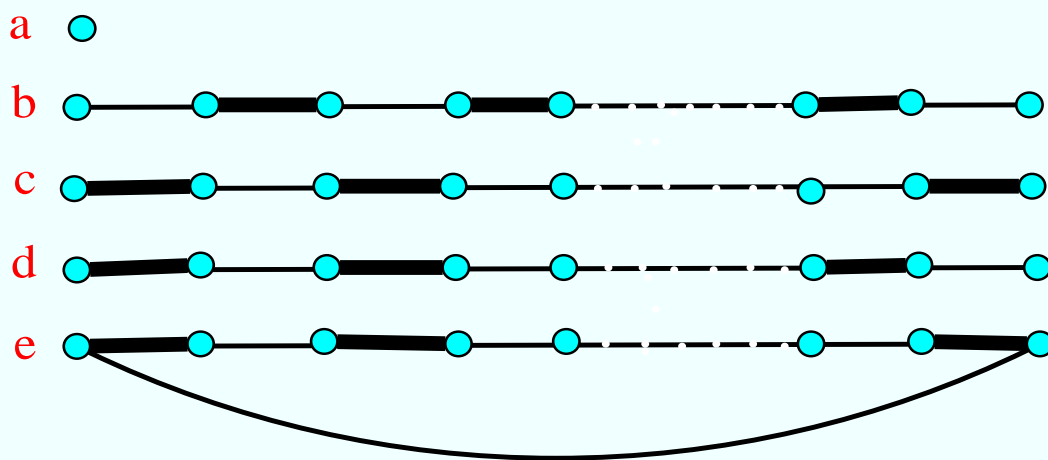
$P = a, b, c, d, f, g, h$ - drum de creștere



$(M-P) \cup (P-M)$

Reciproc, dacă M este cuplaj în G și M nu admite drumuri de creștere, considerăm M^* un cuplaj de cardinal maxim. Vom arăta că $|M^*| = |M|$, ceea ce demonstrează teorema.

Fie G' graful parțial al lui G , $G' = (V, M \Delta M^*)$. Evident, $d_{G'}(v) \leq 2 \ \forall v \in V$ și deci componentele conexe ale lui G' sînt drumuri (eventual de lungime 0), sau circuite și avem următoarele posibilități (muchii îngroșate sînt din M^* și cele subțiri, din M):



Situația b) nu poate avea loc, pentru că reprezintă un drum de creștere relativ la M^* care-i cuplaj de cardinal maxim. Situația c) nu are loc, întrucît M nu admite drumuri de creștere.

Deci, dacă notăm cu $m_M(C)$ numărul muchiilor din M ale componentei conexe C a lui G' și cu $m_{M^*}(C)$ numărul corespunzător, de muchii din M^* avem $m_M(C) = m_{M^*}(C)$. Rezultă că

$$|M - M^*| = \sum_C m_C(M) = \sum_C m_C(M^*) = |M^* - M|$$

(suma se face după toate componentele conexe C ale grafului G') și deci, $|M| = |M^*|$.

Această teoremă justifică următoarea strategie de construire a unui cuplaj de cardinal maxim:

- a) fie M un cuplaj oarecare a lui G (eventual $M = \emptyset$);
- b) **while** $\exists P$ drum de creștere relativ la M **do**
 $M \leftarrow M \Delta P$

La fiecare iterație a ciclului while, cuplajul curent "crește" (cardinalul său se mărește cu o unitate) și deci în cel mult $\frac{n}{2}$ iterații se obține un cuplaj de cardinal maxim, care nu va admite drumuri de creștere.

Neajunsul acestui algoritm este acela că testul de oprire a ciclului - **inexistența drumului de creștere** - poate conduce la un număr exponențial de operații.

Primul care a demonstrat posibilitatea implementării acestui algoritm astfel încât numărul total de operații să fie polinomial în raport cu numărul de vârfuri ale grafului G , a fost **Edmonds (1965)**, obținând astfel unul din primele rezultate ale teoriei complexității (cantitative) a algoritmilor.

În 1973 **Hopcroft și Karp**, fac o analiză mai detaliată a procesului succesiv de creșteri ale cuplajului curent, cu ipoteza suplimentară că se alege de fiecare dată **un drum de creștere cu număr minim de muchii** (printre toate drumurile de creștere posibile). Această idee stă la baza celui mai eficient algoritm, cunoscut, pentru rezolvarea problemei.

Lemă. 1. Fie $M, N \in \mathcal{M}_G$, $|M| = r$, $|N| = s$ și $s > r$. Atunci în $M \Delta N$ există cel puțin $s - r$ drumuri de creștere ale cuplajului M , disjuncte ca vîrfuri.

Demonstrație: Fie $G' = (V, M \Delta N)$ graful parțial secționat în G de $M \Delta N$.

Fie C_i $i = 1, p$ componentele conexe ale lui G' .
Definim pentru fiecare $i \in \{1, \dots, p\}$

$$\delta(C_i) = |E(C_i) \cap N| - |E(C_i) \cap M|.$$

Observăm că:

$\delta(C_i) \in \{-1, 0, 1\}$ (M, N cuplaje, $\Rightarrow C_i$ sînt drumuri sau circuite);

$\delta(C_i) = 1$ dacă și numai dacă C_i este un drum de creștere relativ la M .

În plus,

$$\sum_{i=1, p} \delta(C_i) = |N - M| - |M - N| = s - r.$$

Rezultă că există măcar $s - r$ C_i cu $\delta(C_i) = 1$, adică există măcar $s - r$ drumuri de creștere disjuncte ca vîrfuri (deci și ca muchii) conținute în $M \Delta N$.

Lemă. 2. Dacă $\nu(G) = s$ și $M \in \mathcal{M}_G$ $|M| = r < s$, atunci există în G un drum de creștere relativ la M de lungime $\leq 2\lfloor r/(s-r) \rfloor + 1$.

Demonstrație. Fie $N \in \mathcal{M}_G$ cu $|N| = s = \nu(G)$. Conform lemei precedente, vor exista $s - r$ drumuri de creștere disjuncte pe muchii, conținute în $M \Delta N$. Acestea au împreună cel mult r muchii din M . Rezultă că există unul care conține cel mult $\lfloor r/(s-r) \rfloor$ muchii din M , a cărei lungime este deci $\leq 2\lfloor r/(s-r) \rfloor + 1$.

Definiție Dacă $M \in \mathcal{M}_G$, se numește **drum minim de creștere al lui M în G** , un drum de creștere cu număr minim de muchii printre toate drumurile de creștere ale lui M în G .

Lemă. 3. Fie $M \in \mathcal{M}_G$, P drum minim de creștere relativ la M , și P' drum de creștere al lui $M \Delta P$. Atunci, $|P'| \geq |P| + 2|P \cap P'|$.

Demonstrație: Fie $N = (M \Delta P) \Delta P'$.

Avem $M \Delta N = P \Delta P'$ și $|N| = |M| + 2$.

Folosind lema 1 obținem că există P_1, P_2 drumuri de creștere disjuncte ca muchii relativ la M conținute în $M \Delta N$. Cum P este drum minim de creștere avem: $|P \Delta P'| \geq |P_1| + |P_2| \geq 2|P|$ deci $|P| + |P'| - 2|P \cap P'| \geq 2|P|$.

Considerăm următorul algoritm (*):

$M_0 \leftarrow \emptyset$;

$M_{i+1} \leftarrow M_i \Delta P_i$

(P_i dr. minim de creșt. rel. la M_i ; $i \geq 0$.)

Se obține șirul de drumuri minime de creștere $P_0, P_1, \dots, P_{\nu(G)-1}$.

Lemă. 4.

a) $\forall i = 1, \nu(G) - 2 \quad |P_i| \leq |P_{i+1}|$;

$|P_i| = |P_{i+1}| \Leftrightarrow P_i$ și P_{i+1} sînt disjuncte ca vîrfuri.

b) $\forall i < j < \nu(G) - 1 \quad |P_i| = |P_j|$, implică P_i și P_j sînt disjuncte ca vîrfuri.

Demonstrție: a) Considerînd $P = P_i$ și $P' = P_{i+1}$ în lema 3 se obține $|P_{i+1}| \geq |P_i| + 2|P_i \cap P_{i+1}| \geq |P_i|$. Egalitatea are loc dacă și numai dacă P_i și P_{i+1} sînt disjuncte ca muchii, condiție care, avînd în vedere alternanța drumurilor, implică faptul că nu au vîrfuri comune.

b) rezultă aplicînd succesiv a).

Teoremă. 7. (Hopcroft, Karp 1973) Fie G un graf și $\nu(G) = s$. Numărul întregilor distincți din șirul $|P_0|, |P_1|, \dots, |P_{s-1}|$, construit în algoritmul de mai sus este cel mult $2\lceil\sqrt{s}\rceil + 2$.

Demonstratie: Fie $r = \lfloor s - \sqrt{s} \rfloor$. Atunci $|M_r| = r$ și $|P_r| \leq 2\lfloor r/(s-r) \rfloor + 1 = 2\lfloor \lfloor s - \sqrt{s} \rfloor / (s - \lfloor s - \sqrt{s} \rfloor) \rfloor + 1 < 2\lceil\sqrt{s}\rceil + 1$.

Rezultă că oricare ar fi $i < r$, $|P_i|$ este unul din cei $\lceil\sqrt{s}\rceil + 1$ întregi impari $\leq 2\lceil\sqrt{s}\rceil + 1$. În subșirul $|P_r|, \dots, |P_{s-1}|$ mai sînt cel mult $s - r \leq \lceil\sqrt{s}\rceil + 1$ întregi distincți, deci în total avem cel mult $2\lceil\sqrt{s}\rceil + 2$ întregi distincți.

Dacă algoritmul (*) se descompune în etape, astfel încât la fiecare etapă se determină o mulțime maximală de drumuri minime de creștere disjuncte ca vîrfuri, din lema 4, rezultă că, în etapa următoare, lungimea drumurilor minime de creștere utilizate va crește strict (altfel s-ar contrazice maximalitatea mulțimii de drumuri alese).

Utilizînd teorema 7, rezultă că numărul fazelor nu va depăși $2\lceil\sqrt{\nu(G)}\rceil + 2$.

Rezultă că următorul algoritm pentru aflarea unui cuplaj maxim, într-un graf cu măcar o muchie:

0. $M \leftarrow \emptyset$;

1. **repeat**

Determină \mathcal{P} o familie maximală (\subseteq)
de drumuri minime de creștere;

for $P \in \mathcal{P}$ **do** $M \leftarrow M \Delta P$

until $\mathcal{P} = \emptyset$.

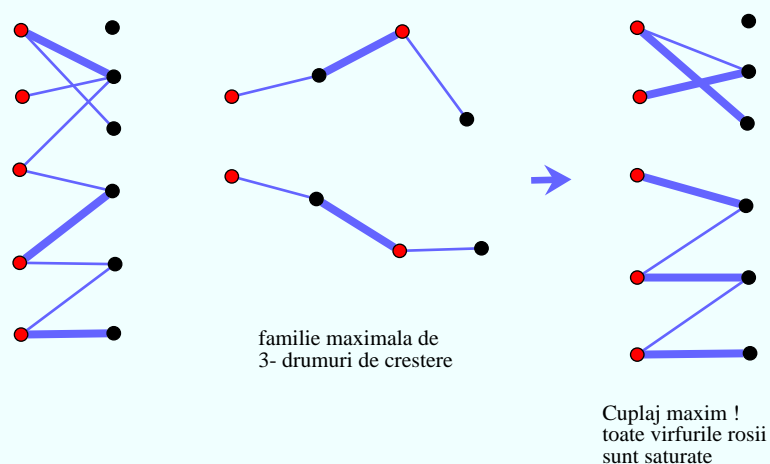
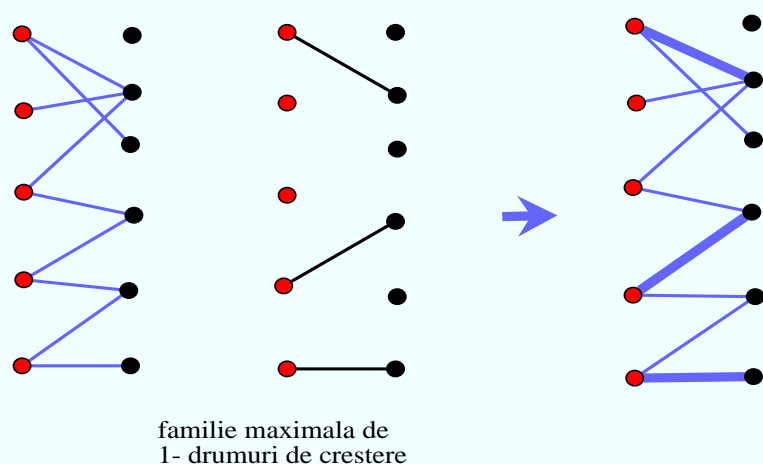
are complexitatea $O(\sqrt{n}A)$ unde A este complexitatea determinării familiei \mathcal{P} .

Hopcroft și Karp au arătat cum se poate implementa pasul 1 pentru un graf bipartit, astfel încât $A = O(m + n)$, deci s-a obținut un algoritm de complexitate $O(mn^{1/2})$ pentru aflarea unui cuplaj de cardinal maxim într-un graf bipartit.

Pentru un graf oarecare, structurile de date necesare obținerii aceleiași complexități sînt mult mai elaborate și au fost descrise de **Micali și Vazirani 1980**. Considerăm, în continuare, numai cazul grafurilor bipartite.

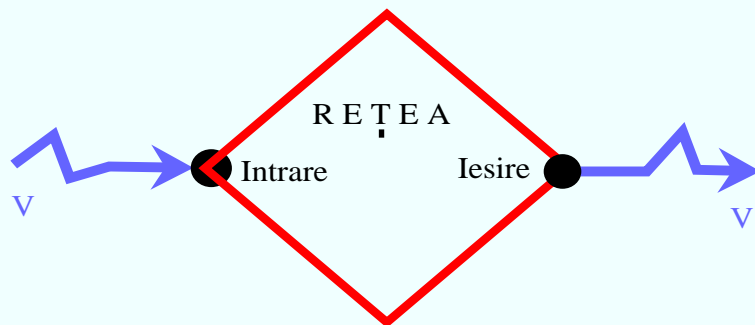
Fie $G = (R, S; E)$ un graf bipartit și $M \in \mathcal{M}_G$. Pornind din una din clase, R de exemplu, se consideră mulțimea extremităților drumurilor de creștere posibile, $R \cap E(M)$, și din fiecare astfel de vîrf, se începe construcția în paralel, de drumuri alternate.

Prima depistare a unui drum de creștere (sau, concluzia de inexistență) oprește construcția, oferind lungimea minimă a unui drum de creștere, și un sistem evident de etichetare va permite depistarea familiei \mathcal{P} . Complexitatea $O(m + n)$ rezultă prin utilizarea listelor de adiacență. Exemplu:



V. FLUXURI ÎN REȚELE

1. Problema fluxului maxim.



Data Reteaua, care este v MAXIM ?

Numim **rețea** (de transport) cu **intrarea** s și **ieșirea** t , 4-uplul $\mathbf{R} = (G, s, t, c)$ unde:

- $G = (V, E)$ este un digraf,
- $s, t \in V$; $s \neq t$; $d_G^+(s) > 0$; $d_G^-(t) > 0$,
- $c : E \rightarrow \mathbf{R}_+$; $c(e)$ este **capacitatea** arcului e .

Vom presupune că

$V = \{1, 2, \dots, n\}$ ($n \in \mathbf{N}^*$) și că $|E| = m$.

Extindem funcția c la $c : V \times V \rightarrow \mathbf{R}_+$ prin

$$c((i, j)) = \begin{cases} c(ij) & \text{dacă } ij \in E \\ 0 & \text{dacă } ij \notin E \end{cases}$$

și vom nota $c((i, j)) = c_{ij}$.

Definiție: Numim **flux** în rețeaua $R = (G, s, t, c)$ o funcție $x : V \times V \rightarrow \mathbf{R}$, care satisface

$$(i) \quad 0 \leq x_{ij} \leq c_{ij} \quad \forall ij \in V \times V$$

$$(ii) \quad \sum_{j \in V} x_{ji} - \sum_{j \in V} x_{ij} = 0 \quad \forall i \in V - \{s, t\}.$$

Observații

1⁰ Dacă $ij \in E$ atunci x_{ij} se numește **fluxul** (transportat) **pe arcul** ij .

Evident, condiția (i) cere ca **fluxul pe orice arc să fie nenegativ și subcapacitar**, iar condiția (ii) (*legea de conservare a fluxului*) cere ca **suma fluxurilor pe arcele care intră în vârful i să fie egală cu suma fluxurilor pe arcele care ies din vârful i** .

Se putea cere ca fluxul să fie definit numai pe arcele rețelei, dar cu convenția făcută la extensia funcției de capacitate, se observă că pentru perechile (i, j) care nu sînt arce în rețea condiția (i) impune ca fluxul să fie 0, și evident cele două definiții sînt echivalente.

O preferăm pe cea dată, pentru simplitatea notațiilor, deși în implementări, structurile de date folosite vor ignora perechile (i, j) care nu sînt arce în rețea.

2^0 Dacă se sumează relațiile (ii) (pentru $i \in V - \{s, t\}$) se obține:

$$0 = \sum_{i \neq s, t} \left(\sum_{j \in V} x_{ji} - \sum_{j \in V} x_{ij} \right) =$$

$$\sum_{i \neq s, t} \sum_{j \neq s, t} x_{ji} - \sum_{i \neq s, t} \sum_{j \neq s, t} x_{ij} +$$

$$\sum_{i \neq s, t} x_{si} + \sum_{i \neq s, t} x_{ti} - \sum_{i \neq s, t} x_{is} - \sum_{i \neq s, t} x_{it} =$$

$$- \left(\sum_i x_{is} - \sum_i x_{si} \right) - \left(\sum_i x_{it} - \sum_i x_{ti} \right), \text{ adică}$$

$$\sum_{j \in V} x_{jt} - \sum_{j \in V} x_{tj} = - \left(\sum_{j \in V} x_{js} - \sum_{j \in V} x_{sj} \right).$$

Definiție: Dacă x este un flux în rețeaua $R = (G, s, t, c)$ se numește **valoarea fluxului** x numărul

$$v(x) = \sum_{j \in V} x_{jt} - \sum_{j \in V} x_{tj}.$$

$v(x)$ se poate interpreta ca fiind fluxul net care ajunge în ieșirea rețelei sau (conform egalității obținute mai sus) fluxul net care iese din intrarea rețelei.

În orice rețea $R = (G, s, t, c)$ există un flux, fluxul nul $x_{ij} = 0 \ \forall ij$, de valoare 0.

Problema fluxului maxim:

Dată $R = (G, s, t, c)$ o rețea, să se determine un flux de valoare maximă.

Observații: 1⁰. Problema se poate formula, evident, ca o problemă de programare liniară:

max v

$$\sum_j x_{ji} - \sum_j x_{ij} = 0, \ \forall i \neq s, t$$

$$\sum_j x_{js} - \sum_j x_{sj} = -v$$

$$\sum_j x_{jt} - \sum_j x_{tj} = v$$

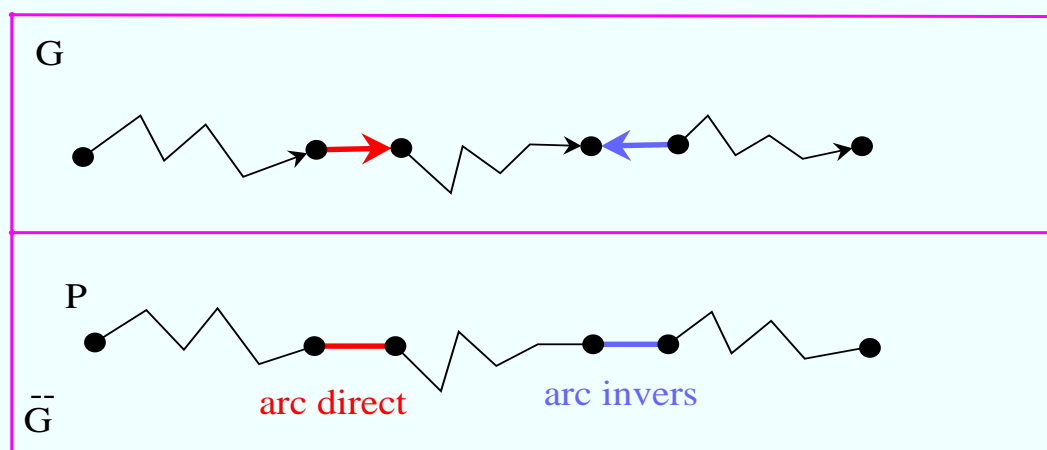
$$0 \leq x_{ij} \leq c_{ij} \ \forall ij$$

Particularitățile combinatorii ale problemei, numărul mare de restricții și mai ales dificultățile legate de restricțiile de integritate ce s-ar putea impune variabilelor, care uneori în practică sînt esențiale, au condus la dezvoltarea de metode specifice de rezolvare.

Definiție. Dacă P este un drum în \overline{G} , multi-graful suport al digrafului G , și $e = v_i v_j$ este o muchie a lui P atunci:

dacă e corespunde arcului $v_i v_j$ al lui G , e se numește **arc direct** al drumului P ;

dacă e corespunde arcului $v_j v_i$ al lui G , atunci e se numește **arc invers**.



Definiție. Fie $R = (G, s, t, c)$ și x flux în R . Se numește **C-drum** (în R relativ la fluxul x) un drum D în \overline{G} cu proprietatea că $\forall ij \in E(D)$:

$x_{ij} < c_{ij}$ dacă ij este arc direct,

$x_{ji} > 0$ dacă ij este arc invers.

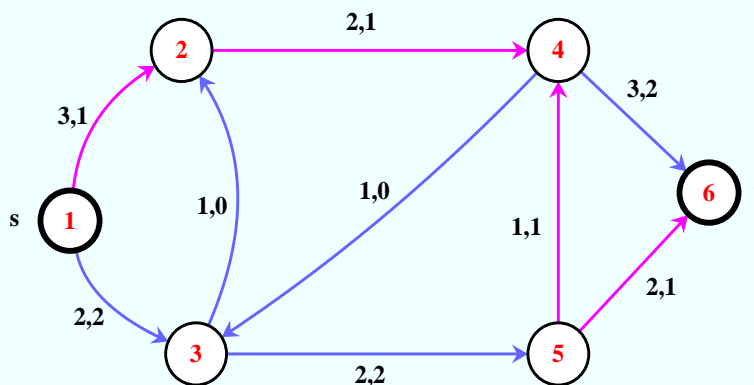
Dacă D este un C-drum și $ij \in E(D)$, se numește **capacitatea reziduală** a lui ij (relativ la C-drumul D) numărul

$$r(ij) = \begin{cases} c_{ij} - x_{ij} & \text{dacă } ij \text{ arc direct în } D \\ x_{ji} & \text{dacă } ij \text{ arc invers în } D . \end{cases}$$

Capacitatea reziduală a drumului D este

$$r(D) = \min_{e \in E(D)} r(e).$$

Exemplu: Fie rețeaua de mai jos, în care pe arce este precizată mai întâi capacitatea și apoi fluxul:



Atunci $D : 1, 12, 2, 24, 4, 45, 5, 56, 6$ este un C-drum de la s la t cu arcele directe $12(x_{12} = 1 < c_{12} = 3)$; $24(x_{24} = 1 < c_{24} = 2)$; $56(x_{56} = 1 < c_{56} = 2)$ și arcul invers $45(x_{54} = 1 > 0)$. Capacitatea reziduală a lui D este $r(D) = \min(\min(2, 1, 1), 1) = 1$.

Definiție. Se numește **drum de creștere** a fluxului x , în rețeaua $R = (G, s, t, c)$, un C-drum de la s la t .

Lemă. 1. Dacă D este un drum de creștere a fluxului x în rețeaua $R = (G, s, t, c)$, atunci $x^1 = x \otimes r(D)$ definit prin

$$x_{ij}^1 = \begin{cases} x_{ij} & \text{dacă } \overline{ij} \notin E(D) \\ x_{ij} + r(D) & \text{dacă } ij \in E(D), ij \text{ arc direct în } D \\ x_{ij} - r(D) & \text{dacă } ji \in E(D), ji \text{ arc invers în } D \end{cases}$$

este flux în R și $v(x^1) = v(x) + r(D)$.

Demonstrație. Definiția lui $r(D)$ implică îndeplinirea de către x^1 , a condițiilor (i).

Condițiile (ii) verificate de x , nu sînt afectate pentru niciun vîrf $i \notin V(D)$.

Dacă $i \neq s, t$ este un vîrf al drumului D , i este incident cu exact două arce ale lui D , fie ele li și ik .

Avem următoarele cazuri posibile:

a) li și ik arce directe:

$$\begin{aligned}\sum_j x_{ji}^1 - \sum_j x_{ij}^1 &= \sum_{j \neq l} x_{ji} - \sum_{j \neq k} x_{ij} + x_{li}^1 - x_{ik}^1 = \\ \sum_{j \neq l} x_{ji} - \sum_{j \neq k} x_{ij} + x_{li} + r(D) - x_{ik} - r(D) &= \\ \sum_j x_{ji} - \sum_j x_{ij} &= 0.\end{aligned}$$

b) li direct ik invers:

$$\begin{aligned}\sum_j x_{ji}^1 - \sum_j x_{ij}^1 &= \sum_{j \neq l, k} x_{ji} - \sum_j x_{ij} + x_{li}^1 + x_{ki}^1 = \\ \sum_{j \neq l, k} x_{ji} - \sum_j x_{ij} + x_{li} + r(D) + x_{ki} - r(D) &= \\ \sum_j x_{ji} - \sum_j x_{ij} &= 0.\end{aligned}$$

c) li invers, ik direct: similar cu b).

d) li invers, ik invers: similar cu a).

Valoarea fluxului x^1 se obține considerînd lt unicul arc al lui D incident cu t :

Dacă lt direct atunci

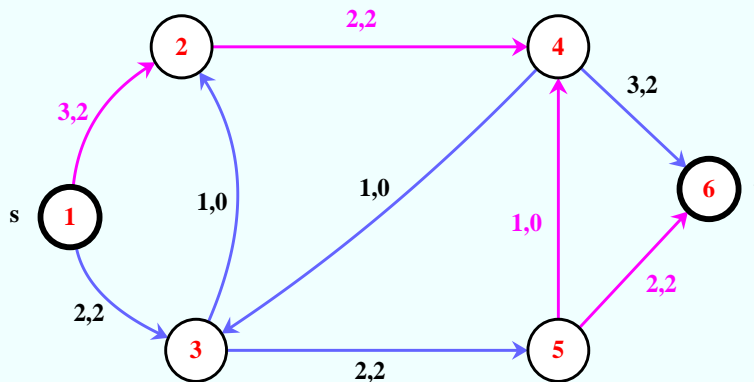
$$\begin{aligned}v(x^1) &= \sum_j x_{jt}^1 - \sum_j x_{tj}^1 = \sum_{j \neq l} x_{jt} - \sum_j x_{tj} + x_{lt}^1 = \\ \sum_{j \neq l} x_{jt} - \sum_j x_{tj} + x_{lt} + r(D) &= v(x) + r(D).\end{aligned}$$

Dacă lt invers atunci

$$\begin{aligned}v(x^1) &= \sum_j x_{jt}^1 - \sum_j x_{tj}^1 = \sum_j x_{jt} - \sum_{j \neq l} x_{tj} - x_{tl}^1 = \\ \sum_j x_{jt} - \sum_{j \neq l} x_{tj} - (x_{tl} - r(D)) &= v(x) + r(D).\end{aligned}$$

Deci lema are loc.

Pentru exemplul anterior, fluxul $x^1 = x \otimes r(D)$ de valoare 4 este precizat pe arce:



Observații: 1⁰ Această lemă justifică denumirea de drum de creștere, precum și cea de capacitate reziduală.

2⁰ Din definiție, dacă D este drum de creștere, $r(D) > 0$ și deci avem $v(x \otimes r(D)) > v(x)$.

Rezultă că

dacă x admite un drum de creștere atunci x nu este flux de valoare maximă.

Pentru a demonstra că si reciproc este adevărat avem nevoie de o nouă noțiune.

Definiție. Fie $R = (G, s, t, c)$. Se numește **secțiune** în rețeaua R , o partiție (S, T) a lui V cu $s \in S$ și $t \in T$.

Capacitatea secțiunii (S, T) este

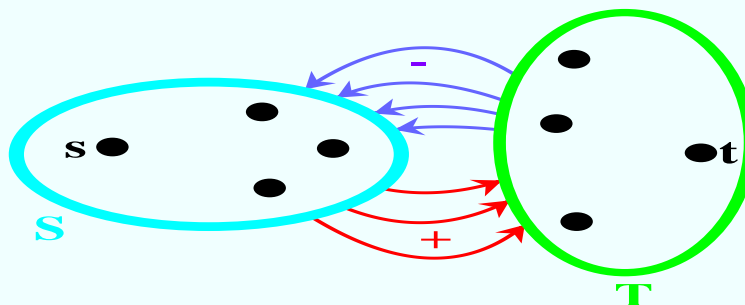
$$c(S, T) = \sum_{i \in S} \sum_{j \in T} c_{ij}$$

(suma capacităților arcelor de la S la T).

Lemă. 2. *Daca x este un flux în $R = (G, s, t, c)$ și (S, T) este o secțiune a rețelei, atunci*

$$v(x) = \sum_{i \in S} \sum_{j \in T} (x_{ij} - x_{ji}).$$

(valoarea fluxului este egală cu fluxul net ce trece prin orice secțiune.)



Demonstrație:

$$\begin{aligned} v(x) &= (\sum_j x_{jt} - \sum_j x_{tj}) - 0 \\ &= -(\sum_j x_{js} - \sum_j x_{sj}) - \sum_{i \in S, i \neq s} (\sum_j x_{ji} - \sum_j x_{ij}) \\ &= \sum_{i \in S} (\sum_j x_{ij} - \sum_j x_{ji}) \\ &= \sum_{i \in S} \sum_{j \in S} (x_{ij} - x_{ji}) + \sum_{i \in S} \sum_{j \in T} (x_{ij} - x_{ji}) \\ &= \sum_{i \in S} \sum_{j \in T} (x_{ij} - x_{ji}). \end{aligned}$$

Lemă. 3. *Dacă x este un flux în $R = (G, s, t, c)$ și (S, T) este o secțiune, atunci $v(x) \leq c(S, T)$.*

Demonstrație:

$$\begin{aligned} v(x) &= \sum_{i \in S} \sum_{j \in T} (x_{ij} - x_{ji}) \quad (\text{lema 2}) \\ &\leq \sum_{i \in S} \sum_{j \in T} (c_{ij} - x_{ji}) \quad (x_{ij} \leq c_{ij}) \\ &\leq \sum_{i \in S} \sum_{j \in T} c_{ij} \quad (x_{ji} \geq 0). \end{aligned}$$

Observații:

- 1) Dacă \bar{x} este un flux în $R = (G, s, t, c)$ și (\bar{S}, \bar{T}) o secțiune astfel încât $v(\bar{x}) = c(\bar{S}, \bar{T})$, atunci $\forall x$ flux în R $v(x) \leq c(\bar{S}, \bar{T}) = v(\bar{x})$, deci \bar{x} este flux de valoare maximă.
- 2) În exemplul dat, x^1 este flux de valoare maximă întrucât $v(x^1) = 4 = c(\{1, 2, 3\}, \{4, 5, 6\})$.

Teoremă. 1. (*Teorema drumului de creștere*)
Un flux x este de valoare maximă într-o rețea R , dacă și numai dacă, nu există drumuri de creștere a fluxului x în rețeaua R .

Demonstrație: O implicație este deja stabilită. Reciproc, fie x un flux în R care nu admite drumuri de creștere. Considerăm $S = \{i \mid i \in V \wedge \exists D \text{ C-drum în } R \text{ de la } s \text{ la } i\}$.

Evident $s \in S$ (există D de lungime 0) și $t \notin S$ (nu există C-drumuri de la s la t). Fie $T = V - S$. Rezultă că (S, T) este o secțiune.

Să observăm că $\forall i \in S$ și $\forall j \in T$ avem:

dacă $ij \in E$ atunci $x_{ij} = c_{ij}$ și

dacă $ji \in E$ atunci $x_{ji} = 0$

(altminteri C-drumul de la s la i se poate extinde la un C-drum de la s la j).

Deci, conform lemei 2, $v(x) = \sum_{i \in S} \sum_{j \in T} (x_{ij} - x_{ji}) = \sum_{i \in S} \sum_{j \in T} (c_{ij} - 0) = c(S, T)$ și prin urmare x este flux de valoare maximă.

Teoremă. 2. *(Teorema fluxului întreg)*

Dacă toate capacitățile sînt întregi, atunci există un flux de valoare maximă cu toate componentele întregi (flux întreg de valoare maximă).

Demonstrație: Fie algoritmul

- 1: $x^0 \leftarrow 0; i \leftarrow 0;$
- 2: **while** $(\exists P_i \text{ drum de creștere relativ la } x^i)$ **do**
 { $x^{i+1} \leftarrow x^i \otimes r(P_i);$
 $i \leftarrow i + 1$
 }

Se observă că " x^i are componente întregi" este un invariant al algoritmului (din definiția lui $r(P_i)$, dacă toate capacitățile sînt întregi, rezultă că $r(P_i)$ este întreg în ipoteza că x^i e întreg) și că la fiecare iterație a pasului 2 valoarea fluxului curent crește cu măcar o unitate, deci pasul 2 se repetă de cel mult $c(\{s\}, V - \{s\}) \in \mathbf{Z}_+$ ori. Fluxul final obținut este, conform teoremei 1, de valoare maximă.

Observație. Algoritmul, descris mai sus, este finit și în cazul capacităților raționale.

Teoremă. 3. (*Ford-Fulkerson, 1956*)

Valoarea maximă a unui flux în rețeaua $R = (G, s, t, c)$ este egală cu capacitatea minimă a unei secțiuni a rețelei.

Demonstrație: Dacă dispunem de un algoritm care, pornind de la un flux inițial x^0 (x^0 există întotdeauna, de exemplu $x^0 = 0$), construiește într-un număr finit de pași un flux x , care nu admite drumuri de creștere, atunci secțiunea construită în demonstrația teoremei 1 satisface împreună cu x enunțul teoremei.

Pentru cazul capacităților raționale algoritmul descris în demonstrația teoremei 2, satisface această condiție.

Pentru cazul capacităților reale vom prezenta, mai târziu, un astfel de algoritm, datorat lui Edmonds și Karp (1972).

Observații: i) În demonstrația teoremei 3 avem nevoie de fapt, doar să arătăm că există un flux x de valoare maximă și apoi să-i aplicăm construcția din demonstrația teoremei 1; existența fluxului x maxim rezultă imediat, considerînd transcrierea problemei fluxului maxim ca o problemă de programare liniară; am preferat demonstrația de mai sus care (deși va fi completată abia după analiza algoritmului lui Edmonds-Karp) este constructivă.

ii) Importanța algoritmică a teoremei 3 este evidentă: mulțimea secțiunilor rețelei este finită, pe cînd mulțimea fluxurilor din rețea este infinită.

Algoritmul lui Ford și Fulkerson pentru aflarea unui flux de valoare maximă

Se va folosi un procedeu de etichetare a vîrfurilor rețelei, în vederea depistării drumurilor de creștere a fluxului curent x . Dacă nu există drumuri de creștere, fluxul va fi de valoare maximă.

Eticheta atribuită unui vîrf $j \in V$ are trei componente (e_1, e_2, e_3) unde $e_1 \in V \cup \{0\}$; $e_2 \in \{direct, invers\}$; $e_3 \in \mathbf{R}_+$ și au următoarea semnificație:

- dacă $e_2 = direct$ și $e_1 = i$ atunci \exists un C-drum P de la s la j cu ultimul arc ij , arc direct și $r(P) = e_3$;
- dacă $e_2 = invers$ și $e_1 = i$ atunci \exists un C-drum P de la s la j cu ultimul arc ij , arc invers și $r(P) = e_3$.

Inițial, se etichetează sursa s cu eticheta $(0, ., \infty)$. Celelalte vîrfuri primesc etichetă prin "cercetarea" vîrfurilor deja etichetate:

Dacă i este un vîrf etichetat, atunci $\forall j \in V$

Dacă j neetichetat, $ij \in E$ și $x_{ij} < c_{ij}$ atunci

j se etichet. $e = (i, \text{direct}, \min(e_3[i], c_{ij} - x_{ij}))$;

Dacă j neetichetat, $ji \in E$ și $x_{ji} > 0$ atunci

j se etichet. $e = (i, \text{invers}, \min(e_3[i], x_{ji}))$.

Evident, în acest fel se respectă semnificația celor trei componente ale etichetelor.

Numim această procedură *etichetare*(i).

Atunci cînd în procedura de etichetare s-a atribuit etichetă vîrfului t , s-a obținut un drum de creștere P a fluxului curent, de capacitate reziduală $r(P) = e_3[t]$ și ale cărei vîrfuri se depistează în $O(n)$ explorînd prima componentă a etichetelor. Modificarea fluxului $x \otimes r(P)$ se execută în acest mers înapoi, tot în $O(n)$.

Pentru noul flux se reia procedura de etichetare.

Dacă toate vîrfurile etichetate au fost cercetate și nu s-a reușit etichetarea vîrfului t , rezultă că fluxul curent nu admite drumuri de creștere, este deci de valoare maximă, iar dacă $S =$ mulțimea vîrfurilor etichetate atunci $(S, V - S)$ este o secțiune de capacitate minimă.

Descrierea algoritmului

- 1: Se alege $x = (x_{ij})$ flux inițial (de ex. fluxul nul);
Se etichetează s cu $(0, ., \infty)$
 - 2: **while** $(\exists$ vîrfuri etichetate necercetate) **do**
 { "alege" un vîrf etichetat și necercetat i ;
 etichetare(i);
 if (t a primit etichetă) **then**
 { modifică fluxul pe drumul dat de etichete;
 șterge toate etichetele;
 etichetează s cu $(0, ., \infty)$
 }
 }
 }
 - 3: $S \leftarrow \{i | i \in V, i \text{ are etichetă}\}$
 $T \leftarrow V - S$
- x este flux de valoare maximă
 (S, T) este secțiune de capacitate minimă.

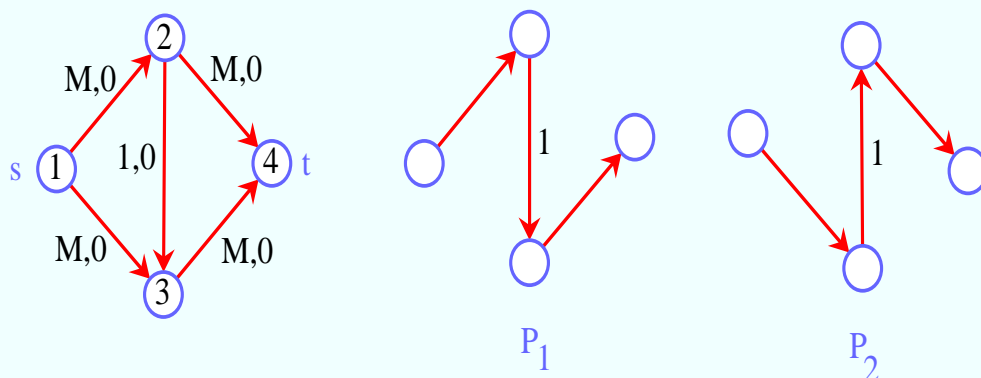
Complexitatea algoritmului:

Pentru fiecare creștere a fluxului, sînt necesare cel mult $2m$ ($m = |E|$) inspecții de arce în vederea etichetării.

Dacă toate capacitățile sînt întregi atunci vor fi necesare cel mult v ($v =$ valoarea fluxului maxim) creșteri succesive. Rezultă că algoritmul are complexitatea $O(mv)$.

Dacă U este o margine superioară a capacităților arcelor atunci $v \leq (n - 1)U$ ($(n - 1)U$ este o margine superioară a capacității secțiunii $(\{s\}, V - \{s\})$), deci algoritmul are complexitatea $O(nmU)$.

Observații. 1⁰. Dezavantajele algoritmului sînt legate de neconvergența în cazul capacităților iraționale (deși practic, în implementări nu este cazul), și de faptul că mărimile capacităților influențează comportarea sa, acestea neconstituind o măsură a volumului datelor de intrare. Exemplu:



Dacă "alegerea", din pasul 2 al algoritmului, face ca drumurile de creștere succesive (pornind de la fluxul nul) să fie $P_1, P_2, P_1, P_2, \dots$ unde $P_1 = 1, 2, 3, 4$, $P_2 = 1, 3, 2, 4$ atunci, fiecare creștere a fluxului mărește cu 1 valoarea fluxului curent și, deci, vor fi necesare $2M$ creșteri, ceea ce este inadmisibil pentru $M \in \mathbb{Z}_+$ foarte mari.

2^0 . Aceste dezavantaje pot fi evitate dacă alegerile vîrfurilor etichetate supuse cercetării se fac judicios. Primii care au observat acest fenomen, au fost Dinic(1970) și independent, Edmonds și Karp (1972).

Modificarea lui Edmonds și Karp a algoritmului lui Ford & Fulkerson

Numim **drum minim de creștere a fluxului** x în rețeaua R , un drum de creștere de **lungime minimă** printre toate drumurile de creștere.

Fie x un flux oarecare în rețeaua R . Definim șirul de fluxuri x^k în R astfel:

$$\begin{aligned} x^0 &\leftarrow x; \\ x^k &\leftarrow x^{k-1} \otimes r(P_{k-1}), \quad P_k \text{ este drum minim de creștere} \\ &\text{relativ la } x^{k-1}; \quad k = 1, 2, \dots \end{aligned}$$

Vom dovedi că șirul de fluxuri astfel construit este finit.

Notăm, pentru $\forall i \in V$ și $\forall k = 0, 1, 2, \dots$

$\sigma_i^k =$ lungimea minimă a unui C-drum de la s la i în R relativ la fluxul x^k .

$\tau_i^k =$ lungimea minimă a unui C-drum de la i la t în R relativ la fluxul x^k .

Lemă. 4. Pentru $\forall i \in V$ și $\forall k = 0, 1, 2, \dots$ avem

$$\sigma_i^{k+1} \geq \sigma_i^k \text{ și } \tau_i^{k+1} \geq \tau_i^k.$$

Teoremă. 4. (Edmonds, Karp)

Dacă $x = x^0$ este un flux oarecare în rețeaua R , atunci șirul de fluxuri x^1, x^2, \dots obținut din x^0 prin creșteri succesive pe drumuri minime de creștere, are cel mult $\frac{mn}{2}$ elemente (în cel mult $\frac{mn}{2}$ creșteri succesive, se obține un flux care nu admite drumuri de creștere).

Demonstrație: Dacă P este un drum de creștere relativ la un flux în rețeaua R , cu capacitatea reziduală $r(P)$, vom numi **arc critic** în P orice arc $e \in P$ cu $r(e) = r(P)$.

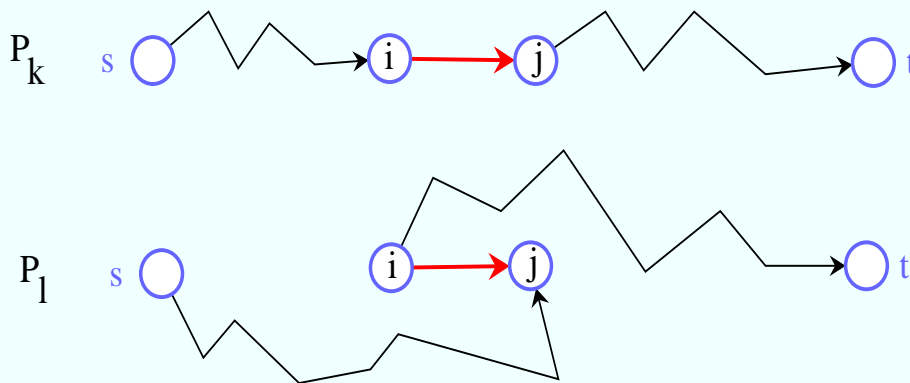
În $x \otimes r(P)$, fluxul pe arcele critice devine sau egal cu capacitatea (pentru arcele directe) sau egal cu 0 (pentru arcele inverse).

Fie ij un arc critic pe drumul minim de creștere P_k relativ la x^k . Lungimea lui P_k este:

$$\sigma_i^k + \tau_i^k = \sigma_j^k + \tau_j^k.$$

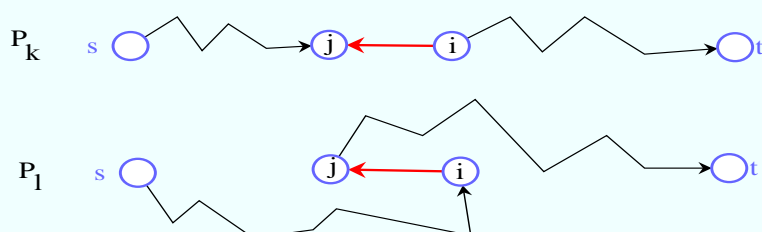
Cum ij este critic în P_k , în x^{k+1} nu va putea fi folosit în aceeași direcție ca în P_k . Prima oară când fluxul pe arcul ij se va modifica, el va apare într-un drum de creștere P_l cu $l > k$ relativ la x^l și va fi folosit în direcție opusă. Avem, deci, două cazuri:

i) **ij direct în P_k** . Atunci $\sigma_j^k = \sigma_i^k + 1$. În P_l ij va fi arc invers, deci $\sigma_i^l = \sigma_j^l + 1$.



Rezultă, $\sigma_i^l + \tau_i^l = \sigma_j^l + 1 + \tau_i^l \geq \sigma_j^k + 1 + \tau_i^k = \sigma_i^k + \tau_i^k + 2$ (s-a folosit lema 1). Am obținut că $\lg(P_l) \geq \lg(P_k) + 2$.

ii) ij **arc invers** în P_k . Atunci $\sigma_i^k = \sigma_j^k + 1$. În P_l ij va fi arc direct, deci $\sigma_j^l = \sigma_i^l + 1$.



Rezultă $\sigma_j^l + \tau_j^l = \sigma_i^l + 1 + \tau_j^l \geq \sigma_i^k + 1 + \tau_j^k = \sigma_j^k + \tau_j^k + 2$ Deci, $\lg(P_l) \geq \lg(P_k) + 2$.

Deci **orice drum minim de creștere în care arcul ij este critic este cu măcar două arce mai lung decât precedentul în care ij a fost critic.**

Cum, un drum în G are cel mult $n - 1$ arce, rezultă că un arc fixat nu poate fi critic în procesul de creștere mai mult de $\frac{n}{2}$ ori.

Cum orice drum de creștere are cel puțin un arc critic, rezultă că nu putem avea mai mult de $\frac{mn}{2}$ drumuri minime de creștere, în șirul construit. Deci, după cel mult $mn/2$ creșteri, se obține un flux care nu admite drumuri de creștere.

Corolar. *Dacă $R = (G, s, t, c)$ este o rețea, atunci există un flux care nu admite drumuri de creștere.*

Observații: 1⁰ Rezultă de aici, că demonstrația teoremei 3 este completă.

2⁰ S-ar putea pune întrebarea dacă nu cumva, alegerea drumurilor minime de creștere, mărește complexitatea algoritmului de flux maxim ?
Răspunsul este însă banal:

Lemă. 5. *Dacă, în pasul 2 al algoritmului lui Ford și Fulkerson, alegerea vîrfurilor etichetate în vederea cercetării se face după regula "primul etichetat - primul cercetat", atunci drumurile de creștere care se depistează sînt cu număr minim de arce.*

Demonstrație: Fie P un drum de creștere depistat și fie P' un drum minim de creștere. Presupunem că $\lg(P) > \lg(P')$.

Fie

$P : si_1, i_1i_2, \dots, i_{k-1}i_k, i_ki_{k+1}, i_{k+1}i_{k+2}, \dots, i_{k+l-1}t$ și

$P' : si_1, i_1i_2, \dots, i_{k-1}i_k, i_kj_{k+1}, j_{k+1}j_{k+2}, \dots, j_{k+l'-1}t.$

Deci, $lg(P) = k + l$, $lg(P') = k + l'$, au primele k arce comune și $l' < l$.

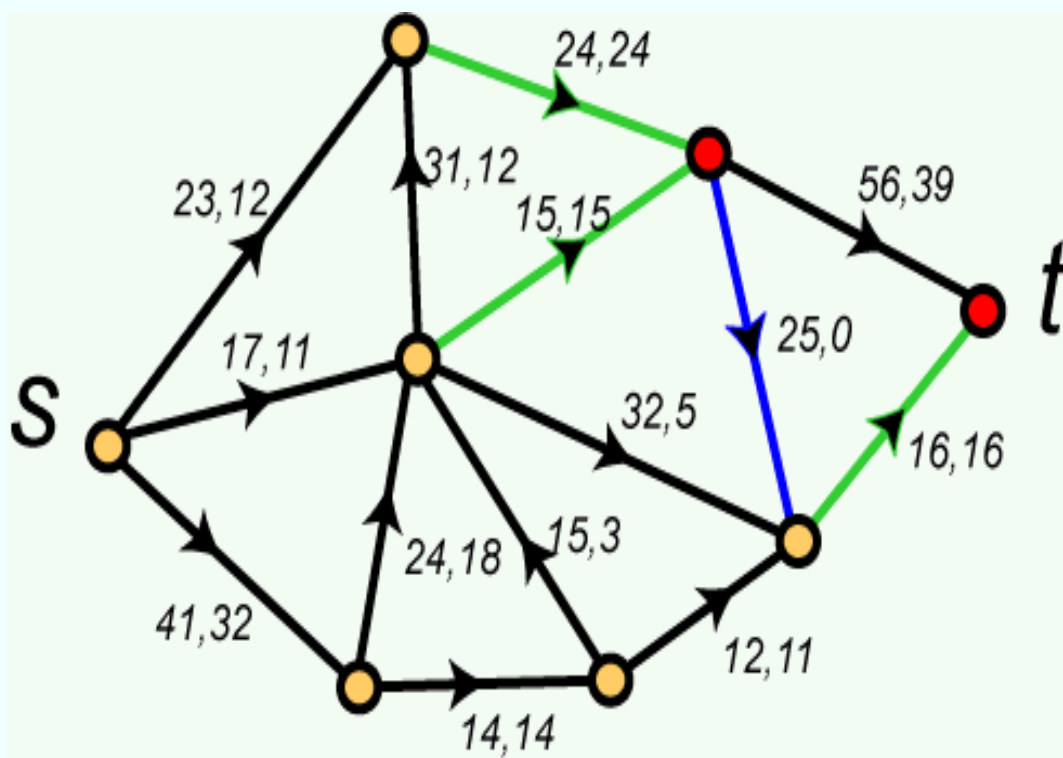
Conform regulii de etichetare, i_{k+1} va primi etichetă înaintea lui j_{k+1} , dar j_{k+1} va primi etichetă înaintea lui i_{k+2} ; j_{k+2} va primi etichetă înaintea lui i_{k+3} și așa mai departe; obținem inductiv că t primește etichetă înaintea lui $i_{k+l'+1}$, deci t primește etichetă pe drumul P' , înainte de a primi etichetă pe drumul P , absurd.

Observație: Regula "primul etichetat - primul cercetat" corespunde unei explorări *bfs* a vârurilor etichetate, ceea ce se poate realiza, utilizând o coadă pentru memorarea lor. Aceasta nu afectează complexitatea algoritmului, care va necesita tot $O(m)$ operații pentru fiecare creștere a fluxului, și din teorema 4 obținem

Teoremă. 5. (Edmonds- Karp 1972)

Dacă se modifică algoritmul lui Ford și Fulker-son cu precizarea alegerii bfs a vîrfurilor etichetate în vederea cercetării, atunci, fluxul maxim se obține în timpul $O(m^2n)$.

Exercițiu. Aplicați algoritmul lui Ford & Fulker-son modificat pentru obținerea unui flux de valoare maximă în rețeaua de mai jos (în care este precizat un flux inițial de valoare 55).



Algoritmi de tip preflux.

Fie $R = (G, s, t, c)$ o rețea.

Definiție. Se numește **preflux** în rețeaua R , o funcție $x : E \rightarrow \mathbf{R}$ astfel încât

$$(i) \quad 0 \leq x_{ij} \leq c_{ij} \quad \forall ij \in E$$

$$(ii) \quad \forall i \neq s \quad e_i = \sum_{j:ji \in E} x_{ji} - \sum_{j:ij \in E} x_{ij} \geq 0.$$

Numărul $e_i \quad i \in V - \{s, t\}$ se numește **excesul** din vârful i .

Dacă $i \in V - \{s, t\}$ și $e_i > 0$ atunci i se numește **nod activ**.

Dacă $ij \in E$ x_{ij} va fi numit **fluxul pe arcul** ij .

Observații: 1^0 . Dacă în rețeaua R nu există noduri active, atunci **prefluxul** x **este flux** de la s la t în R de valoare e_t .

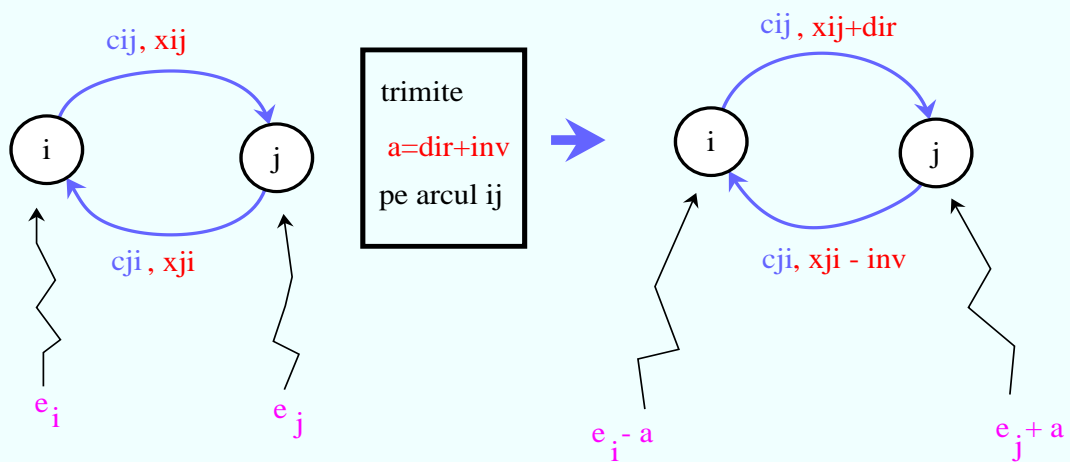
2^0 . *Ideea algoritmilor de tip preflux este: se pornește cu un preflux în R și se transformă prin modificări ale fluxului pe arce într-un flux care nu admite drumuri de creștere.*

3^0 . În definiția unui preflux, nu am mai utilizat convenția că vom introduce toate perechile de arce din digraful complet simetric de ordin n , întrucât în analiza algoritmilor pe care îi vom prezenta va fi esențială reprezentarea digrafului G cu ajutorul listelor de adiacență. **Totuși, vom considera că dacă $ij \in E$ atunci și $ji \in E$ (altminteri, adăugăm arcul ji cu capacitate 0).**

Definiție: Dacă x este un preflux în R și $ij \in E$, atunci **capacitatea reziduală** a arcului ij este

$$r_{ij} = c_{ij} - x_{ij} + x_{ji}$$

(reprezentând fluxul adițional ce poate fi "*trimis*" de la nodul i la nodul j utilizând arcele ij și ji).



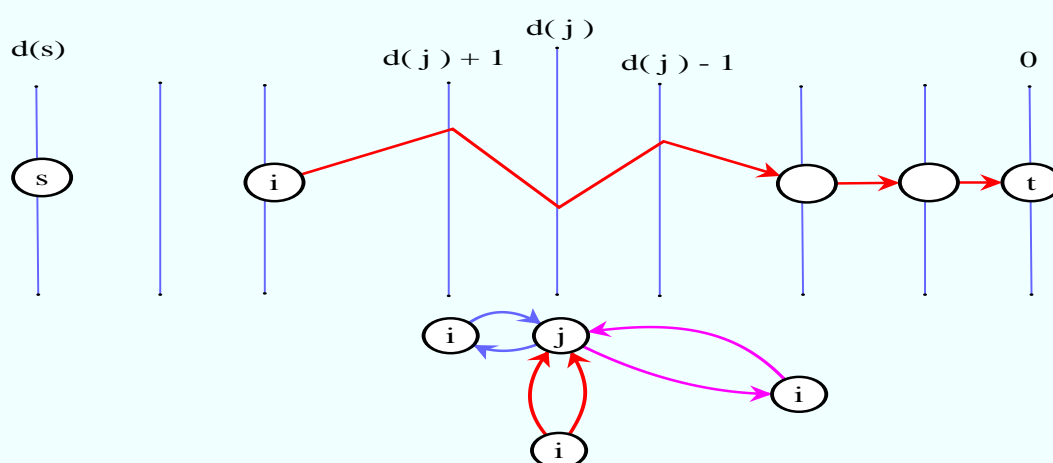
Observație. Peste tot, în cele ce urmează, a "*trimite*" flux de la i la j înseamnă să creștem fluxul pe arcul ij sau să micșorăm fluxul pe arcul ji .

Definiție: Se numește **C-drum** în R relativ la prefluxul x , un drum al lui G ale cărui arce au capacitatea reziduală pozitivă.

Definiție: Se numește *funcție de distanță* în R relativ la prefluxul x , o funcție $d : V \rightarrow \mathbf{Z}_+$ care satisface

$$(D1) \quad d(t) = 0$$

$$(D2) \quad \forall ij \in E, r_{ij} > 0 \Rightarrow d(i) \leq d(j) + 1$$



Observații: 1^0 . Dacă P este un C-drum relativ la prefluxul x în R de la i la t atunci $d(i) \leq \lg(P)$ (arcele unui C-drum au capacitate reziduală pozitivă și se aplică (D2)).

Rezultă că $d(i) \leq \tau_i$ (lungimea minimă a unui C-drum de la i la t).

2⁰. Vom nota cu $A(i)$, pentru orice vîrf i , lista sa de adiacență, care conține arcele $ij \in E$.

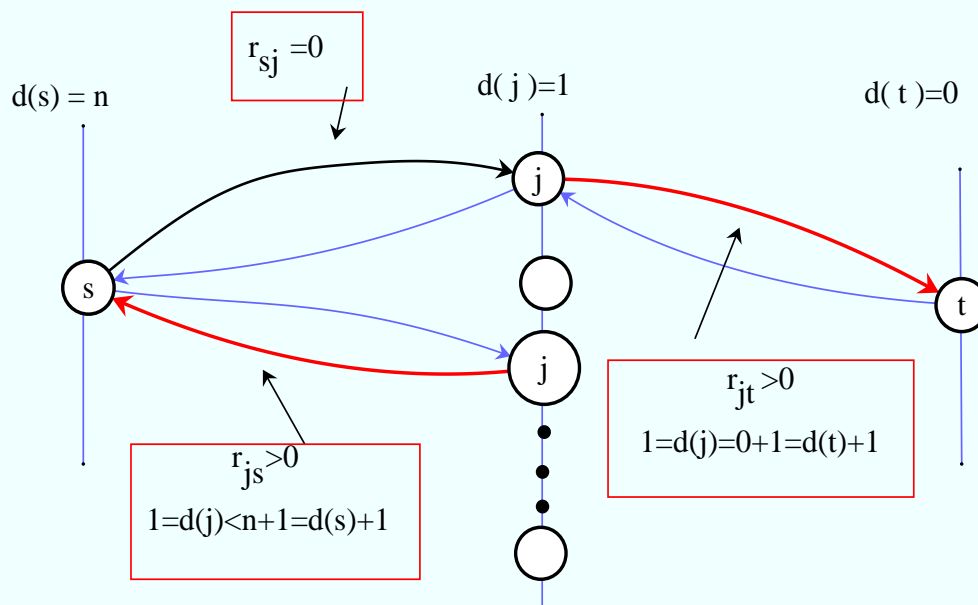
Definiție. Fie x un preflux în R și d o funcție de distanță relativ la x . Un arc $ij \in E$ se numește **admisibil** dacă

$$r_{ij} > 0 \quad \wedge \quad d(i) = d(j) + 1.$$

Dacă R este o rețea, considerăm *inițializare* următoarea procedură care construiește în $O(m)$ un preflux x și o funcție de distanță d corespunzătoare acestuia, astfel:

```
procedure inițializare;  
{  
    for  $\forall ij \in E$  do  
        if  $i = s$  then  $x_{sj} \leftarrow c_{sj}$  else  $x_{ij} \leftarrow 0$ ;  
     $d[s] \leftarrow n$ ;  $d[t] \leftarrow 0$ ;  
    for  $\forall i \in V - \{s, t\}$  do  $d[i] \leftarrow 1$   
}
```

Observații: 1^0 . După execuția acestei proceduri, $\forall sj \in A(s)$ avem $r_{sj} = 0$, deci alegerea lui $d(s) = n$ nu afectează condiția D2. Pentru arcele cu capacitate reziduală pozitivă de forma js D2 este evident verificată.



2^0 . Alegerea lui $d(s) = n$ are următoarea interpretare: "nu există C-drum de la s la t în R relativ la x " (întrucât, altminteri, ar trebui ca lungimea acestuia să fie $\geq n$).

Dacă, în algoritmi de tip preflux vom păstra acest invariant, atunci când x va deveni flux, va rezulta că nu admite drumuri de creștere și deci x va fi de valoare maximă.

Considerăm următoarele proceduri

procedure *pompează* (i);

// i este un vârf diferit de s, t

{

alege $ij \in A(i)$ ij **admisibil**;

"trimite" $\delta = \min(e_i, r_{ij})$ unități de flux de la i la j

}

Dacă $\delta = r_{ij}$ avem o **pompare saturată**, altminteri pomparea este **nesaturată**.

procedure *reetichetare* (i);

// i este un vârf diferit de s, t

{

$d(i) \leftarrow \min\{d(j) + 1 \mid ij \in A(i) \wedge r_{ij} > 0\}$

}

Schema generală a unui algoritm de tip preflux este:

```
{
    inițializare;
    while  $\exists$  noduri active în  $R$  do
    {
        selectează un nod activ  $i$ ;
        if  $\exists$  arce admisibile în  $A(i)$ 
            then pompează( $i$ )
            else reetichetare( $i$ )
    }
}
```

Lemă. 6. *Algoritmul de tip preflux, de mai sus, are ca invariant " d este funcție de distanță relativ la prefluxul x ". La fiecare apel al lui reetichetare(i), $d(i)$ crește strict.*

Demonstrație: Procedura inițializare construiește, evident, o funcție de distanță relativ la prefluxul inițial.

Dacă înaintea execuției unei iterații a lui *while*, d e funcție de distanță relativ la prefluxul curent x , atunci:

- a)** dacă se execută *pompare*(i), singura pereche ce poate viola D2 este $d(i)$ și $d(j)$. Cum arcul ij a fost ales admisibil, avem $d(i) = d(j) + 1$. După *pompare*, arcul ji poate avea $r_{ji} > 0$ (fără ca înainte să fi fost), dar condiția $d(j) \leq d(i) + 1$ este, evident, satisfăcută;
- b)** dacă se execută *reeticetare*(i), modificarea lui $d(i)$ se face astfel încât D2 să rămână valabilă pentru orice arc ij cu $r_{ij} > 0$. Cum apelul lui *reeticetare* implică $d(i) < d(j) + 1 \quad \forall ij$ cu $r_{ij} > 0$, rezultă că după apel $d(i)$ crește măcar cu o unitate.

Pentru finitudinea algoritmului va trebui să ne asigurăm că, dacă, în timpul execuției, avem un nod i activ, atunci în $A(i)$ există măcar un arc ij cu $r_{ij} > 0$. Aceasta rezultă din

Lemă. 7. *Dacă pe parcursul algoritmului, i_0 este un nod activ, atunci există un C-drum de la i_0 la s , în R , relativ la prefluxul curent x .*

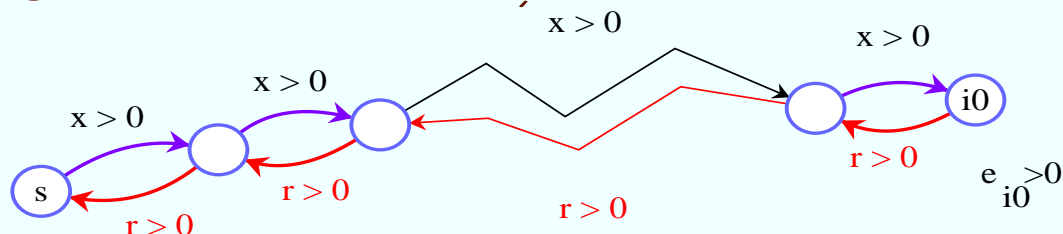
Demonstrație: Dacă x este un preflux în R , atunci x se poate scrie ca o sumă finită $x = x^1 + x^2 + \dots + x^p$, unde fiecare x^k satisface: mulțimea $A^k = \{ij \mid ij \in E, x_{ij}^k \neq 0\}$ este

- a) mulțimea arcelor unui drum de la s la t ,
- b) m. arcelor unui dr. de la s la un nod activ,
- c) mulțimea arcelor unui circuit.

În plus, în situațiile a) și c), x^k este flux.

(demonstrația rezultă algoritmic, construind mai întâi mulțimile a), apoi a celor de tip c) și b); la fiecare etapă se caută inversul unui drum de tipul a), b), sau c); prefluxul construit se scade din cel curent; excesele nenegative, permit efectuarea construcției; ea este finită, căci după fiecare etapă nr. arcelor cu flux curent nul crește.)

Cum i_0 este un nod activ în R relativ la x , rezultă că situația b) va apare pentru nodul i_0 (întrucât situațiile b) și c) nu afectează excesul din nodul i_0). Arcele inverse ale acestui drum au capacitatea reziduală strict pozitivă și ele formează C-drumul din enunțul lemei.



Corolar. 1. $\forall i \in V \quad d(i) < 2n.$

Demonstrație: Dacă i nu a fost reetichetat, atunci $d(i) = 1 < 2n$.

Dacă i a fost reetichetat, atunci înainte de reetichetare i este nod activ, deci există C-drum de la i la s de lungime cel mult $n - 1$.

Din modul de modificare a lui $d(i)$ și din D2 rezultă că după reetichetare $d(i) \leq d(s) + n - 1 = 2n - 1$, întrucât $d(s) = n$ nu se schimbă pe parcursul algoritmului.

Corolar. 2. *Numărul total de apeluri ale procedurii reetichetare este mai mic decât $2n^2$.*

Demonstrație: Fiecare din cele $n - 2$ vîrfuri ce pot fi supuse etichetării poate fi etichetat de cel mult $2n - 1$ ori, avînd în vedere corolarul 1, lema 6 și etichetarea inițială.

Corolar. 3. *Numărul total de pompări saturate este $\leq nm$.*

Demonstrație: După ce un arc ij devine saturat (situație în care $d(i) = d(j) + 1$), pe acest arc nu se va mai putea trimite flux pînă cînd nu se va trimite flux pe arcul ji situație în care vom avea $d'(j) = d'(i) + 1 \geq d(i) + 1 = d(j) + 2$; această schimbare de flux nu va avea loc pînă ce $d(j)$ nu crește cu două unități. Deci, un arc nu poate deveni saturat mai mult de n ori și în total vom avea cel mult mn pompări saturate.

Lemă. 8. *(Goldberg și Tarjan 1986) Numărul pompărilor nesaturate este cel mult $2n^2m$.*

Lemă. 9. *La terminarea algoritmului x este flux de valoare maximă.*

Demonstrație: Din lemele 6 și 8 și corolarul 3 rezultă că algoritmul se termină după cel mult $2n^2m$ iterații și cum $d(s) = n$ nu se modifică pe parcurs, rezultă că fluxul obținut este fără drumuri de creștere.

Vom prezenta în continuare, algoritmul lui **Ahuja și Orlin** (1988) care, utilizând o metodă de **scalare**, va mărgini numărul pompărilor nesaturate de la $O(n^2m)$ la $O(n^2 \log U)$.

Vom presupune că toate capacitățile sînt întregi și că $\max_{ij \in E}(1 + c_{ij}) = U$. Notăm $\lceil \log_2 U \rceil = K$. *Ideia algoritmului:*

Se vor executa $K + 1$ etape. Pentru fiecare etapă p , cu p luînd succesiv valorile $K, K - 1, \dots, 1, 0$ vor fi îndeplinite următoarele condiții:

- (a)** - la începutul etapei p , $\forall i$ satisface $e_i \leq 2^p$
- (b)** - în timpul etapei p se utilizează procedurile *pompare-etichetare* în vederea eliminării nodurilor active cu $e_i \in (2^{p-1}, 2^p]$.

Din alegerea lui K , în etapa inițială ($p = K$) condiția (a) este satisfăcută și deci, dacă (b) va fi invariant al algoritmului, după $K+1$ etape, excesele nodurilor vor fi $\leq \frac{1}{2}$.

Dacă, toate transformările datelor vor păstra integritatea exceselor, va rezulta că excesul oricărui nod este 0, și, deci, dispunem de un flux de valoare maximă (datorită proprietăților funcției distanță $d(i)$).

Realizarea lui (b) se poate face astfel:

- se începe etapa p , construind lista $L(p)$ a tuturor nodurilor $i_1, i_2, \dots, i_{l(p)}$ cu excesele $e_{i_j} > 2^{p-1}$, ordonate crescător după d (avînd în vedere că d poate lua valori între 1 și $2n - 1$, o sortare de tip hash rezolvă problema în $O(n)$).
- nodul activ selectat pentru *pompare-reetichetare* va fi pe tot parcursul etapei, **primul nod din** $L(p)$. Va rezulta că, dacă se face o pompare pe arcul ij admisibil, cum $d(j) = d(i) - 1$ și i este primul din $L(p)$, vom avea $e_i > 2^{p-1}$ și $e_j \leq 2^{p-1}$. Dacă, în plus, se va limita δ , fluxul "trimis" de la i la j , în procedura de *pompare* la $\delta = \min(e_i, r_{ij}, 2^p - e_j)$,

atunci, cum $2^p - e_j \geq 2^{p-1}$ va rezulta că orice **pompare nesaturată** trimite cel puțin 2^{p-1} unități de flux. După pompare, excesul din nodul j (singurul nod al cărui exces poate crește) va fi $e_j + \min(e_i, r_{ij}, 2^p - e_j) \leq e_j + 2^p - e_j \leq 2^p$ (deci b) rămîne îndeplinită).

- etapa (p) se termină atunci cînd $L(p)$ devine vidă.

Pentru realizarea eficientă a operațiilor de depistare a unui arc pe care se face pomparea, sau a examinării arcelor care ies dintr-un vîrf i pentru reetichetare, vom considera listele $A(i)$ organizate astfel:

- fiecare nod al listei conține: vîrf j ; x_{ij} ; r_{ij} ; pointer către arcul ji (din lista de adiacență a vîrfului j); pointer către următorul element din lista $A(i)$.
- parcurgerea listei se face cu ajutorul unui pointer $p(i)$ către elementul curent din listă.

Evident, organizarea acestor liste se face înaintea apelului lui *inițializare* și necesită $O(m)$ operații.

Algoritmul Ahuja-Orlin

```
inițializare;  
 $K \leftarrow \lceil \log_2(U) \rceil$  ;  $\Delta \leftarrow 2^{K+1}$ ;  
for  $p = K, K - 1, \dots, 0$  do  
{  construiește  $L(p)$ ;  $\Delta \leftarrow \frac{\Delta}{2}$   
  while  $L(p) \neq \emptyset$  do  
  {  fie  $i$  primul element din  $L(p)$ ;  
    parcurge lista  $A(i)$  din locul curent pînă  
    se determină un arc admisibil sau se  
    depistează sfîrșitul ei;  
    if  $ij$  este arcul admisibil găsit then  
    {   $\delta \leftarrow \min(e_i, r_{ij}, \Delta - e_j)$ ;  
       $e_i \leftarrow e_i - \delta$ ;  $e_j \leftarrow e_j + \delta$ ;  
      "trimite"  $\delta$  unități de flux de la  $i$  la  $j$ ;  
      if  $e_i \leq \frac{\Delta}{2}$  then șterge  $i$  din  $L(p)$ ;  
      if  $e_j > \frac{\Delta}{2}$  then adaugă  $j$  ca prim nod în  $L(p)$   
    }  
  else // s-a depistat sfîrșitul listei  
  {  șterge  $i$  din  $L(p)$ ;  
    parcurge toată lista  $A(i)$  pentru calculul lui  
     $d(i) = \min\{d(j) + 1; ij \in A(i) \wedge r_{ij} > 0\}$ ;  
    introdu  $i$  în  $L(p)$  la locul său (hash);  
    pune pointerul curent al listei  $A(i)$  la început  
  }  
}
```

Operațiile care domină complexitatea timp, în cazul cel mai nefavorabil, sînt pompările nesaturate. Toate celelalte operații sînt dominate de $O(nm)$.

Lemă. 8'. *Numărul pompărilor nesaturate este cel mult $8n^2$ în fiecare etapă a scalării, deci $O(n^2 \log U)$ în total.*

Demonstrație: Fie

$$F(p) = \sum_{i \in V, i \neq s, t} \frac{e_i \cdot d(i)}{2^p}.$$

La începutul etapei p , $F(p) < \sum_{i \in V} \frac{2^p \cdot (2n)}{2^p} = 2n^2$.

Dacă în etapa p , atunci cînd se analizează nodul activ i , se apelează *reetichetare*, înseamnă că nu există arce ij admisibile.

Operația de reetichetare mărește $d(i)$ cu $\epsilon \geq 1$ unități, ceea ce conduce la o creștere a lui F cu cel mult ϵ unități. Cum pentru fiecare i , creșterea lui $d(i)$ pe parcursul întregului algoritm este $< 2n$ rezultă că F va crește pînă la cel mult valoarea $4n^2$, la sfîrșitul etapei p .

Dacă, pentru un nod i se execută *pompare*, atunci aceasta se execută pe arcul ij cu $r_{ij} > 0$ și $d(i) = d(j) + 1$.

Cum numărul unităților de flux pompat este $\delta \geq 2^{p-1}$, după pompare $F(p)$ va avea valoarea $F'(p)$, unde

$$F'(p) = F(p) - \frac{\delta \cdot d(i)}{2^p} + \frac{\delta \cdot d(j)}{2^p} = F(p) - \frac{\delta}{2^p} \leq F(p) - \frac{2^{p-1}}{2^p} = F(p) - \frac{1}{2}.$$

Rezultă că această situație nu poate apărea mai mult de $8n^2$ (întrucât $F(p)$ poate crește cel mult la $4n^2$).

Cu atât mai mult, pompările nesaturate nu vor depăși $8n^2$.

Sumarizând toate rezultatele anterioare obținem

Teoremă. 6. (*Ahuja-Orlin 1988*) Algoritmul de tip preflux cu scalarea exceselor are complexitatea $O(nm + n^2 \log U)$.

2. Aplicații (combinatorii) ale problemei fluxului maxim.

A. Determinarea cuplajului maxim și a stăbilei maxime într-un graf bipartit.

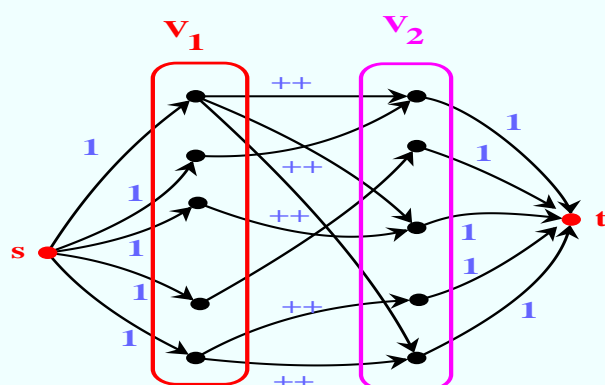
Fie $G = (V_1, V_2; E)$ un graf bipartit cu n vîrfuri și m muchii.

Considerăm rețeaua $R = (G_1, s, t, c)$, unde

- $V(G_1) = \{s, t\} \cup V_1 \cup V_2$;
- $E(G_1) = E_1 \cup E_2 \cup E_3$
 - . $E_1 = \{sv_1 \mid v_1 \in V_1\}$,
 - . $E_2 = \{v_2t \mid v_2 \in V_2\}$,
 - . $E_3 = \{v_1v_2 \mid v_1 \in V_1, v_2 \in V_2, v_1v_2 \in E\}$;
- $c : E(G_1) \rightarrow \mathbf{Z}_+$ definită prin

$$c(e) = \begin{cases} 1 & \text{dacă } e \in E_1 \cup E_2 \\ \infty & \text{dacă } e \in E_3 \end{cases}$$

(vezi figura următoare).



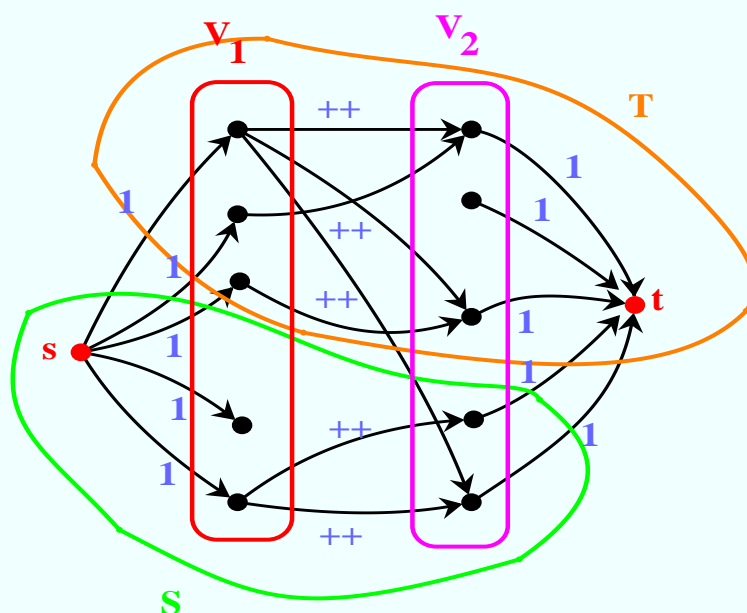
Dacă $x = (x_{ij})$ este un flux cu componente întregi în R atunci se observă că mulțimea de arce $\{ij \mid i \in V_1, j \in V_2 \wedge x_{ij} = 1\}$ induce în graful G bipartit un cuplaj $M(x)$. În plus, $v(x)$ este cardinalul cuplajului $M(x)$.

Reciproc, orice cuplaj din G induce o mulțime de arce neadiacente în G_1 ; dacă pe fiecare astfel de arc ij ($i \in V_1, j \in V_2$) se consideră fluxul x_{ij} egal cu 1 și de asemenea $x_{si} = x_{jt} = 1$, și luînd fluxul $x = 0$ pe orice alt arc, atunci fluxul construit are valoarea $|M|$.

Rezultă că rezolvînd problema fluxului maxim pe rețeaua R se determină (pornind de la fluxul nul) în $O(nm + n^2 \log n)$ un cuplaj de cardinal maxim în graful bipartit G .

Fie (S, T) secțiunea de capacitate minimă ce se poate construi, din fluxul maxim obținut, în $O(m)$ operații.

Vom avea, $c(S, T) = \nu(G)$, din teorema fluxului maxim-secțiunii minime.



Cum $\nu(G) < \infty$, rezultă că punând $S_i = S \cap V_i$ și $T_i = T \cap V_i$ ($i = 1, 2$), avem: $|T_1| + |S_2| = \nu(G)$, iar $X = S_1 \cup T_2$ este **mulțime stabilă** în graful G (pentru a avea $c(S, T) < \infty$).

În plus, $|X| = |V_1 - T_1| + |V_2 - S_2| = n - \nu(G)$. Rezultă că X este stabilă de cardinal maxim, întrucât $n - \nu(G) = \alpha(G)$ (teorema lui König).

B. Recunoașterea secvențelor digrafice.

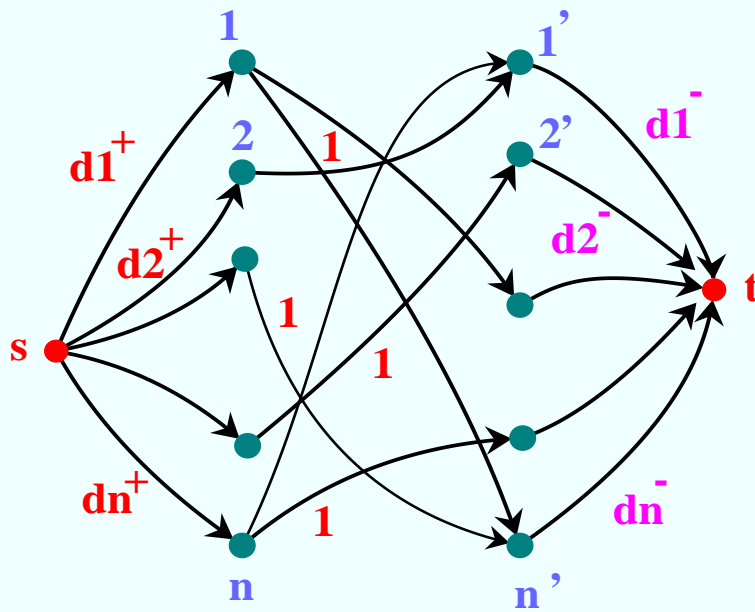
Date $(d_i^+)_{i=1,n}$ și $(d_i^-)_{i=1,n}$, **există un digraf** G **cu** n **vîrfuri astfel încît** $G = (\{1, \dots, n\}, E)$ **și** $d_G^+(i) = d_i^+$ **și** $d_G^-(i) = d_i^- \quad \forall i = 1, n$? (Problema poate apare în proiectarea circuitelor integrate).

Evident, va trebui ca $0 \leq d_i^+ \leq n-1$ și $0 \leq d_i^- \leq n-1 \quad \forall i = 1, n$ și $\sum_{i=1,n} d_i^+ = \sum_{i=1,n} d_i^- = m$ (unde, $m = |E|$, iar $d_i^+, d_i^- \in \mathbf{Z}$).

În aceste ipoteze, construim rețeaua bipartită $R = (G_1, s, t, E)$ unde G_1 se obține din $K_{n,n} - \{11', 22', \dots, nn'\}$ prin orientarea muchiilor ij' $\forall i \neq j \in \{1, \dots, n\}$, și introducerea arcelor si , $i \in \{1, \dots, n\}$ și $j't$, $j \in \{1, \dots, n\}$.

Funcția de capacitate va fi $c(si) = d_i^+ \quad \forall i = 1, n$, $c(j't) = d_j^- \quad \forall j = 1, n$ și $c(ij') = 1$.

Dacă în această rețea, există un flux întreg de valoare maximă m , atunci din orice vîrf $i \in \{1, \dots, n\}$ vor pleca exact d_i^+ arce pe care fluxul este 1, de forma ij' și în fiecare vîrf j' vor intra exact d_j^- arce pe care fluxul este 1, de forma ij' .



Considerînd mulțimea de vîrfuri $\{1, \dots, n\}$ și introducînd toate arcele ij astfel încît $x_{ij'} = 1$, se obține un digraf G cu secvențele gradelor interioare și exterioare $(d_i^+)_{i=1,n}$ și $(d_i^-)_{i=1,n}$. Reciproc, dacă acest digraf există, atunci procedînd invers ca în construcția anterioară, se obține un flux întreg în rețeaua R de valoare m (deci maxim). Rezultă că recunoașterea secvențelor digrafice (și construcția digrafului în cazul răspunsului afirmativ) se poate face în $O(nm + n^2 \log n) = O(n^3)$.

C. Determinarea nr. de muchie-conexiune al unui graf

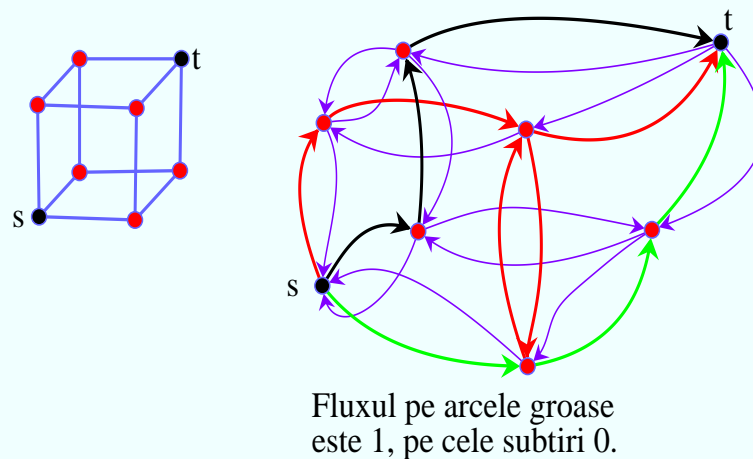
Fie $G = (V, E)$ un graf. Pentru $s, t \in V, s \neq t$, definim:

- $p_e(s, t)$ = numărul maxim de drumuri cu muchii disjuncte ce unesc s și t în G .
- $c_e(s, t)$ = cardinalul minim al unei mulțimi de muchii, prin îndepărtarea căreia din graf, între s și t nu mai există drumuri.

Teoremă. 7. $p_e(s, t) = c_e(s, t)$.

Demonstrație: Construim din G digraful G_1 , înlocuind fiecare muchie a lui G cu o pereche de arce simetrice. Considerăm $c : E(G_1) \rightarrow \mathbb{Z}_+$ prin $c(e) = 1, \forall e \in E(G_1)$.

Fie x^0 un flux întreg de valoare maximă în $R = (G_1, s, t, c)$. Fluxul x^0 se poate scrie ca o sumă de $v(x^0)$ fluxuri x^k întregi de valoare 1, înlocuind, eventual, fluxul pe unele circuite cu 0.



Fiecare astfel de flux x^k induce un drum de la s la t în G_1 (considerînd arcele pe care fluxul este nenul), și deci, și în G .

Rezultă că $v(x^0) = p_e(s, t)$, întrucît orice mulțime de drumuri disjuncte pe muchii, generează un flux de la s la t în R de valoare egală cu numărul acestor drumuri.

Fie (S, T) o secțiune de capacitate minimă; avem $c(S, T) = v(x^0)$, din teorema fluxului maxim-secțiunii minime.

Pe de altă parte, $c(S, T)$ este numărul arcelor cu o extremitate în S și cealaltă în T , deoarece $c(e) = 1, \forall e \in E(G_1)$. Această mulțime de arce generează în G o mulțime de muchii de același cardinal și cu proprietatea că deconectează, prin îndepărtare, s și t .

Rezultă că avem $c(S, T) = v(x^0) = p_e(s, t)$ muchii în G care deconectează, prin îndepărtare, s și t . Deci $c_e(s, t) \leq p_e(s, t)$.

Cum, inegalitatea $c_e(s, t) \geq p_e(s, t)$ este evidentă, rezultă că teorema este demonstrată.

Corolar. Dacă G este un graf conex $\lambda(G)$ (valoarea maximă a lui $p \in \mathbb{Z}_+$ astfel încît G este p -muchie-conex) este

$$\min_{\substack{s, t \in V(G) \\ s \neq t}} c_e(s, t). \quad (*)$$

Rezultă că, pentru a afla $\lambda(G)$, rezolvăm cele $\frac{n(n-1)}{2}$ probleme de flux, descrise în demonstrația teoremei.

Totuși, să observăm că pentru o pereche fixată s și t avem: dacă (S, T) este secțiunea de capacitate minimă, atunci

$$\forall v \in S \text{ și } \forall w \in T \quad c_e(v, w) \leq c(S, T) \quad (**).$$

În particular, dacă (s, t) este perechea pentru care se realizează minimul în (*) vom avea egalitate în (**).

Rezultă că dacă fixăm un vîrf s_0 și rezolvăm $n - 1$ probleme de flux cu $t \in V - s_0$ se va obține în mod necesar o pereche s_0, t_0 pentru care $c(s_0, t_0) = \lambda(G)$ (ne asigurăm, astfel, depistarea unui vîrf t_0 , care să nu fie în aceeași clasă cu s_0 în partiția (S, T)).

Rezultă că în $O(n \cdot (nm + n^2c)) = O(n^2m)$ **se pot determina $\lambda(G)$ și o mulțime separatoare de muchii de cardinal minim în G .**

D. Determinarea numărului de conexiune al unui graf.

Dacă $G = (V, E)$ este un graf și

$s, t \in V, \quad s \neq t, \quad st \notin E$ atunci, notînd

- $p(s, t)$ = numărul maxim de st -drumuri cu mulțimile de vîrfuri disjuncte (cu excepția extremităților),

- $c(s, t)$ = cardinalul minim al unei mulțimi de vîrfuri st -separatoare,

avem, din teorema lui Menger,

$$p(s, t) = c(s, t) \quad (*)$$

În plus, numărul de conexiune $k(G)$ al grafului G (valoarea maximă a lui $p \in \mathbf{Z}_+$ pentru care G este p -conex) este

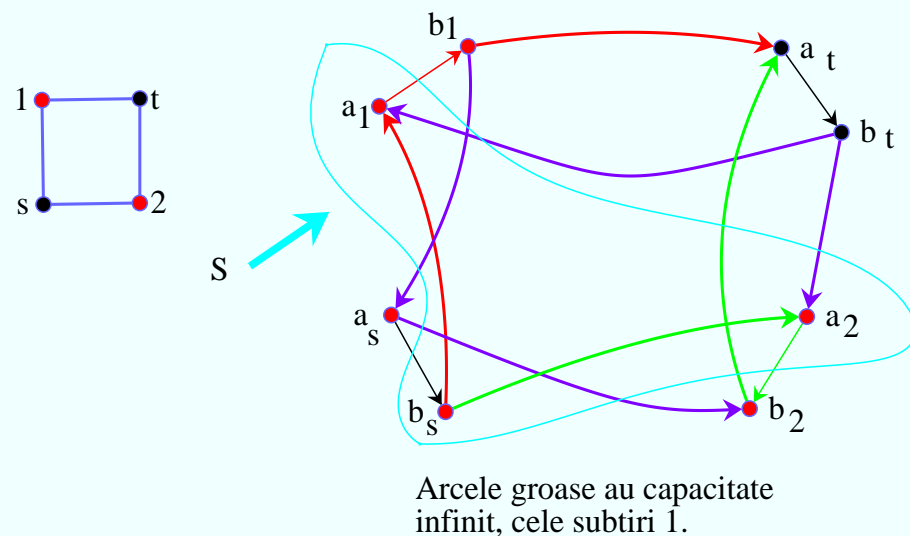
$$k(G) = \begin{cases} n - 1 & \text{dacă } G = K_n \\ \min_{\substack{s, t \in V \\ st \notin E}} c(s, t) & \text{dacă } G \neq K_n \end{cases} \quad (**)$$

Vom arăta că egalitatea (*) rezultă și din teorema fluxului maxim-secțiunii minime pe o rețea convenabil aleasă.

Fie $G_1 = (V(G_1), E(G_1))$ digraful construit din G astfel:

- $\forall v \in V$ considerăm $a_v, b_v \in V(G_1)$ și $a_v b_v \in E(G_1)$;
- $\forall vw \in E$ considerăm $b_v a_w, b_w a_v \in E(G_1)$.

Exemplu:



Definim $c : E(G_1) \rightarrow \mathbb{Z}_+$ prin

$$c(e) = \begin{cases} 1 & \text{dacă } e = a_v b_v \\ \infty & \text{altfel.} \end{cases}$$

Considerăm rețeaua $R = (G_1, b_s, a_t, c)$.

Fie x^0 un flux întreg de la b_s la a_t în R de valoare maximă.

În vîrfurile $b_v (v \in V)$ intră exact un arc de capacitate 1 și din vîrfurile $a_v (v \in V)$ pleacă exact un arc de capacitate 1. Rezultă că, pentru ca să fie satisfăcută legea de conservare a fluxului, $x_{ij}^0 \in \{0, 1\} \quad \forall ij \in E(G_1)$.

Aceasta înseamnă că x^0 se poate descompune în $v(x^0)$ fluxuri x^k , fiecare de valoare 1, și astfel încît arcele pe care x^k sînt nenule corespund la $v(x^0)$ drumuri disjuncte de la b_s la a_t în G_1 , care induc o mulțime de $v(x^0)$ drumuri intern disjuncte de la s la t în G .

Cum, pe de altă parte, dintr-o mulțime de p drumuri intern disjuncte de la s la t în G , se pot construi p drumuri intern disjuncte de la b_s la a_t în G_1 pe care se poate transporta cîte o unitate de flux, rezultă că avem

$$: \quad v(x^0) = p(s, t).$$

Fie (S, T) o secțiune în R astfel încât $v(x^0) = c(S, T)$.

Cum $v(x^0)$ este finit, rezultă că $\forall i \in S, \forall j \in T$ cu $ij \in E(G_1)$, avem $c(ij) < \infty$, deci $c(ij) = 1$, adică $\exists u \in V$ astfel încât $i = a_u$ și $j = b_u$.

Deci secțiunii (S, T) îi corespunde o mulțime de vîrfuri $A_0 \subseteq V$ astfel încât, $c(S, T) = |A_0|$ și evident A_0 este st -separatoare.

Cum, pe de altă parte, $\forall A$ st -separatoare, $|A| \geq p(s, t) = v(x_0)$ rezultă că
 $c(s, t) = |A_0| = c(S, T) = v(x_0) = p(s, t)$.

Demonstrația de mai sus, arată că pentru a determina $k(G)$ va fi suficient să determinăm minimul din $(**)$ prin rezolvarea a $|E(\overline{G})|$ probleme de flux, unde \overline{G} este graful complementar al lui G .

Deci algoritmul va avea complexitatea

$$O\left(\left(\frac{n(n-1)}{2} - m\right)(nm + n^2 \log n)\right).$$

O simplă observație ne conduce la un algoritm mai eficient. Evident,

$$k(G) \leq \min_{v \in V} d_G(v) = \frac{1}{n}(n \cdot \min_{v \in V} d_G(v)) \leq \frac{1}{n}(\sum_{v \in V} d_G(v)) = \frac{2m}{n}.$$

Dacă A_0 este o mulțime de articulație în G cu $|A_0| = k(G)$ atunci $G - A_0$ este neconex și se poate partiționa $V - A_0 = V' \cup V''$ astfel încât, $\forall v' \in V', \forall v'' \in V''$ avem $p(v', v'') = k(G)$.

Rezultă că, rezolvînd o problema de flux cu $s_0 \in V'$ și $t_0 \in V''$ va rezulta că $p(s_0, t_0) =$ valoarea fluxului maxim $= k(G)$.

Vom fi siguri că depistăm o astfel de pereche, dacă procedăm astfel:

considerăm $l = \lceil \frac{2m}{n} \rceil + 1$, alegem l vîrfuri oarecare din G , și pentru fiecare astfel de vîrf v rezolvăm toate problemele $p(v, w)$ cu $vw \notin E$.

Se vor rezolva în total $O(nl) = O(n(\frac{2m}{n} + 1)) = O(m)$ probleme. Deci complexitatea întregului algoritm va fi $O(m(nm + n^2 \log n))$.

3. Fluxuri de cost minim

Fie $R = (G, s, t, c)$ o rețea și x un flux de la s la t în R .

Considerăm $a : E \rightarrow \mathbf{R}$ o funcție de cost care asociază fiecărui arc $ij \in E$ $a(ij) = a_{ij}$ costul (transportului unei unități de flux) pe arcul ij .

Costul fluxului x se definește ca fiind

$$a(x) = \sum_{i,j} a_{ij}x_{ij}.$$

Problema fluxului de cost minim

Dată R o rețea, $v \in \mathbf{R}^+$ și $a : E \rightarrow \mathbf{R}$ funcție de cost, să se determine x^0 flux în R astfel încât

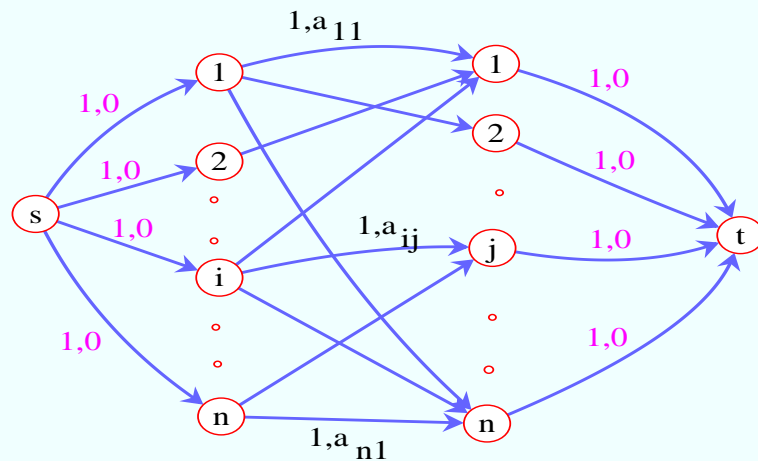
$$a(x^0) = \min\{a(x) \mid x \text{ flux în } R, v(x) = v\}.$$

Observăm că, dacă v nu depășește valoarea fluxului maxim în rețeaua R , atunci problema are întotdeauna soluții, $a(x)$ fiind liniară, iar mulțimea fluxurilor de valoare dată v fiind mărginită și închisă în \mathbf{R}^m .

Exemple.

1^0 . Se dispune de n lucrători și n lucrări. Costul atribuirii lucrătorului i la lucrarea j este a_{ij} ($i, j \in \{1, \dots, n\}$). **Să se atribue fiecare dintre cele n lucrări la câte un lucrător, astfel încât costul total al atribuirii să fie minim.** (*Problema simplă a atribuirii*).

Considerăm rețeaua descrisă mai jos, unde pe fiecare arc este trecută mai întâi capacitatea și apoi costul. Deci $c_{ij} = 1, c_{si} = 1, a_{si} = 0, c_{jt} = 1, a_{jt} = 0 \quad \forall i, j \in \{1, \dots, n\}$.



Evident, un flux întreg de valoare n și de cost minim, reprezintă soluția problemei.

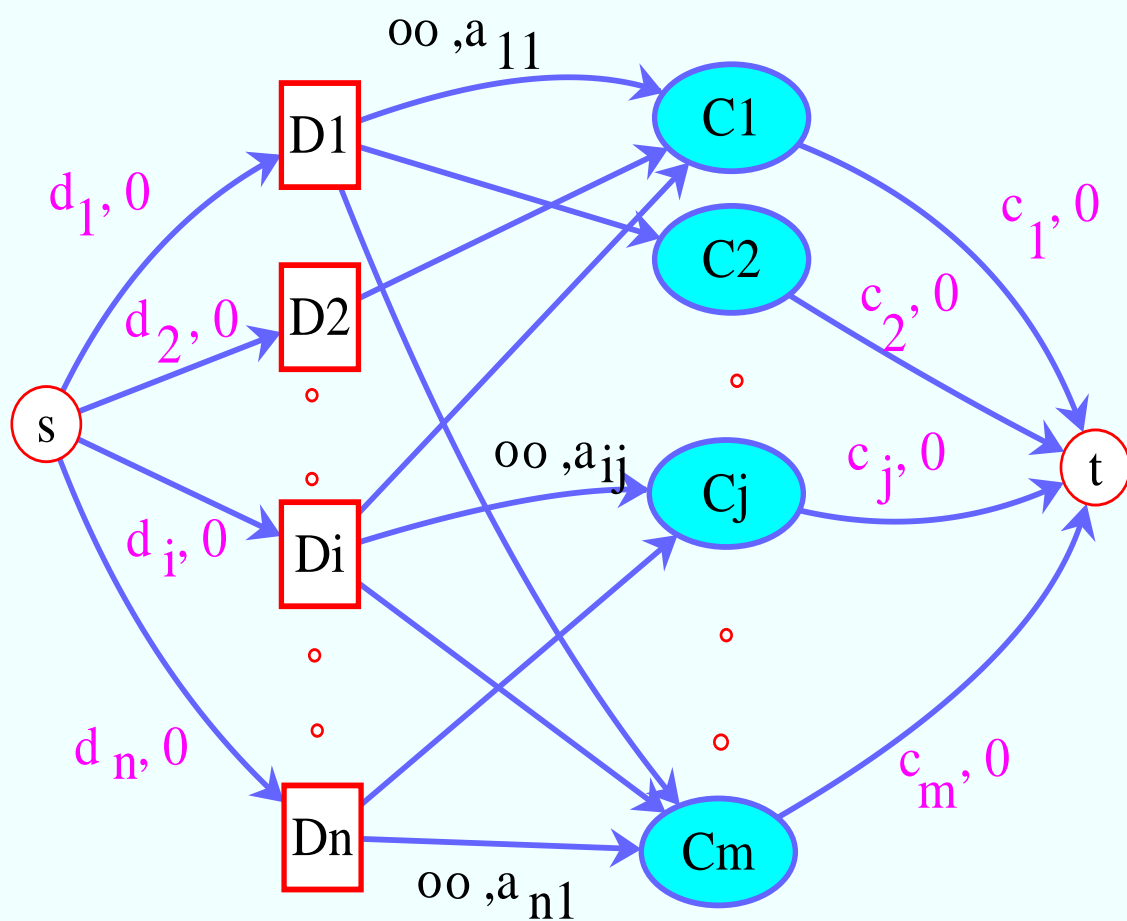
Notăm că în mod similar se poate rezolva orice problemă de cuplaj perfect de pondere minimă într-un graf bipartit.

2⁰. O marfă disponibilă în depozitele D_1, \dots, D_n în cantitățile d_1, \dots, d_n este solicitată în centrele de consum C_1, C_2, \dots, C_m în cantitățile c_1, c_2, \dots, c_m .

Se cunoaște costul a_{ij} al transportului unei unități de marfă de la depozitul D_i la centrul de consum C_j ($\forall i \in \{1, \dots, n\} \forall j \in \{1, \dots, m\}$). Se cere **să se stabilească un plan de transport care să satisfacă toate cererile și să aibă costul total minim** (*problema simplă a transporturilor Hitchcock-Koopmans*).

Evident, problema are soluție numai dacă $\sum_{i=1,n} d_i \geq \sum_{j=1,m} c_j$.

În acest caz, un flux de cost minim și de valoare $v = \sum_{i=1,m} c_i$ în rețeaua următoare, rezolvă problema.



Definiție. Fie x un flux în $R = (G, s, t, c)$ și $a : E \rightarrow \mathbf{R}$ o funcție de cost.

Dacă P este un C-drum în R relativ la fluxul x , atunci **costul drumului** P se definește

$$a(P) = \sum_{\substack{ij \in P \\ ij \text{ direct}}} a_{ij} - \sum_{\substack{ij \in P \\ ij \text{ invers}}} a_{ji}.$$

Dacă C este un C-drum închis, $a(C)$ se calculează după aceeași formulă, după stabilirea unui sens de parcurgere a lui C (este posibil ca ambele sensuri de parcurgere ale lui C să satisfacă definiția unui C-drum).

Observații: 1⁰ Din definiția dată, rezultă că dacă P este drum de creștere relativ la fluxul x , atunci $x^1 = x \otimes r(P)$ este un flux de valoare $v(x^1) = v(x) + r(P)$ și de cost $a(x^1) = a(x) + r(P) \cdot a(P)$.

2⁰ Dacă C este un C-drum închis relativ la x , atunci $x^1 = x \otimes r(C)$ este un flux de valoare $v(x^1) = v(x)$ și de cost $a(x^1) = a(x) + r(C) \cdot a(C)$.

Dacă $a(C) < 0$ atunci x^1 este un flux de aceeași valoare ca și x , dar de cost strict mai mic.

Teoremă. 8. *Un flux de valoare v este de cost minim dacă și numai dacă nu admite C-drumuri închise de cost negativ.*

Demonstrație: Necesitatea este evidentă din observația anterioară.

Suficiența. Fie x un flux de valoare v , care nu admite C-drumuri închise de cost negativ. Fie x^* un flux de valoare v , de cost minim (există !) astfel încât

$$\Delta(x, x^*) = \min\{\Delta(x, x') \mid x' \text{ flux de val. } v \text{ și cost minim}\}$$

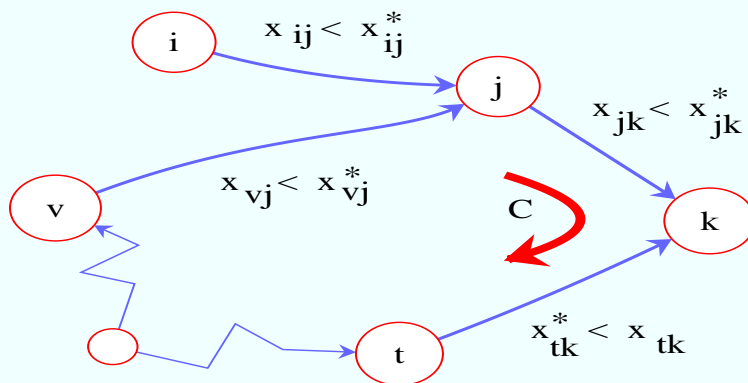
$$\text{unde } \Delta(x, x') = |\{ij \mid x_{ij} \neq x'_{ij}\}|.$$

Dacă $\Delta(x, x^*) = 0$ rezultă $x = x^*$ și deci x este de cost minim.

Dacă $\Delta(x, x^*) > 0$, fie ij astfel încât $x_{ij} \neq x^*_{ij}$. Presupunem $0 \leq x_{ij} < x^*_{ij} \leq c_{ij}$ (altfel, raționamentul este similar). Din legea de conservare a fluxurilor rezultă că

$$\exists jk \in E \text{ astfel încât } 0 \leq x_{jk} < x^*_{jk} \leq c_{jk}, \text{ sau}$$

$$\exists kj \in E \text{ astfel încât } 0 \leq x^*_{kj} < x_{kj} \leq c_{kj}.$$



Repetînd acest raționament, deoarece numărul vîrfurilor este finit, se va obține C , un C -drum închis relativ la x în R .

Considerînd sensul invers de parcurgere pe C se obține un C -drum C' , închis relativ la x^* .

Deoarece $a(C) \geq 0$ din ipoteză, iar $a(C') = -a(C)$, rezultă, din necesitatea teoremei (x^* este de cost minim), că $a(C) = 0$.

Modificînd fluxul x^* cu $\delta(C')$ pe C' , unde

$$\delta(C') = \min \left\{ \min_{kj \text{ direct în } C'} x_{kj} - x_{kj}^*, \min_{kj \text{ invers în } C'} x_{jk}^* - x_{jk} \right\}$$

se obține un flux x' cu $v(x') = v(x^*) = v$, $a(x') = a(x^*) + \delta(C') \cdot a(C') = a(x^*)$, deci de

cost minim, dar, cu $\Delta(x, x') < \Delta(x, x^*)$,
contradicție.

Deci $\Delta(x, x^*) = 0$, și demonstrația este încheiată.

Teoremă. 9. *Dacă x este un flux de valoare v și de cost minim iar P_0 este un drum de creștere, astfel încât*

*$a(P_0) = \min\{a(P) \mid P \text{ drum de creștere relativ la } x\}$,
atunci $x^1 = x \otimes r(P_0)$ este un flux de valoare
 $v(x^1) = v + r(P_0)$ și de cost minim.*

Linia demonstrației este următoarea :
presupunînd prin reducere la absurd că x^1 nu
este de cost minim, atunci x^1 admite un C-
drum închis C de cost negativ. Cum x era flux
de cost minim rezultă că $E(C) \cap E(P_0) \neq \emptyset$.

Dacă $ij \in E(C) \cap E(P_0)$, atunci va rezulta că
 $P_0 \cup C - ij$ conține un drum de creștere relativ
la x de cost mai mic decît P_0 .

Un drum de creștere de cost minim poate fi depistat cu ajutorul algoritmilor de drum minim. Dacă x este un flux în R și $a : E \rightarrow \mathbf{R}$ este funcția de cost atunci considerînd $a_{ij} = \infty$ dacă $ij \notin E$ (caz în care $x_{ij} = 0$), construim

$$\bar{a}_{ij} = \begin{cases} a_{ij} & \text{dacă } x_{ij} < c_{ij} \text{ și } x_{ji} = 0, \\ \min\{a_{ij}, -a_{ji}\} & \text{dacă } x_{ij} < c_{ij} \text{ și } x_{ji} > 0, \\ -a_{ji} & \text{dacă } x_{ij} = c_{ij} \text{ și } x_{ji} > 0, \\ +\infty & \text{dacă } x_{ij} = c_{ij} \text{ și } x_{ji} = 0. \end{cases}$$

Un drum de pondere minimă de la s la t în raport cu ponderile \bar{a}_{ij} corespunde unui drum minim de creștere în R relativ la fluxul x .

Un circuit de pondere negativă în raport cu ponderile \bar{a}_{ij} corespunde unui C-drum închis în R relativ la x , de cost negativ.

Rezultă, următorul algoritm pentru rezolvarea problemei fluxului de cost minim, obținut prin combinarea mai multor algoritmi clasici (*Klein, Busacker, Gowan, etc.*).

Algoritm generic de rezolvare a problemei fluxului de cost minim

```
{
0: Se consideră  $x = (x_{ij})$  un flux cu valoarea  $v' \leq v$ ;
   {  $x$  poate fi fluxul nul sau un flux  $y$  determinat
   cu ajutorul algoritmului de flux maxim și apoi
   considerînd  $x = (\frac{v}{v(y)}y)$  }
1: while  $(\exists$  circuite de pondere  $< 0$  relativ la  $\bar{a}_{ij})$  do
   { determină un astfel de circuit;
     modifică fluxul pe acest circuit
   }
2: while  $v(x) < v$  do
   { aplică un algoritm de drum minim în raport cu
     ponderile  $\bar{a}_{ij}$  pentru depistarea unui
     C-drum  $P$  de cost minim;
      $x \leftarrow x \otimes \min(r(P), v - v(x))$ 
   }
}
```

Complexitatea pasului 2 este $O(n^3v)$, dacă se pleacă de la fluxul nul. Se poate dovedi că pasul 1 se poate implementa astfel ca numărul iterațiilor să fie $O(nm^2 \log n)$.

VI. Reduceri polinomiale pentru probleme de decizie pe grafuri.

Definiție. Spunem că problema de decizie $P_1 : I_1 \rightarrow \{da, nu\}$ **se reduce polinomial** la problema de decizie $P_2 : I_2 \rightarrow \{da, nu\}$ și notăm aceasta prin $P_1 \propto P_2$, dacă există o funcție $\Phi : I_1 \rightarrow I_2$ polinomial calculabilă, astfel încât, $\forall i \in I_1 \quad P_1(i) = P_2(\Phi(i))$.

Funcția Φ se va da indicând un algoritm care construiește pentru orice instanță $i_1 \in I_1$, în timp polinomial în raport cu $|i_1|$, o instanță $i_2 \in I_2$ cu proprietatea că $P_1(i_1) = da$ dacă și numai dacă $P_2(i_2) = da$.

Se observă că relația de reducere polinomială \propto este o relație tranzitivă pe mulțimea problemelor de decizie (datorită închiderii mulțimii funcțiilor polinomiale la compunere).

Din punct de vedere algoritmic, construcția din spatele oricărei reduceri polinomiale este interesantă evidențiind modul în care prima problemă poate fi rezolvată eficient cu ajutorul unui oracol care rezolvă a doua problemă.

Vom considera cunoscut faptul că $SAT \propto 3SAT$ unde

SAT

Instantă: $U = \{u_1, \dots, u_n\}$ o mulțime finită de var. booleene.

$C = C_1 \wedge C_2 \wedge \dots \wedge C_m$ o formulă în formă

conjunctivă peste U :

$C_i = v_{i_1} \vee v_{i_2} \vee \dots \vee v_{i_{k_i}} \quad \forall i = \overline{1, m}$, unde

$\forall i_j \quad \exists \alpha \in \{1, \dots, n\}$ a. î. $v_{i_j} = u_\alpha$ sau $v_{i_j} = \overline{u_\alpha}$.

Intrebare: Există o atribuire $t : U \rightarrow \{A, F\}$ a. î. $t(C) = A$?

$3SAT$ este cazul particular al lui SAT în care fiecare clauză C_i are exact trei literali ($k_i = 3$), un literal v_{i_j} fiind, așa cum este descris mai sus, o variabilă sau negația ei.

Problema SAT este celebră datorită teoremei lui Cook (1971): SAT este NP -completă.

1. Mulțimi stabile

SM

Instanță: $G = (V, E)$ graf și $k \in \mathbb{N}$.

Intrebare: Există S mulțime stabilă în G a. î. $|S| \geq k$?

Teoremă. 1. (Karp 1972) $3SAT \propto SM$.

Demonstrație: Fie $U = \{u_1, u_2, \dots, u_n\}$,
($n \in \mathbb{N}^*$), $C = C_1 \wedge \dots \wedge C_m$ ($m \in \mathbb{N}^*$) cu
 $C_i = v_{i_1} \vee v_{i_2} \vee v_{i_3} \quad \forall i = 1, m$, (unde $\forall v_{i_j} \exists \alpha \in \{1, \dots, n\}$ astfel încît $v_{i_j} = u_\alpha$ sau $v_{i_j} = \bar{u}_\alpha$),
reprezentînd datele unei instanțe oarecare a
problemei 3SAT.

Vom construi în timp polinomial în raport cu
 $m + n$, un graf G și $k \in \mathbb{N}$ astfel încît ex-
istă o atribuire t a valorilor de adevăr sau fals
pentru variabilele booleene din U care să facă
adevărată formula C , dacă și numai dacă există
o stabilă S în graful G astfel încît $|S| \geq k$.

Graful G va fi construit astfel:

(1) Pentru orice $i \in \{1, \dots, n\}$ considerăm grafurile disjuncte $T_i = (\{u_i, \bar{u}_i\}, \{u_i \bar{u}_i\})$.

(2) Pentru orice $j = 1, m$ considerăm grafurile disjuncte

$$Z_j = (\{a_{j1}, a_{j2}, a_{j3}\}, \{a_{j1}a_{j2}, a_{j2}a_{j3}, a_{j3}a_{j1}\}) .$$

(3) Pentru orice $j = 1, m$ considerăm mulțimea de muchii

$$E_j = \{a_{j1}v_{j1}, a_{j2}v_{j2}, a_{j3}v_{j3}\} \text{ unde } v_{j1} \vee v_{j2} \vee v_{j3} \text{ este factorul } C_j.$$

$$\text{Considerăm } V(G) = \cup_{i=1}^n V(T_i) \cup \cup_{j=1}^m V(Z_j) \text{ și } E(G) = \cup_{i=1}^n E(T_i) \cup \cup_{j=1}^m (E(Z_j) \cup E_j).$$

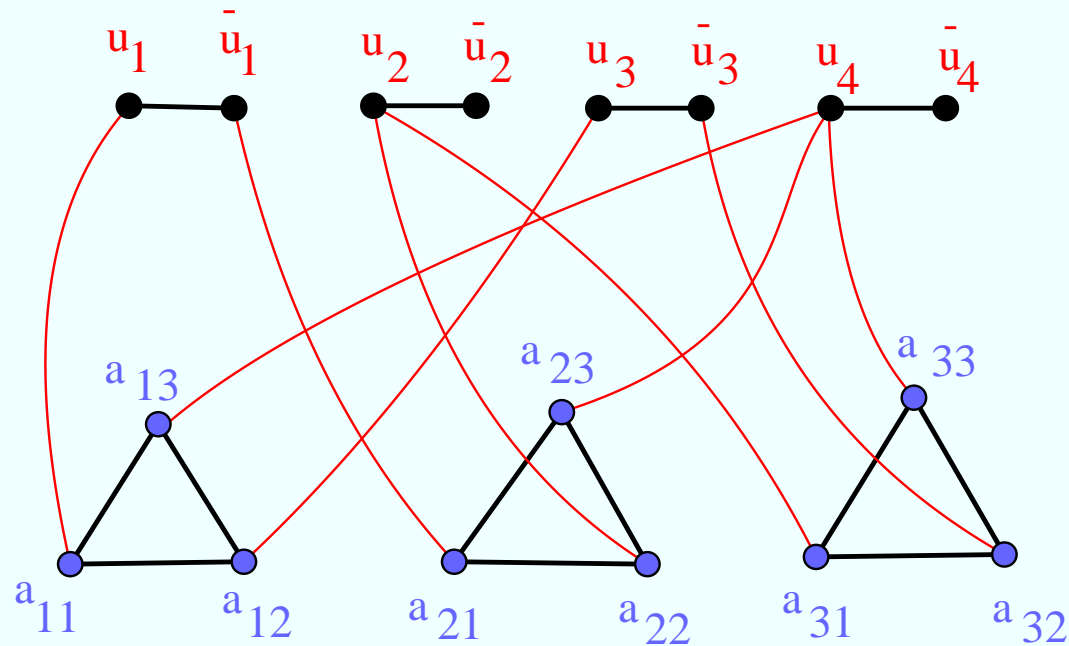
Evident, construcția este polinomială în raport cu $m + n$ (G are $2n + 3m$ vîrfuri și $n + 6m$ muchii).

Considerăm $k = n + m$.

Exemplu: $U = \{u_1, u_2, u_3, u_4\}$;

$C = (u_1 \vee u_3 \vee u_4) \wedge (\bar{u}_1 \vee u_2 \vee u_4) \wedge (u_2 \vee \bar{u}_3 \vee u_4)$;

$k = 4 + 3 = 7$.



Să presupunem că răspunsul la problema SM pentru instanța G și k astfel construite este da. Deci $\exists S \in \mathcal{S}_G$ astfel încât $|S| \geq k$. Cum orice mulțime stabilă poate avea cel mult un vîrf din orice $V(T_i)$ și din orice $V(Z_j)$ ($i = 1, n$, $j = 1, m$) rezultă că avem $|S| = k$ și deci $|S \cap V(T_i)| = 1$, $|S \cap V(Z_j)| = 1 \ \forall i = 1, n$ și $\forall j = 1, m$.

Considerăm $t : U \rightarrow \{A, F\}$ prin

$$t(u_i) = \begin{cases} A & \text{dacă } S \cap V(T_i) = \{\bar{u}_i\} \\ F & \text{dacă } S \cap V(T_i) = \{u_i\}. \end{cases}$$

Atunci, $\forall j = 1, m$ avem $t(C_j) = A$ (și deci $t(C) = A$).

În adevăr, $\forall j = 1, m$ dacă $C_j = v_{j1} \vee v_{j2} \vee v_{j3}$ și $S \cap V(Z_j) = a_{jk}$ ($k \in \{1, 2, 3\}$) atunci, deoarece $a_{jk}v_{jk} \in E$ rezultă că $v_{jk} \notin S$.

Dacă $v_{jk} = u_\alpha$, atunci $u_\alpha \notin S$ deci $\bar{u}_\alpha \in S$ și din definiția lui t avem $t(u_\alpha) = A$, adică $t(v_{jk}) = A$ ceea ce implică $t(C_j) = A$.

Dacă $v_{jk} = \bar{u}_\alpha$, atunci $\bar{u}_\alpha \notin S$ implică $u_\alpha \in S$, deci $t(\bar{u}_\alpha) = A$, adică $t(v_{jk}) = A$, ceea ce implică $t(C_j) = A$.

Reciproc, dacă răspunsul la problema $3SAT$ este da, atunci există o atribuire $t : U \rightarrow \{A, F\}$ astfel încît $t(C_j) = A \ \forall j = 1, m$.

Considerăm în graful G mulțimea stabilă \overline{S}_1 , $\overline{S}_1 = \cup_{i=1,n} V'_i$, unde

$$V'_i = \begin{cases} \{\overline{u}_i\} & \text{dacă } t(u_i) = A \\ \{u_i\} & \text{dacă } t(u_i) = F \end{cases}.$$

Atunci, $\forall j = 1, m$, cum $t(C_j) = A$, rezultă că există $k_j \in \{1, 2, 3\}$ astfel încît $t(v_{jk_j}) = A$. Considerăm $\overline{S}_2 = \cup_{j=1,m} \{a_{jk_j}\}$.

Evident, \overline{S}_2 este stabilă în G . Am construit $\overline{S}_1 \in \mathcal{S}_G$ cu $|\overline{S}_1| = n$, $\overline{S}_2 \in \mathcal{S}_G$ cu $|\overline{S}_2| = m$.

Considerăm $\overline{S} = \overline{S}_1 \cup \overline{S}_2$. Evident, $|\overline{S}| = n + m = k$ (deci $|\overline{S}| \geq k$) și în plus \overline{S} este mulțime stabilă în G (**deci răspunsul la SM pentru intrarea G, k este da**).

Faptul că \overline{S} este mulțime stabilă în G rezultă astfel: dacă $\exists v, w \in \overline{S}$ astfel încît $vw \in E(G)$ atunci o extremitate este din \overline{S}_1 și cealaltă din \overline{S}_2 .

Presupunînd $v \in \overline{S}_1$ avem două cazuri de considerat

a) $v = u_\alpha$, $w = a_{jk_j}$ $\alpha \in \{1, \dots, n\}$, $j \in \{1, \dots, m\}$, $k_j \in \{1, 2, 3\}$ și $v_{jk_j} = u_\alpha$. Cum $t(v_{jk_j}) = A$ rezultă $t(u_\alpha) = A$ deci $u_\alpha \notin \overline{S}_1$, contradicție.

b) $v = \overline{u}_\alpha$, $w = a_{jk_j}$ $\alpha \in \{1, \dots, n\}$, $j \in \{1, \dots, m\}$, $k_j \in \{1, 2, 3\}$ și $v_{jk_j} = \overline{u}_\alpha$. Cum $t(v_{jk_j}) = A$, rezultă $t(\overline{u}_\alpha) = A$ deci $t(u_\alpha) = F$ ceea ce implică $\overline{u}_\alpha \notin \overline{S}_1$, contradicție.

Cu aceasta teorema este complet demonstrată.

Să observăm că reducerea lui SAT la SM este complet similară, singura deosebire fiind că grafurile Z_i sunt grafuri complete cu k_i vârfuri.

2. Colorarea vârfurilor.

COL

Instanță: $G = (V, E)$ graf și $p \in \mathbb{N}^*$.

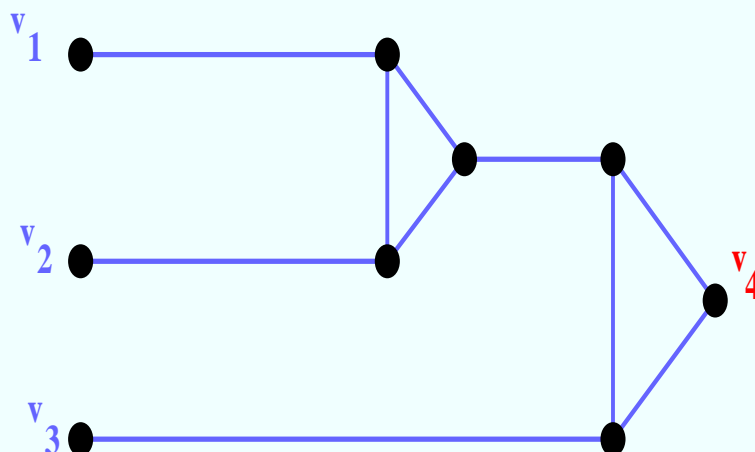
Intrebare: Există o p -colorare a vârfurilor lui G ?

Teoremă. 2. $3SAT \propto COL$.

Această teoremă evidențiază complexitatea problemelor de colorare a vârfurilor unui graf.

Vom demonstra chiar mai mult: fixînd $p = 3$ în enunțul lui COL, reducerea polinomială a lui 3SAT este încă posibilă !

Lemă. 1. *Fie H graful:*

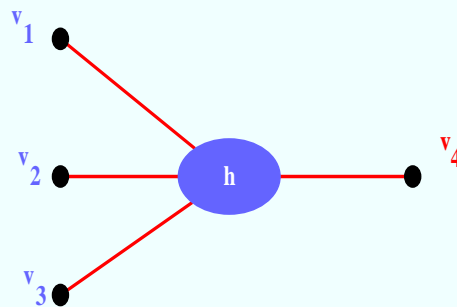


a) *Dacă c este o 3-colorare a lui H astfel încât $c(v_1) = c(v_2) = c(v_3) = a \in \{1, 2, 3\}$ atunci în mod necesar $c(v_4) = a$.*

b) *Dacă $c : \{v_1, v_2, v_3\} \rightarrow \{1, 2, 3\}$ satisface $c(\{v_1, v_2, v_3\}) \neq \{a\}$ atunci c poate fi extinsă la o 3-colorare c a lui H cu $c(v_4) \neq a$.*

Demonstrația lemei se poate face examinând lista 3-colorărilor lui H .

În cele ce urmează, vom desemna (pentru simplitate) graful H astfel:

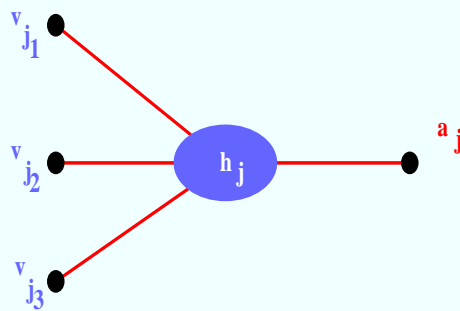


Demonstrația teoremei 2. Considerăm datele unei instanțe ale problemei 3SAT:

$U = \{u_1, \dots, u_n\}$ o mulțime de variabile booleene ($n \in \mathbb{N}^*$) și $C = C_1 \wedge C_2 \wedge \dots \wedge C_m$ ($m \in \mathbb{N}^*$) o formulă astfel încît $\forall i = 1, m$ $C_i = v_{i1} \vee v_{i2} \vee v_{i3}$, unde $\forall j = 1, 3 \exists \alpha$ astfel încît $v_{ij} = u_\alpha$ sau $v_{ij} = \bar{u}_\alpha$.

Vom construi un graf G , astfel încît, considerînd $p = 3$ în COL, vom obține că G este 3-colorabil dacă și numai dacă răspunsul la 3SAT este da, adică există $t : U \rightarrow \{A, F\}$, astfel încît $t(C) = A$. În plus, construcția lui G se va face în timp polinomial, parcurgînd următoarele etape:

1. $\forall i = 1, n$ considerăm grafurile disjuncte (V_i, E_i) unde $V_i = \{u_i, \bar{u}_i\}$ și $E_i = \{u_i \bar{u}_i\}$.
2. $\forall j = 1, m$, pentru $C_j = v_{j_1} \vee v_{j_2} \vee v_{j_3}$, considerăm grafurile:



unde v_{j_k} ($k = 1, 3$) sînt vîrfurile de la pasul 1, corespunzătoare literalilor v_{j_k} , grafurile h_j sînt disjuncte și a_j sînt vîrfuri distincte.

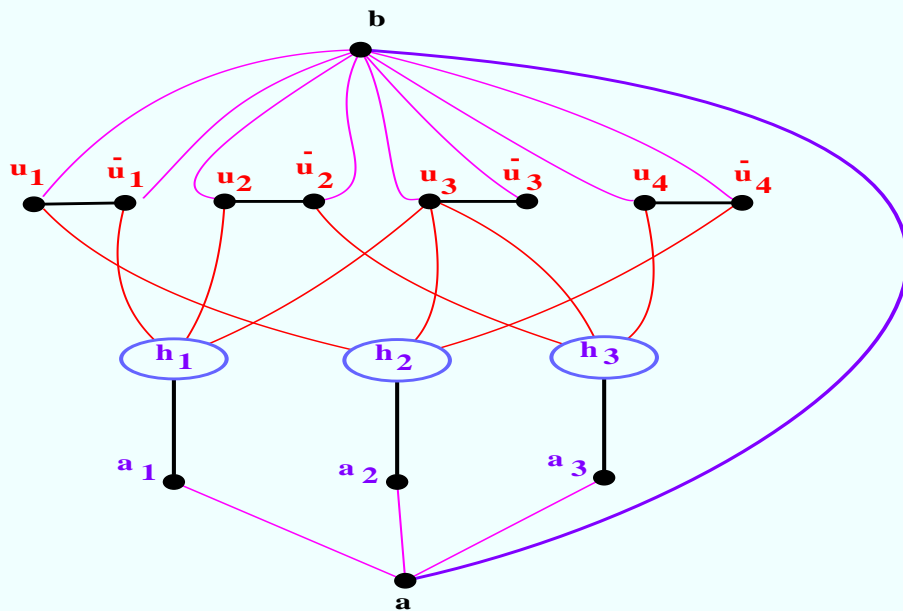
3. Considerăm a , un vîrf diferit de toate cele construite în pașii 1 și 2 și unim a cu vîrfurile a_j $j = 1, m$.
4. Considerăm b un vîrf nou, unit cu u_i și \bar{u}_i $\forall i = 1, n$ și de asemenea, cu vîrfurile a .

Graful G astfel construit are un număr liniar de vîrfuri în raport cu $n + m$.

Exemplu: $U = \{u_1, u_2, u_3, u_4\}$,

$$C = (\bar{u}_1 \vee u_2 \vee u_3) \wedge (u_1 \vee u_3 \vee \bar{u}_4) \wedge (\bar{u}_2 \vee u_3 \vee u_4)$$

Graful G va fi:



Presupunem ca răspunsul la 3SAT este da.

Deci $\exists t : U \rightarrow \{A, F\}$ astfel încît $t(C) = A$, deci $t(C_j) = A \ \forall j = 1, m$.

Construim o 3-colorare C a grafului G .

Definim mai întâi $c(u_i)$ și $c(\bar{u}_i) \forall i = 1, n$ astfel:

$c(u_i) = 1$ și $c(\bar{u}_i) = 2$, dacă $t(u_i) = A$ și

$c(u_i) = 2$ și $c(\bar{u}_i) = 1$, dacă $t(u_i) = F$.

Se observă că dacă v este un literal (u_α sau \bar{u}_α), atunci vârful v este colorat $c(v) = 2$ dacă și numai dacă $t(v) = F$.

Deci $\forall j = 1, m$ nu avem $c(v_{j_1}) = c(v_{j_2}) = c(v_{j_3}) = 2$.

Folosind lema 1,b) rezultă că putem extinde în fiecare graf h_j colorarea c astfel încît $c(a_j) \neq 2$, deci $c(a_j) \in \{1, 3\}$.

Rezultă că atribuind $c(a) = 2$ și $c(b) = 3$, c este o 3- colorare a lui G .

Reciproc, **presupunem că G este 3-colorabil.** Putem presupune (eventual renumerotînd culorile) că $c(b) = 3$ și $c(a) = 2$.

Va rezulta că $\{c(u_i), c(\bar{u}_i)\} = \{1, 2\}$ și $c(a_j) \in \{1, 3\} \ \forall i = 1, n, \ \forall j = 1, m$.

Din lema 1 a) rezultă că nu vom avea $c(v_{j1}) = c(v_{j2}) = c(v_{j3}) = 2 \ \forall j = 1, 3$.

Deci $\forall j = 1, m \ \exists v_{jk}$ astfel încît $c(v_{jk}) = 1$.

Definim $t : U \rightarrow \{A, F\}$ prin

$$t(u_i) = A \Leftrightarrow c(u_i) = 1.$$

Conform observației anterioare, vom avea că $t(C_j) = A \ \forall j = 1, m$, deci $t(C) = A$, adică **răspunsul la 3SAT este da.**

3. Colorarea muchiiilor

Considerăm următoarea problemă de decizie.

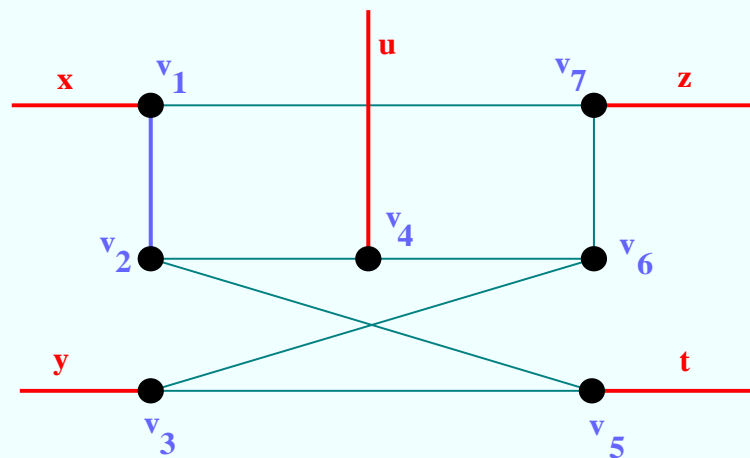
HOL

Instantă: $G = (V, E)$ graf 3-regulat.

Intrebare: Există o 3-colorare a muchiilor lui G ?

Teorema. 3. (Holyer 1981) $3SAT \propto HOL$.

Lemă. 2. *Fie H graful desenat mai jos.*



În orice 3-colorare c a muchiilor lui H , avem
 $c(x) = c(y)$, $|c(\{z, t, u\})| = 3$ sau
 $c(z) = c(t)$, $|c(\{x, y, u\})| = 3$.

Demonstrație Fie c o 3-colorare a lui H .

a) presupunem $c(x) = c(y)$. Eventual, după o renumerotare a culorilor vom avea $c(x) = c(y) = 1$ și $c(v_1v_7) = 2$, $c(v_1v_2) = 3$.

a1). $c(u) = 1$. Avem următorul șir de implicații:
 $c(v_2v_4) = 2, c(v_2v_5) = 1, c(v_4v_6) = 3, c(v_3v_6) = 2, c(v_3v_5) = 3, c(t) = 2, c(v_6v_7) = 1, c(z) = 3$.

a2). $c(u) = 2$. Avem următorul șir de implicații:
 $c(v_2v_4) = 1, c(v_2v_5) = 2, c(v_4v_6) = 3, c(v_3v_6) = 2, c(v_3v_5) = 3, c(t) = 1, c(v_6v_7) = 1, c(z) = 3$.

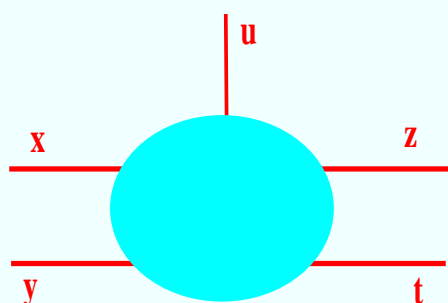
a3). $c(u) = 3$. Muchia v_2v_4 poate avea culorile 1 sau 2.

a3.1.) $c(v_2v_4) = 1$. Avem atunci, $c(v_2v_5) = 2, c(v_4v_6) = 2, c(v_3v_6) = 3$, și nu putem atribui $c(v_3v_5)$.

a3.2.) $c(v_2v_4) = 2$. Avem atunci, $c(v_2v_5) = 1, c(v_4v_6) = 1, c(v_6v_7) = 3, c(z) = 1, c(v_3v_6) = 2, c(v_3v_5) = 3, c(t) = 2$.

b) Argument similar, dacă $c(z) = c(t)$.

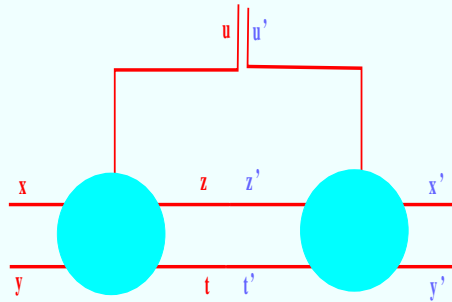
Să observăm că în graful H muchiile etichetate nu au precizată una din extremități. În construcția pe care o vom face, aceste extremități neprecizate vor fi identificate 2 câte 2; cu alte cuvinte, perechi de astfel de muchii vor fi identificate. Vom reprezenta simplificat graful H astfel:



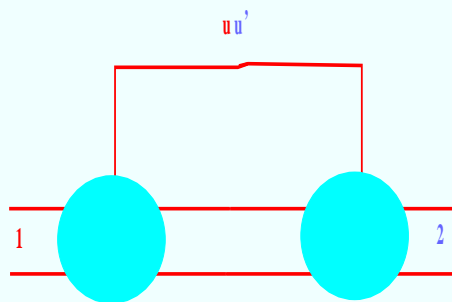
Dacă graful H este subgraf într-un graf care admite o 3- colorare a muchiilor, vom interpreta perechile de muchii (x, y) și (z, t) , ca reprezentând valoarea "adevăr", dacă sînt la fel colorate, respectiv, valoarea "fals", dacă sînt colorate diferit.

Lema 2 și această interpretare justifică denumirea de componentă inversoare pentru H (dacă (x, y) este "intrare" în H atunci (z, t) este "ieșire" și H transformă adevăr în fals și fals în adevăr).

Definim graful $2H$, considerînd două componente inversoare H cărora le identificăm o pereche de muchii etichetate.

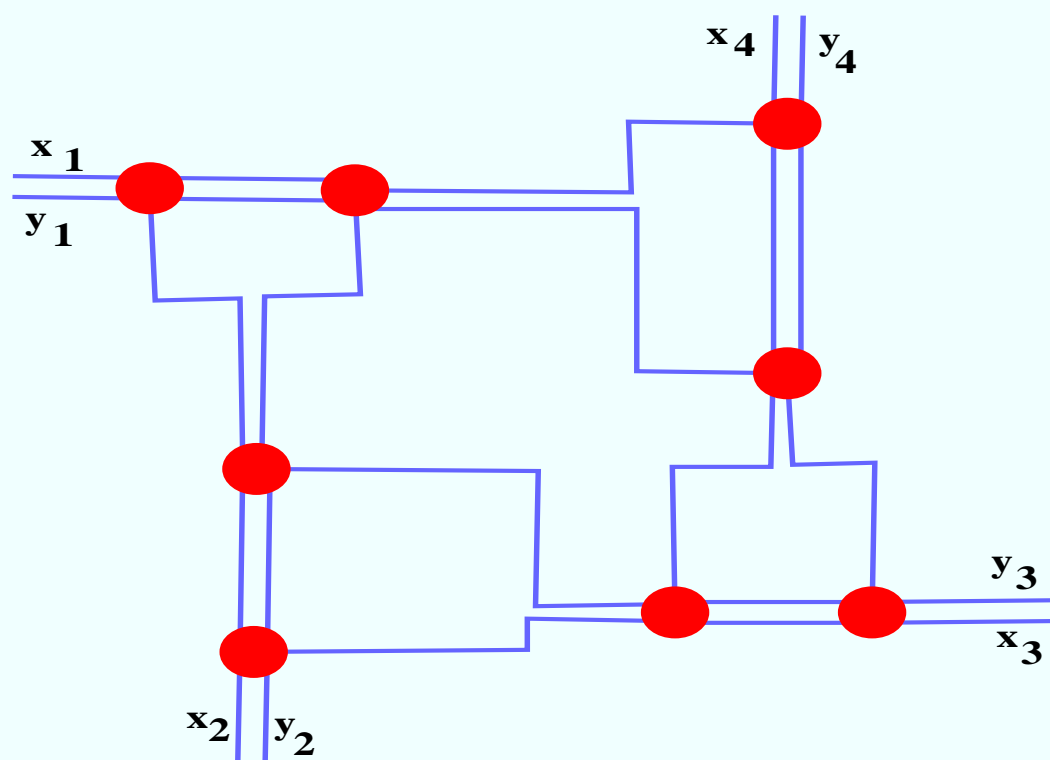


Pentru orice $n \geq 2$, $n \in \mathbb{N}$, definim graful H_n astfel: 1. H_2 se obține identificînd muchiile u și u' din $2H$:



2. $\forall n > 2$ H_n se construiește astfel:

- considerăm n copii izomorfe disjuncte ale lui $2H$, $2H^i$ $i = 1, n$, avînd perechile de muchii libere (x_i, y_i) , (x'_i, y'_i) , (u_i, u'_i) .
- pentru $i = 1, n-1$ identificăm perechile $(u_i, u'_i) \equiv (x'_{i+1}, y'_{i+1})$.
- identificăm $(u_n, u'_n) \equiv (x'_1, y'_1)$.



H_n are n perechi de intrare (x_i, y_i) $i = 1, n$, și este construit din $2n$ componente H .

În orice 3-colorare c a lui H_n avem

a) $c(x_1) = c(y_1) \Rightarrow c(u_1) = c(u'_1)$ (din lema 2 și construcția lui $2H$), $\Rightarrow c(x'_2) = c(y'_2)$ (din identificarea de la construcția lui H_n) $\Rightarrow c(x_2) = c(y_2)$ (lema 2) $\Rightarrow c(u_2) = c(u'_2) \Rightarrow \dots \Rightarrow c(x_n) = c(y_n)$.

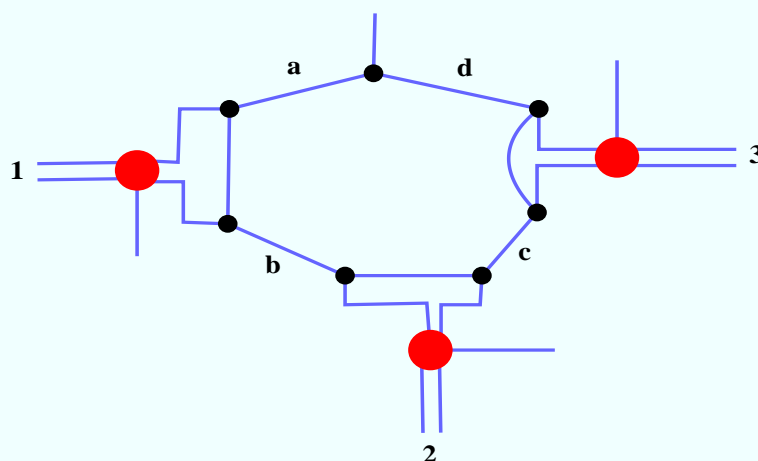
b) $c(x_1) \neq c(y_1) \Rightarrow c(x'_1) \neq c(y'_1)$ (lema 2) $\Rightarrow c(u_n) \neq c(u'_n)$ (din identificarea de la construcția lui H_n) $\Rightarrow c(x_n) \neq c(y_n)$ (din lema 2, prin reducere la absurd) $\Rightarrow c(x'_n) \neq c(y'_n) \Rightarrow c(u_{n-1}) \neq c(u'_{n-1}) \Rightarrow \dots \Rightarrow c(x_2) \neq c(y_2)$.

Rezultă de aici că în orice 3-colorare a muchiilor lui H_n ($n \geq 2$) toate perechile de muchii de intrare reprezintă aceeași valoare de adevăr.

Pentru $n = 1$ se poate construi H_1 cu o singură pereche de intrare, care să poată reprezenta oricare din cele 2 valori de adevăr. Am dovedit

Lemă. 3. Pentru orice $n \in \mathbb{N}^*$ se poate construi în timp polinomial graful H_n avînd n perechi de muchii de "intrare", cu proprietatea că în orice 3-colorare a muchiilor lui H_n , toate cele n perechi reprezintă aceeași valoare de adevăr (sau, în orice pereche de intrare cele două muchii au culori diferite, sau, în orice pereche de intrare cele două muchii sînt la fel colorate).

Lemă. 4. Fie F graful



În orice 3-colorare c a muchiilor lui F , măcar una din cele 3 perechi de muchii de intrare are muchiile la fel colorate (reprezintă valoarea adevăr).

Demonstrația este imediată și rezultă prin reducere la absurd: dacă există o 3-colorare a muchiilor lui F , astfel încît în cele 3 perechi muchiile sînt colorate diferit, atunci $c(a) = c(b)$, $c(b) = c(c)$ și $c(c) = c(d)$ (s-a utilizat lema 2), deci $c(a) = c(d)$ contrazicînd faptul că c este colorare a muchiilor. Notăm că există colorări c cu proprietatea din enunț.

Demonstrația teoremei 3. Fie $U = \{u_1, \dots, u_n\}$ ($n \in \mathbf{N}^*$), $C = C_1 \wedge \dots \wedge C_m$ ($m \in \mathbf{N}^*$), $C_i = v_{i_1} \vee v_{i_2} \vee v_{i_3}$ cu v_{i_j} literali ($v_{i_j} = u_\alpha$ sau $v_{i_j} = \bar{u}_\alpha$) reprezentînd datele unei probleme 3SAT.

Vom construi un graf 3-regulat G cu proprietatea că $\chi'(G) = 3$ dacă și numai dacă există $t : U \rightarrow \{adev, fals\}$ astfel încît $t(C) = adev$.

Construcția se face în timp polinomial în raport cu n și m și se poate descrie astfel:

1. Pentru fiecare variabilă booleană u_i $i = 1, n$ se determină x_i , numărul aparițiilor (negate sau nu) ale lui u_i în C și se consideră câte o copie disjunctă a grafului H_{x_i} .

2. Pentru fiecare clauză C_j $j = 1, m$ se consideră câte o copie disjunctă F_j a grafului F .

3. Pentru $j = 1, m$, fie $C_j = v_{j1} \vee v_{j2} \vee v_{j3}$.

Dacă v_{ji} ($i = 1, 3$) este u_α și reprezintă intrarea cu numărul y a variabilei u_α atunci perechea de intrare cu numărul y din H_{x_α} se identifică cu perechea de intrare numărul i din componenta F_j .

Dacă v_{ji} este \bar{u}_α se plasează o componentă inversoare H între perechea i a lui F_j și perechea de intrare cu numărul y din H_{x_α} , și se identifică corespunzător intrările lui H cu cele precizate mai sus.

4. Graful obținut la 1-3 are anumite muchii pentru care nu s-a precizat o extremitate. Considerăm o copie izomorfă disjunctă a sa și identificăm perechile corespunzătoare de astfel de muchii, din cele două grafuri. Se obține un graf G 3-regulat.

Construcția lui G și lemele 3 și 4 simulează algoritmul de calcul al valorii de adevăr a lui C într-o atribuire t fixată și deci G **admite o 3-colorare dacă și numai dacă C este satisfiabilă.**

4. Probleme hamiltoniene

Definiție: Fie $G = (V(G), E(G))$ un (di)graf. Un circuit C al lui G se numește **circuit hamiltonian** dacă $V(C) = V(G)$.

Un drum deschis D al lui G se numește **drum hamiltonian** dacă $V(D) = V(G)$.

Un (di)graf care are un circuit hamiltonian se numește **(di)graf hamiltonian**.

Un (di)graf care are un drum hamiltonian se numește **(di)graf trasabil**.

Teoremă. 4. (*Nash-Williams 1969*) *Problemele următoare sînt polinomial echivalente:*

CH : *Dat G graf. Este G hamiltonian ?*

TR : *Dat G graf. Este G trasabil ?*

DCH: *Dat G digraf. Este G hamiltonian ?*

DTR: *Dat G digraf. Este G trasabil ?*

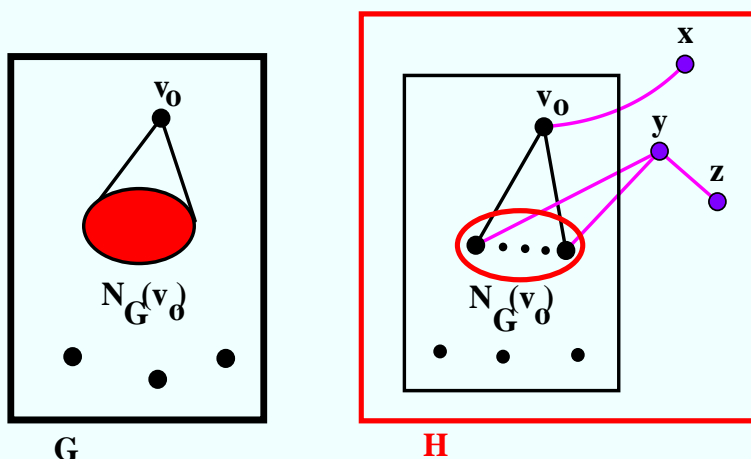
BCH: *Dat G graf bipartit. Este G hamiltonian ?*

Demonstrație:

$CH \propto TR$

Fie G un graf și $v_0 \in V(G)$ un vîrf fixat al său. Construim (în timp polinomial) un graf H astfel încît G este hamiltonian dacă și numai dacă H este trasabil.

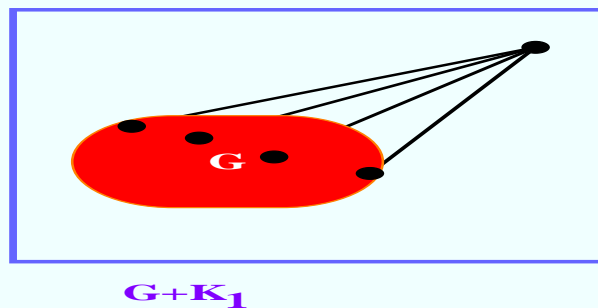
Fie $V(H) = V(G) \cup \{x, y, z\}$ și $E(H) = E(G) \cup \{xv_0, yz\} \cup \{wy \mid w \in N_G(v_0) \wedge wv_0 \in E(G)\}$.



Se observă că H este trasabil dacă și numai dacă are un drum hamiltonian D cu extremitățile x și z (care au gradul 1 în H). D există în H dacă și numai dacă în G există un drum hamiltonian cu o extremitate v_0 și cealaltă un vecin al lui v_0 , deci dacă și numai dacă G este hamiltonian.

$$TR \propto CH$$

Fie G un graf. Considerăm $H = G + K_1$. H este hamiltonian dacă și numai dacă G are un drum hamiltonian.



Echivalența problemelor DCH și DTR de demonstrează în mod similar.

$$CH \propto DCH$$

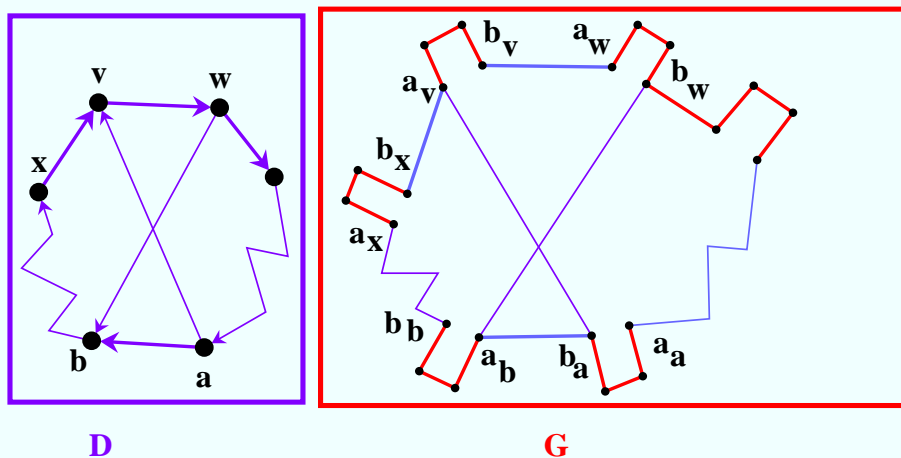
Fie G un graf. Fie D digraful obținut din G înlocuind fiecare muchie cu o pereche de arce simetrice. Orice circuit hamiltonian în G induce un circuit hamiltonian în D și reciproc.

$$DCH \propto CH$$

Fie $D = (V(D), E(D))$ un digraf. Pentru orice vîrf $v \in V(D)$ asociem un drum de lungime 3, P_v cu extremitățile a_v și b_v

$$P_v = (\{a_v, c_v, d_v, b_v\}, \{a_v c_v, c_v d_v, d_v b_v\}).$$

Pentru orice arc $vw \in E(D)$ considerăm muchia $b_v a_w$. Fie G graful cu $V(G) = \cup_{v \in V(D)} V(P_v)$ și $E(G) = \cup_{v \in V(D)} E(P_v) \cup \{b_v a_w \mid vw \in E(D)\}$. Evident, G se poate construi în timp polinomial în raport cu numărul de vîrfuri ale lui D .



Se observă că orice circuit C al lui D induce un circuit în G și reciproc orice circuit al lui D este generat de un circuit al lui G . Rezultă: D este hamiltonian dacă și numai dacă G este hamiltonian.

Să observăm că dacă C este un circuit al lui G , acesta este generat de un circuit C' al lui D și în plus lungimea circuitului C , $l(C)$ satisface $l(C) = 3l(C') + l(C') = 4l(C')$; deci orice circuit al lui G este de lungime pară, adică G este bipartit.

Rezultă că, de fapt, am demonstrat și că $DCH \propto BCH$.

Cum $BCH \propto CH$ este evidentă, rezultă că teorema este demonstrată.

Teoremă. 5. (Karp 1972) $SM \propto CH$.

Demonstrație: Fie $G = (V, E)$ graf și $j \in \mathbf{Z}_+$ datele unei probleme SM . Construim în timp polinomial (în raport cu $n = |V|$) un graf H astfel încât: există S stabilă în G cu $|S| \geq j$ dacă și numai dacă H este hamiltonian.

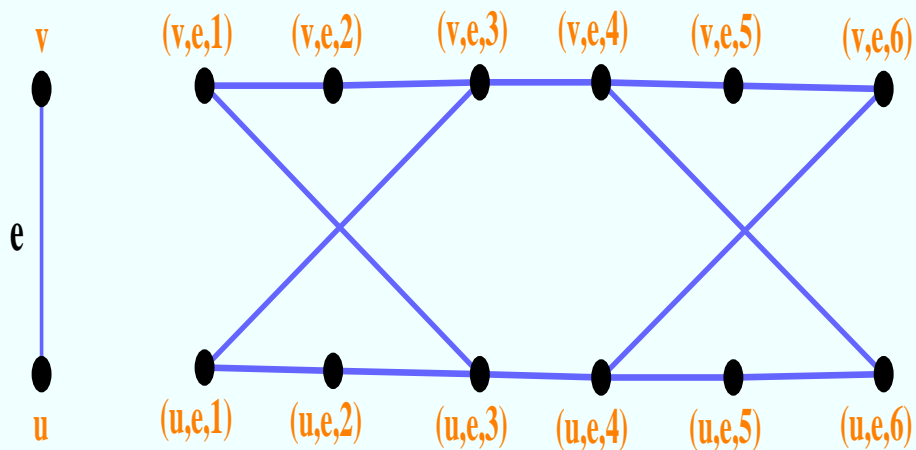
Fie $k = |V| - j$. Putem presupune $k > 0$, altfel luăm drept H orice graf nehamiltonian.

i) Fie $A = \{a_1, a_2, \dots, a_k\}$ o mulțime de k vîrfuri distincte.

ii) Pentru orice $e = uv \in E(G)$ considerăm graful $G'_e = (V'_e, E'_e)$ cu

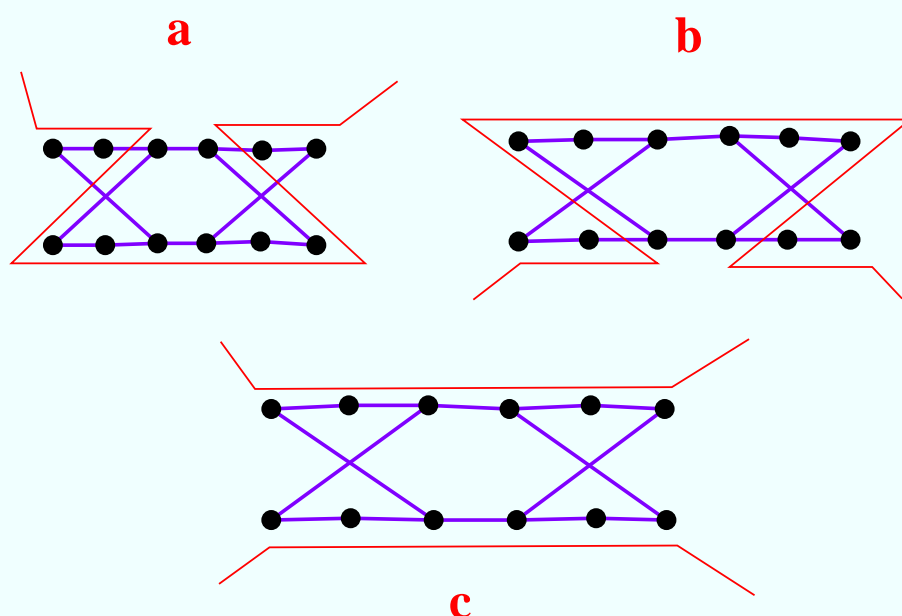
$V'_e = \{(w, e, i); w \in \{u, v\}, i = 1, 6\}$ și

$E'_e = \{(w, e, i)(w, e, i + 1); w \in \{u, v\}, i = 1, 5\} \cup \{(u, e, 1)(v, e, 3), (u, e, 6)(v, e, 4), (v, e, 1)(u, e, 3), (v, e, 6)(u, e, 4)\}$.



Graful G'_e a fost ales astfel încît, dacă este subgraf al unui graf hamiltonian H și niciunul din vîrfurile (w, e, i) cu $w \in \{u, v\}$ și $i = \overline{2, 5}$ nu are alt vecin în H decît cele din G'_e ,

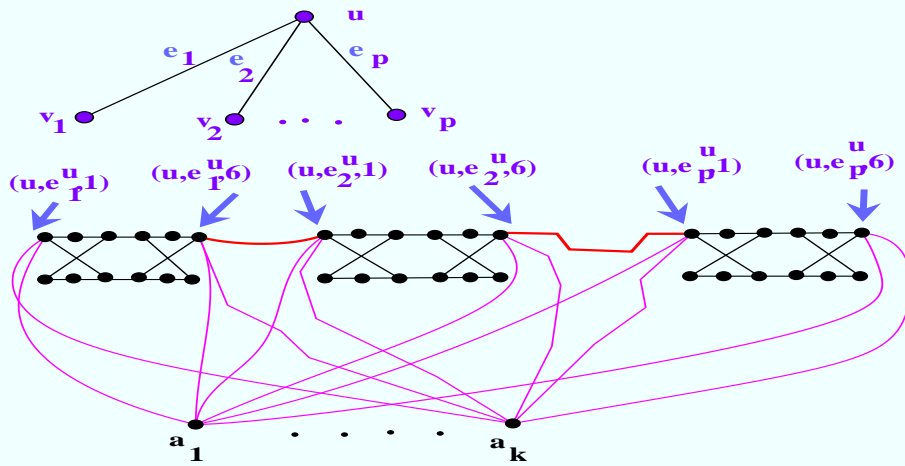
atunci singurele posibilități de traversare de către un circuit hamiltonian al lui H a vîrfurilor din G'_e sînt (a) (b) și (c) indicate în figura următoare (orice altă parcurgere lasă un vîrf netraversat):



Deci, dacă circuitul hamiltonian "intră" în G'_e printr-un vîrf de tip u , $((u, e, 1)$ sau $(u, e, 6))$ atunci, va "ieși" tot printr-un vîrf corespunzător lui u .

(iii) Pentru fiecare vîrf $u \in V$, se consideră, (înr-o ordine oarecare) toate muchiile lui G incidente cu u : $e_1^u = uv_1, e_2^u = uv_2, \dots, e_p^u = uv_p$, unde, $p = d_G(u)$.

Fie $E_u'' = \{(u, e_i^u, 6)(u, e_{i+1}^u, 1); i = 1, p-1\}$ și $E_u''' = \{a_i(u, e_1^u, 1), a_i(u, e_p^u, 6); i = 1, k\}$.

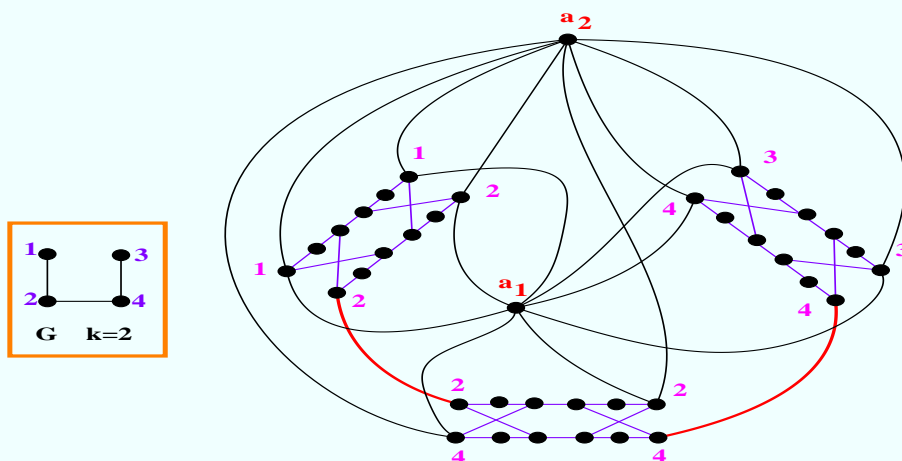


Graful H va avea

$$V(H) = A \cup \cup_{e \in E} V_e' \text{ și } E(H) = \cup_{e \in E} E_e' \cup \cup_{u \in V} (E_u'' \cup E_u''').$$

Avem $|V(H)| = k + 12 \cdot |E|$; $|E(H)| = 14 \cdot |E| + 2k \cdot |V_0| + 2 \cdot |E| - |V_0|$, unde $V_0 \subseteq V$ este mulțimea vîrfurilor neizolate din G ; deci H se poate construi în timp polinomial în raport cu $|V|$.

Exemplu:

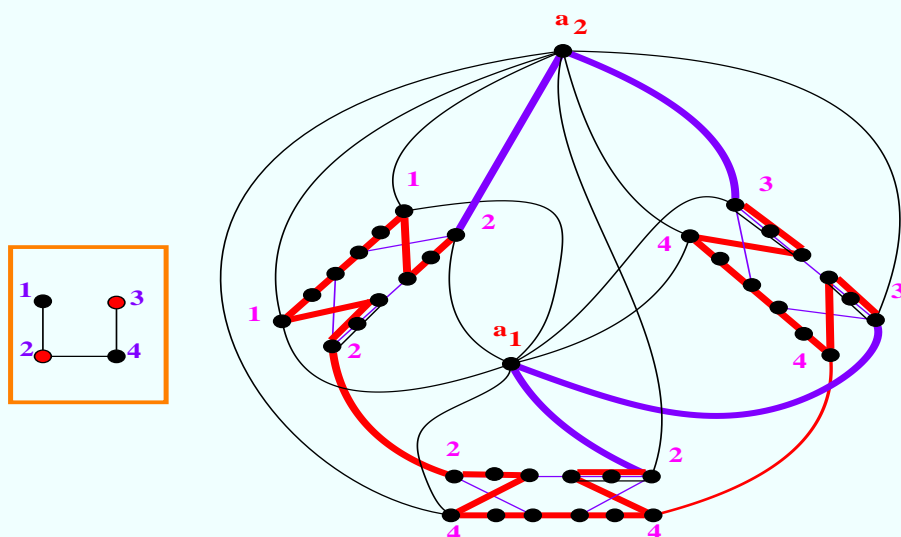


1. Dacă H este hamiltonian, atunci există C un circuit hamiltonian în H . Cum A este o mulțime stabilă în H , A va descompune circuitul C în exact k drumuri disjuncte (cu excepția extremităților): $D_{a_{i_1}a_{i_2}}, D_{a_{i_2}a_{i_3}}, \dots, D_{a_{i_k}a_{i_1}}$.

Fie $D_{a_{i_j}a_{i_{j+1}}}$ un astfel de drum ($j+1 = 1 + (j \pmod k)$).

Din construcția lui H , rezultă că primul vîrf care urmează lui a_{i_j} pe acest drum va fi $(v_{i_j}, e_1^{v_{i_j}}, 1)$ sau $(v_{i_j}, e_p^{v_{i_j}}, 6)$ unde $p = d_G(v_{i_j})$ $v_{i_j} \in V$.

În continuare, $D_{a_{i_j}a_{i_j+1}}$ va intra în componenta G_{e_1} sau G_{e_p} din care va ieși tot printr-un vîrf corespunzător lui v_{i_j} . Dacă vîrfurile următoare nu este a_{i_j+1} , se intră într-o componentă corespunzătoare următoarei muchii incidente cu v_{i_j} și va ieși din aceasta, tot printr-un vîrf corespunzător lui v_{i_j} .



Rezultă că drumului $D_{a_{i_j}a_{i_{j+1}}}$ i se poate asocia în mod unic vârful $v_{i_j} \in V$ și că prima și ultima muchie a acestui drum sînt $a_{i_j}(v_{i_j}, e_t^{v_{i_j}}, x)$, $a_{i_{j+1}}(v_{i_j}, e_{t'}^{v_{i_j}}, x')$ cu $t = 1$ și $t' = d_G(u)$, $x = 1$, $x' = 6$ sau $t = d_G(u)$, $t' = 1$, $x = 6$, $x' = 1$. Aceasta implică faptul că v_{i_j} sînt distincte.

Fie $V^* = \{v_{i_1}, v_{i_2}, \dots, v_{i_k}\}$. Cum C este hamiltonian în H , rezultă că $\forall e \in E$ există un drum $D_{a_{i_j}a_{i_{j+1}}}$ care trece prin G'_e deci există $v \in V^*$ incident cu e . Rezultă că $S = V - V^*$ este stabilă în G și $|S| = j$. Deci am obținut că dacă H este hamiltonian, în G există o mulțime stabilă de cardinal j , prin urmare răspunsul la SM este da.

2. Presupunem că răspunsul la SM este da, deci în G există S_0 stabilă cu $|S_0| \geq j$. Există atunci $S \subseteq S_0$ stabilă cu $|S| = j$.

Fie $V^* = V - S = \{v_1, v_2, \dots, v_k\}$. Considerăm în H pentru fiecare muchie $e = uv \in E$:

- cele două drumuri din G'_e situația (a), dacă $u, v \in V^*$,
- drumul din G'_e situația (b) dacă $u \in V^*, v \notin V^*$
- drumul din G'_e situația (c) dacă $u \notin V^*, v \in V^*$.

Dacă la reuniunea acestor drumuri adăugăm muchiile

$a_i(v_i, e_1^{v_i}, 1),$
 $(v_i, e_1^{v_i}, 6)(v_i, e_2^{v_i}, 1), \dots, (v_i, e_{p-1}^{v_i}, 6)(v_i, e_p^{v_i}, 1),$
 $(v_i, e_p^{v_i}, 6)a_{i+1},$ (cu $p = d_G(v_i)$) pentru $i = 1, k,$
 se obține un circuit hamiltonian în H .

5. Problema comisului voiajor.

Dat $G = (V, E)$ un graf și o funcție de pondere $d : E \rightarrow \mathbf{R}_+$, să se determine un circuit hamiltonian H_0 în G astfel încât suma ponderilor muchiilor lui H_0 să fie minimă.

Dacă graful G reprezintă rețeaua rutieră între o mulțime V de localități, iar funcția d avînd interpretarea $d(uv) =$ distanța pe ruta directă dintre localitățile u și v , și se fixează un centru $v_0 \in V$, atunci circuitul hamiltonian H_0 reprezintă cel mai economic mod de vizitare a localităților din V de către un **comis voiajor** ce pleacă din centrul v_0 și se întoarce în același loc, după ce a vizitat fiecare localitate exact o dată.

Problema este interesantă, nu pentru rezolvarea acestei aplicații, mai mult sau mai puțin importantă, ci pentru că ea apare în numeroase probleme de optimizare discretă, motivate, de exemplu, de construirea circuitelor integrate pe scară mare.

În cele ce urmează, vom considera o formă echivalentă a ei

CV Dat $n \in \mathbf{Z}_+$ ($n \geq 3$) și $d : E(K_n) \rightarrow \mathbf{R}_+$, să se determine H_0 circuit hamiltonian în graful complet K_n cu $d(H_0)$ minim printre toate circuitele hamiltoniene ale lui K_n .

Observații 1⁰. $d(H_0) = \sum_{e \in E(H_0)} d(e)$.

2⁰. Dacă graful pentru care se cere rezolvarea nu este complet, se introduc muchiile lipsă, atribuindu-le drept ponderi, $M \in \mathbf{R}_+$ cu $M > |V| \cdot \max_{e \in E} d(e)$.

Notăm aici, că în enunțul problemei ne-am limitat numai la "cazul simetric", o problemă similară se poate considera și pentru G digraf oarecare.

3⁰. În studiul complexității acestei probleme vom considera că $d(e) \in \mathbf{Z}_+$.

Problema de decizie asociata va fi

DCV

Instanță: $n \in \mathbf{Z}_+$ ($n \geq 3$), $d : E(K_n) \rightarrow \mathbf{Z}_+$ și $B \in \mathbf{Z}_+$

Intrebare: Există H_0 circuit hamiltonian în K_n

astfel încît $d(H_0) \leq B$?

Teoremă. 6. $CH \propto DCV$.

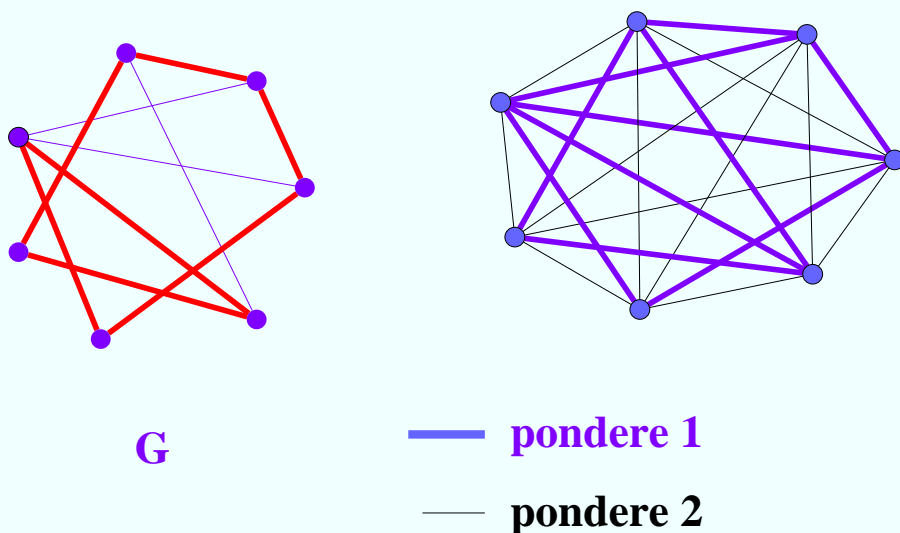
Demonstrație: Fie $G = (V, E)$, ($|V| = n$) reprezentînd datele unei probleme CH . Construim în timp polinomial o problemă DCV cu proprietatea că în K_n există un circuit hamiltonian de pondere totală care nu depășește B dacă și numai dacă G este hamiltonian.

Fie

$$d(vw) = \begin{cases} 1 & \text{dacă } vw \in E(G) \\ 2 & \text{dacă } vw \in E(\overline{G}). \end{cases}$$

și considerăm $B = n$.

Atunci în K_n există un circuit hamiltonian de pondere $\leq n$ dacă și numai dacă G are un circuit hamiltonian.



Rezultă, de aici, dificultatea rezolvării problemei CV.

O soluție pentru abordarea problemei CV, ar fi aceea de a considera algoritmi A , care pentru datele unei probleme CV vor oferi în timp polinomial (în raport cu n) un circuit hamiltonian H_A , care va aproxima soluția optimă H_0 .

Măsuri ale eficienței unei astfel de "euristici" A pot fi considerate numerele:

$$R_A(n) = \sup_{\substack{d: E(K_n) \rightarrow \mathbf{R}_+ \\ d(H_0) \neq 0}} \frac{d(H_A)}{d(H_0)}$$

$$R_A = \sup_{n \geq 3} R_A(n).$$

Evident, s-ar dori ca aceste numere să fie finite. În cazul general, condiția ca R_A să fie finit este la fel de dificilă cu aceea a rezolvării eficiente "exact" a problemei CV. În adevăr, are loc

Teoremă. 8. *Dacă există un algoritm aproximativ A cu timp de lucru polinomial pentru CV, astfel încît $R_A < \infty$, atunci CH se poate rezolva în timp polinomial.*

Demonstrație: Fie A un algoritm cu timp de lucru polinomial și cu $R_A < \infty$. Există deci $k \in \mathbb{Z}_+$ astfel încât $R_A \leq k$.

Fie $G = (V, E)$ un graf arbitrar, intrare pentru CH. Dacă $n = |V|$ atunci definim $d : E(K_n) \rightarrow \mathbb{Z}_+$ prin

$$d(uv) = \begin{cases} 1 & \text{dacă } uv \in E \\ kn & \text{dacă } uv \notin E \end{cases}$$

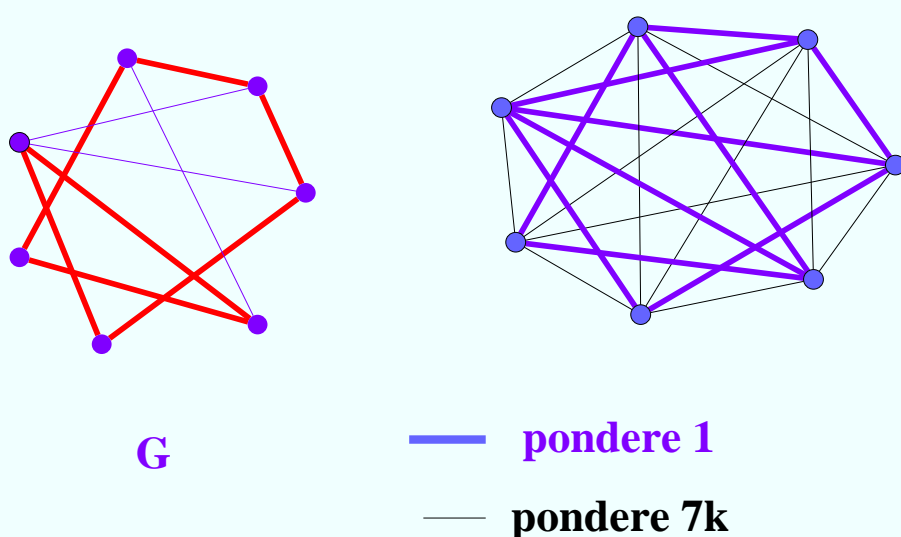
Evident, G este hamiltonian dacă și numai dacă soluția optimă H_0 a lui CV astfel construit satisface $d(H_0) = n$.

Aplicăm A pentru rezolvarea problemei CV. Se obține în timp polinomial o soluție H_A .

Dacă $d(H_A) \leq kn$, atunci $d(H_A) = n$ și $H_A = H_0$.

Dacă $d(H_A) > kn$, atunci $d(H_0) > n$. În adevăr, presupunând că $d(H_0) = n$, avem $\frac{d(H_A)}{d(H_0)} \leq k$, deci $d(H_A) \leq kd(H_0) = kn$, contradicție.

Rezultă că G este hamiltonian dacă și numai dacă $d(H_A) \leq kn$, și cum timpul de lucru al lui A este polinomial, rezultă că CH se poate rezolva în timp polinomial.



Observație. Enunțul teoremei se poate formula echivalent și astfel:

Dacă $P \neq NP$, atunci nu există un algoritm aproximativ A polinomial și cu $R_A < \infty$.

VII. Abordări ale unor probleme NP-dificile pe grafuri.

1. Euristici

Se proiectează algoritmi eficienți care, deși nu rezolvă problema, oferă soluții aproximative, care se pot analiza și uneori se pot folosi în metaeuristici (de exemplu, cele care imită procese din natură). Pentru unele euristici se precizează și instanțe ale problemelor pentru care ele funcționează corect.

Partizanii unor astfel de abordări apelează la așa numita **no free lunch theorem** (Wolpert and Macready, 1994).

all non-repeating search algorithms have the same mean performance when averaged uniformly over all possible objective functions $f : X \rightarrow Y$

1.1 Colorarea vârfurilor unui graf

Algoritmul greedy de colorare

Un mod simplu și natural de a colora un graf, cu nu foarte multe culori, este dat de următorul algoritm:

pornind de la o ordine fixată a vârfurilor lui G , v_1, v_2, \dots, v_n , parcurgem această listă, colorând fiecare vârf v_i cu prima culoare disponibilă, adică cu cea mai mică culoare (număr întreg pozitiv) nefolosită de nici unul dintre vecinii lui v_i , deja întâlniți până în acel moment.

Acestă metodă poartă denumirea de *algoritmul greedy de colorare*, datorită alegerii, de fiecare dată, **a celei mai mici culori** care s-ar putea folosi.

Fie $G = (V, E)$, cu $V = \{1, 2, \dots, n\}$ și π o permutare a lui V (ce corespunde unei ordonări a mulțimii vârfurilor).

Algoritmul construiește colorarea c ce utilizează $\chi(G, \pi)$ culori,
 $c : \{1, 2, \dots, n\} \longrightarrow \{1, 2, \dots, \chi(G, \pi)\}.$

Algoritmul greedy-color

```
-    $c(\pi_1) \leftarrow 1; \chi(G, \pi) \leftarrow 1; S_1 \leftarrow \{\pi_1\};$ 
-   for  $i \leftarrow 2$  to  $n$  do
-   {    $j \leftarrow 0;$ 
-       repeat
-            $j \leftarrow j + 1;$ 
-           determină primul vârf (conform ordonării
-            $\pi$ ),  $v$  din  $S_j$  a.  $\hat{1}. \pi_i v \in E(G);$ 
-           if  $v$  există then
-                $first(\pi_i, j) \leftarrow v$ 
-           else
-               {    $first(\pi_i, j) \leftarrow 0;$ 
-                    $c(\pi_i) \leftarrow j;$ 
-                    $S_j \leftarrow S_j \cup \{\pi_i\};$ 
-               }
-           until  $first(\pi_i, j) = 0$  or  $j = \chi(G, \pi);$ 
-           if  $first(\pi_i, j) \neq 0$  then
-               {    $c(\pi_i) \leftarrow j + 1;$ 
-                    $S_{j+1} \leftarrow \{\pi_i\};$ 
-                    $\chi(G, \pi) \leftarrow j + 1;$ 
-               }
-   }
```

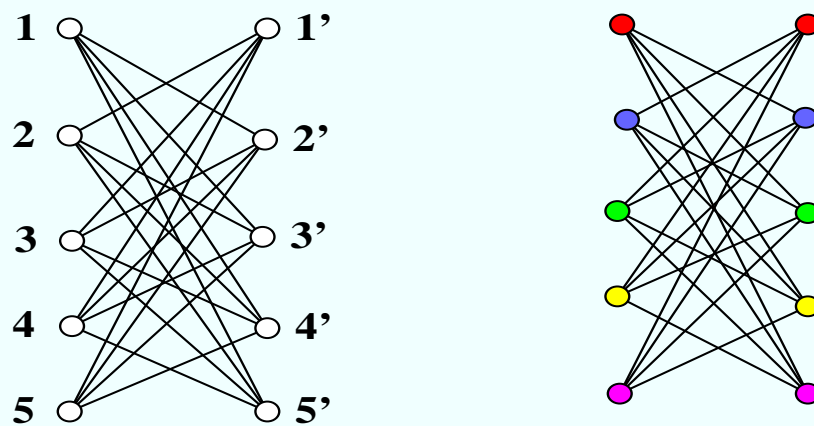

Să observăm că folosind algoritmul greedy nu vom obține niciodată mai mult de $1 + \Delta(G)$ culori.

Rezultă că $\chi(G) \leq 1 + \Delta(G)$.

Însă $\Delta(G)$ se poate îndepărta oricât de mult de $\chi(G)$ și există grafuri și permutări pentru care algoritmul poate să greșească la fel de mult.

Un astfel de exemplu este următorul, unde graful G este obținut din graful bipartit complet $K_{n,n}$, având mulțimea de vârfuri $\{1, 2, \dots, n\} \cup \{1', 2', \dots, n'\}$, din care se elimină muchiile $11', 22', \dots, nn'$.

Fixând ordinea $1, 1', 2, 2', \dots, n, n'$ algoritmul va obține colorarea c cu $c(1) = c(1') = 1, c(2) = c(2') = 2, \dots, c(n) = c(n') = n$ având deci n culori; însă, cum G este bipartit, $\chi(G) = 2$.



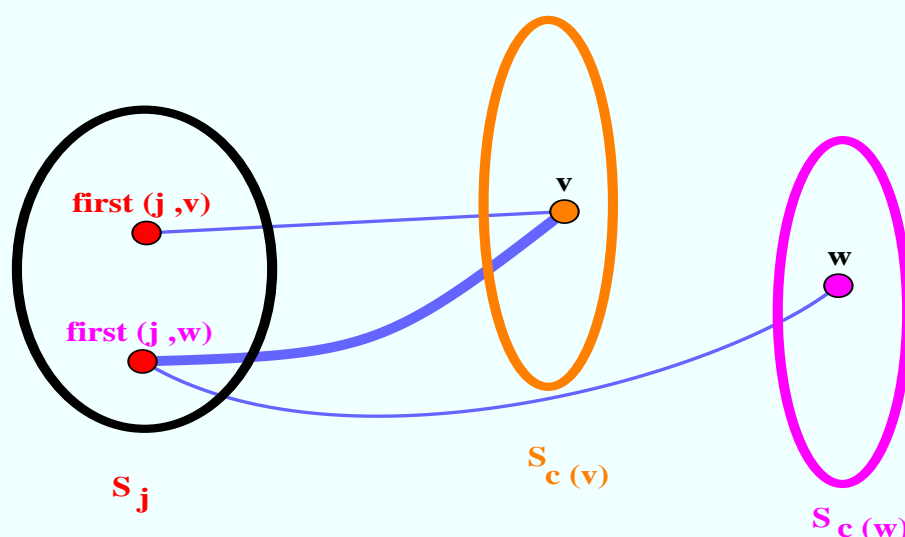
pi: 1, 1', 2, 2', 3, 3', 4, 4', 5, 5'

Este ușor de văzut că, pentru orice graf G , există totuși permutări pentru care algoritmul oferă o colorare optimală.

Într-adevăr dacă $S_1, S_2, \dots, S_{\chi(G)}$ sunt clasele de colorare corespunzătoare unei colorări optimale, atunci pentru orice permutare care păstrează ordinea claselor (adică dacă $i < j$, $v \in S_i$ și $w \in S_j$ atunci v se află înaintea lui w în permutare), executând algoritmul greedy, se vor obține $\chi(G)$ culori.

Teoremă. 1. Dacă pentru orice $vw \in E$ și orice $j < \min\{c(v), c(w)\}$ astfel încât $first(v, j) < first(w, j)$ (în ordonarea π) avem că $vfirst(w, j) \in E$ atunci $\chi(G, \pi) = \chi(G)$.

Condiția din enunț poate fi verificată în timp liniar în raport cu numărul de muchii ale grafului și poate reprezenta un ultim pas al algoritmului.



În demonstrația teoremei se arată că, atunci când condiția este îndeplinită, $\chi(G, \pi) = \omega(G)$ și prin urmare, în acest caz, se obține și numărul de clică al grafului.

Pe de altă parte, există grafuri pentru care această condiție nu va fi îndeplinită de nici o permutare.

Algoritmul **Dsatur** { *Degree of Saturation* }

Acest algoritm (Brélaz, 1979) este o metodă secvențială dinamică de colorare.

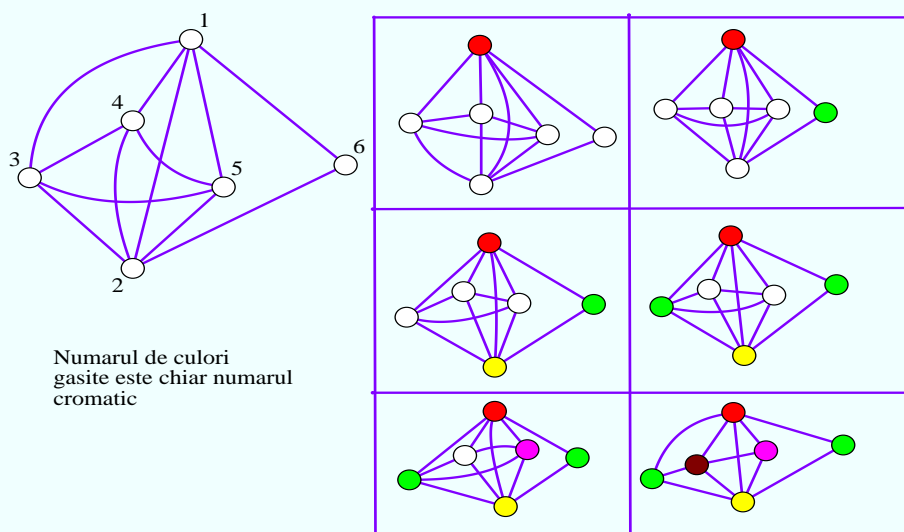
Ideea este de colora vârfurile pe rând, alegând de fiecare dată vârful cu un număr maxim de constrângeri privitoare la culorile disponibile acestuia.

Această abordare este într-un fel opusă primeia (cea greedy) deoarece se aleg vârfuri care formează “clici” mari în raport cu vârfurile deja alese (spre deosebire de mulțimi stabile mari în cazul greedy).

Dacă G este un graf și c o colorare parțială a vârfurilor lui G , definim *gradul de saturație* al unui vârf v , notat $d_{sat}(v)$, ca fiind numărul de culori diferite din vecinătatea acestuia.

Algoritmul DSatur

- ordonează vârfurile în ordinea descrescătoare a gradelor lor;
- atribuie unui vârf de grad maxim culoarea 1;
- **while** există vârfuri necolorate **do**
- { alege un vf. necol. cu gr. de satur. maxim; dacă acesta
- nu-i unic, alege un vf. de grad maxim în subgr.necolorat;
- colorează vârful ales cu cea mai mică culoare posibilă;
- }



Teoremă. 2. Algoritmul DSatur garantează găsirea numărului cromatic pentru grafurile bipartite.

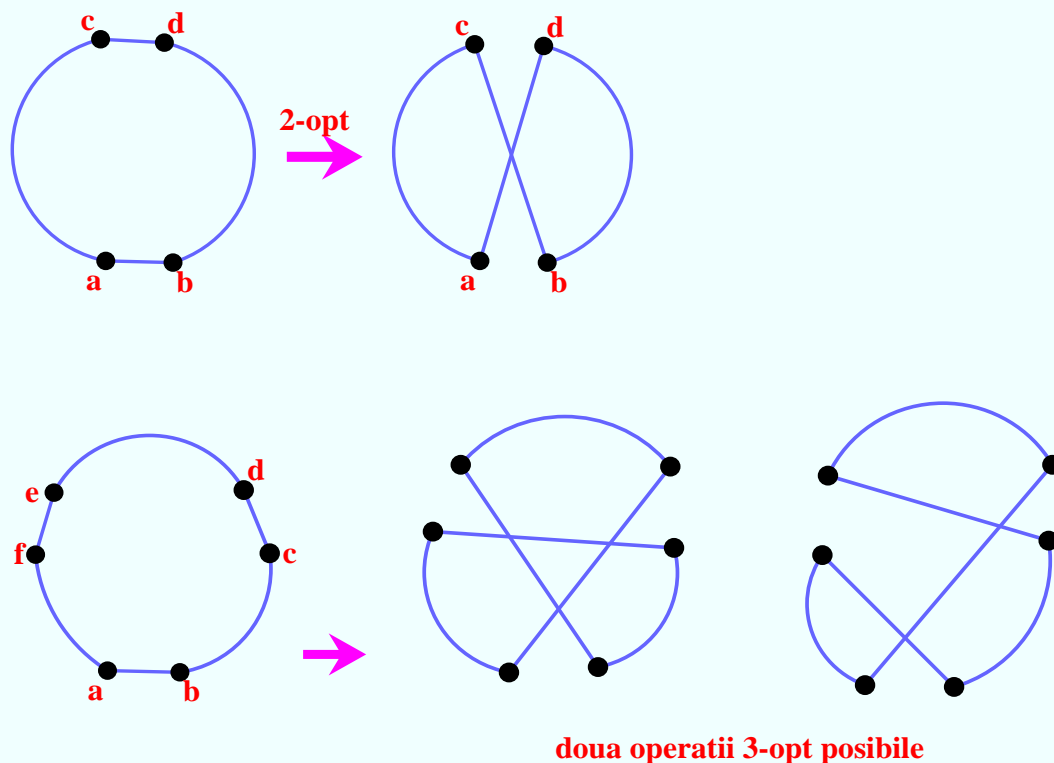
1.2 Problema comisului voiajor

Cele mai performante euristici se bazează pe principiul **optimizării locale**:

- se pornește cu un circuit hamiltonian ales aleator sau obținut cu o **euristică de tip construcție** (tour-construction heuristic), de exemplu greedy;
- se încearcă repetat îmbunătățiri locale, trecând de la soluția curentă la o soluție vecină din spațiul soluțiilor (**neighborhood search**).

Cele mai cunoscute modificări locale sunt așa numitele **2-opt**, **3-opt** sau combinații ale acestora care au condus la faimosul algoritm **Lin-Kernighan**.

Figura următoare sugerează cele două operații 2-opt și 3-opt.



Pentru reprezentarea circuitului hamiltonian se utilizează structura de date *Tour* care trebuie să suporte (eficient) următoarele 4 operații:

$Next(a)$: întoarece vârful următor lui a pe circuit;

$Prev(a)$: întoarece vârful dinaintea lui a pe circuit;

$Between(a, b, c)$: întoarce *true* dacă la traversarea circuitului din a se întâlnește mai întâi b și apoi c ;

$Flip(a, b, c, d)$: Parametrii verifică $a = Next(b)$ și $d = Next(c)$. Se actualizează circuitul prin înlocuirea muchiilor ab și cd cu bc și ad . Se actualizează orientarea circuitului (pointerii $Next$ și $Prev$) în mod corespunzător.

Operația 3-opt se realizează cu ajutorul a două sau trei *Flip*. Pentru evitarea formării de circuite disjuncte este nevoie de funcția *Between*. Operația mai complicată λ -opt, utilizată de algoritmul Lin-Kernighan se exprimă ca un 3-opt și o secvență de 2-opt.

Principala problemă în implementarea operației *Flip* este de a determina eficient șirul de schimbări ale pointerilor *Next* și *Prev* (inversarea unuia din cele două drumuri; care din ele ?).

Un algoritm λ -opt se bazează pe conceptul de *λ -optimalitate*:

Un tur este *λ -optimal* (sau simplu *λ -opt*) dacă nu este posibil să se obțină un tur mai scurt (de cost mai mic) prin înlocuirea unei mulțimi de λ muchii ale sale cu altă mulțime de λ muchii.

Observație: Dacă un tur este λ -optimal atunci el este λ' -optimal pentru orice λ' , $2 \leq \lambda' \leq \lambda$. De asemenea, un tur cu n vârfuri este optimal dacă și numai dacă este n -optimal.

Numărul operațiilor necesare testării tuturor λ -schimbărilor posibile crește rapid pentru n mare. Într-o implementare naivă, testarea unei λ -schimbări necesită $O(n^\lambda)$ operații. Ca urmare, cele mai folosite valori sunt $\lambda \in \{2, 3\}$ (dar se cunosc și abordări cu $\lambda = 4, 5$).

E dificil să se estimeze ce valoare a lui λ trebuie aleasă pentru ca să se obțină cel mai bun compromis între timpul de execuție și calitatea soluției modificate.

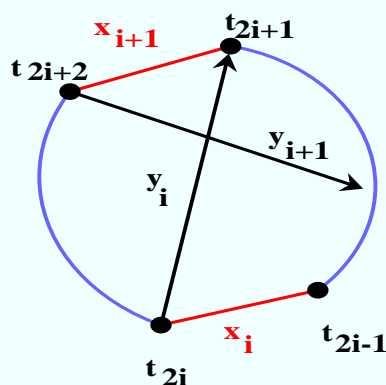
Lin și Kernighan au abordat această problemă prin considerarea unei metode numită *variable λ -opt*: algoritmul va schimba valoarea lui λ în timpul execuției.

La fiecare iterație se încearcă valori crescătoare pentru λ pentru a obține un tur mai scurt.

Fie T turul curent. La fiecare iterație a algoritmului se caută mulțimile de muchii $X = \{x_1, \dots, x_r\}$ și $Y = \{y_1, \dots, y_r\}$ astfel ca înlocind X cu Y în T , să se obțină un tur mai bun. Această interschimbare o numim r -opt move. Cele 2 mulțimi se construiesc element cu element. Inițial sunt vide și în pasul i se adaugă x_i la X și y_i la Y .

Criteriile de selectare ale acestor muchii, urmăresc eficiența algoritmului și în același timp îmbunătățirea turului curent:

-Criteriul de schimb secvențial: x_i și y_i sunt adiacente și la fel x_{i+1} și y_{i+1} . Dacă t_1 este una din extremitățile lui x_1 , atunci $x_i = t_{2i-1}t_{2i}$, $y_i = t_{2i}t_{2i+1}$ și $x_{i+1} = t_{2i+1}t_{2i+2}$.



Deci $x_1, y_1, x_2, y_2, x_3, \dots, x_r, y_r$ formează muchiile unui drum. Pentru **schimbul secvențial** se impune și ca acest drum să fie închis : $y_r = t_{2r}t_1$.

-**Criteriul de admisibilitate:** se va alege $x_i = t_{2i-1}t_{2i}$ astfel ca dacă t_{2i} se va uni cu t_1 , configurația care rezultă să fie tur. Se aplică pentru $i \geq 3$ și garantează că este posibil să închidem un tur. Este introdus în algoritm pentru reducerea timpului de execuție și simplificarea implementării.

-**Criteriul câștigului:** se cere ca y_i să fie ales astfel ca să existe un **câștig**; G_i trebuie să fie pozitiv; dacă $g_i = d(x_i) - d(y_i)$ este câștigul interschimbării lui x_i cu y_i , atunci $G_i = g_1 + g_2 + \dots + g_i$. Rațiunea alegerii acestui criteriu este că dacă suma unei secvențe de numere este pozitivă, atunci există o permutare circulară a acestei secvențe astfel ca orice sumă parțială este pozitivă.

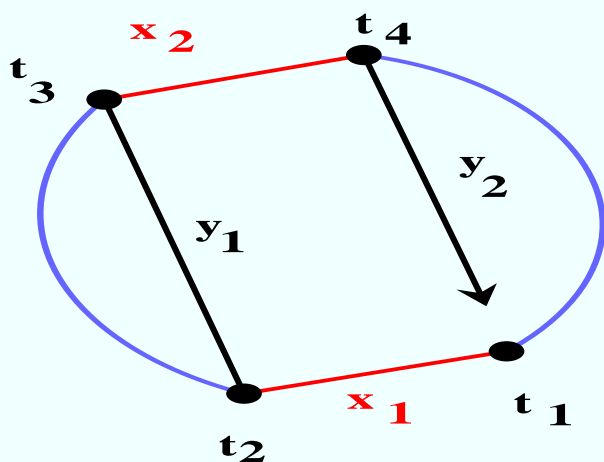
-**Criteriul dijunctivității:** se impune ca mulțimile X și Y să fie disjuncte. Aceasta simplifică implementarea, reduce timpul de execuție și dă un criteriu efectiv de oprire.

Linia algoritmului Lin & Kernighan

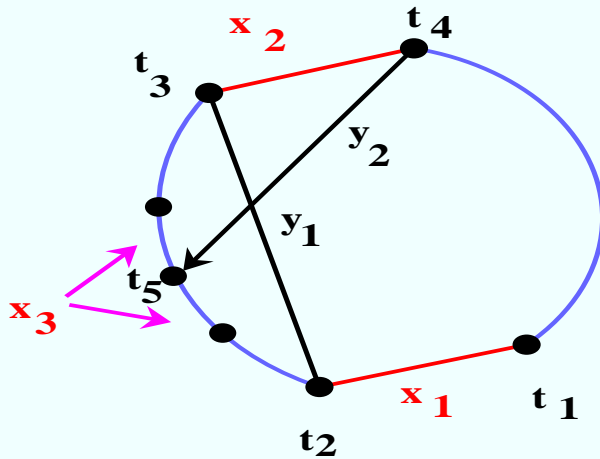
1. Generează (random) un tur inițial T ;
2. $i \leftarrow 1$; Alege t_1 ;
3. Alege $y_1 = t_1 t_2 \in T$;
4. Alege $y_1 = t_2 t_3 \notin T$ a.î. $G_1 > 0$; **If** \nexists **goto** 12;
5. $i \leftarrow i + 1$;
6. Alege $x_i = t_{2i-1} t_{2i} \in T$ a.î.
 - (a) dacă t_{2i} e unit cu t_1 se obține un tur T' și
 - (b) $x_i \neq y_s$ pentru toți $s < i$**If** T' e mai bun ca T **then** fie $T = T'$ și **goto** 2;
7. Alege $y_i = t_{2i} t_{2i+1} \notin T$ a.î.
 - (a) $G_i > 0$,
 - (b) $y_i \neq x_s$ pentru toți $s < i$ și
 - (c) x_{i+1} există.**If** y_i există **then goto** 5;
8. **If** mai \exists alte alegeri pt. y_2 **then** $i \leftarrow 2$ și **goto** 7;
9. **If** mai \exists alte alegeri pt. x_2 **then** $i \leftarrow 2$ și **goto** 6;
10. **If** mai \exists alte alegeri pt. y_1 **then** $i \leftarrow 1$ și **goto** 4;
11. **If** mai \exists alte alegeri pt. x_1 **then** $i \leftarrow 1$ și **goto** 3;
12. **If** mai \exists alte alegeri pt. t_1 **then goto** 2;
13. **Stop** (sau **goto** 1).

În descrierea anterioară "alege" înseamnă selectarea unei alternative neîncercate deja (pentru turul curent). De exemplu în pasul 3 avem două posibilități de a alege o muchie de pe tur incidentă cu t_1 .

În pasul 6 există două alegeri pentru x_i . Pentru un y_{i-1} ($i \geq 2$) numai una dintre acestea va face posibilă închiderea turului (prin adăugarea lui y_i). Cealaltă alegere va conduce la apariția a două subtururi disjuncte. Totuși pentru $i = 2$ este permisă o astfel de alegere neadmisibilă:

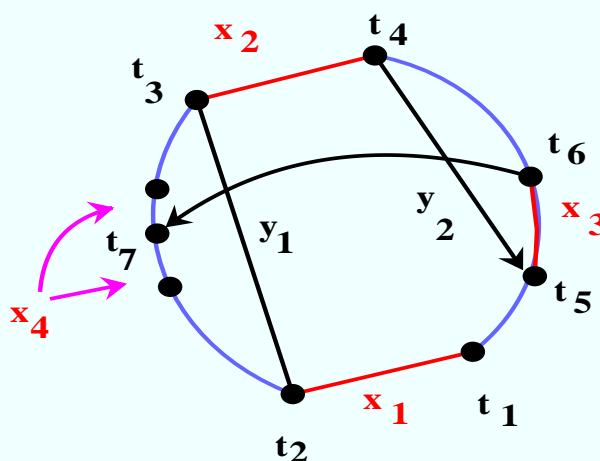


Dacă y_2 va fi ales astfel ca t_5 să fie între t_2 și t_3 , turul va putea fi închis în următorul pas.



Atunci, t_6 poate fi ales în ambele părți ale lui t_5 .

Dacă însă, y_2 va fi ales astfel ca t_5 să fie între t_4 și t_1 , atunci există doar o singură alegere pentru t_6 (între t_4 și t_5) și t_7 trebuie să se afle între t_2 și t_3 . Dar atunci t_8 trebuie să fie de cealaltă parte a lui t_7 ; algoritmul alege alternativa cu $d(t_7t_8)$ maximă.



Pașii 8-12 ai algoritmului cauzează backtracking-ul. El este permis numai dacă nu se obține o îmbunătățire și numai la nivelele 1 și 2.

Algoritmul se termină cu un tur după ce toate valorile posibile ale lui t_1 au fost examinate fără îmbunătățire.

Numeroase alte modificări ale metodei de bază au fost examinate. De exemplu, Lin și Kernighan limitează căutarea lui $y_i = t_{2i}t_{2i+1}$ la vecinii de pe tur ai lui t_{2i} la distanță cel mult 5.

Unele reguli au fost alese pentru salvarea timpului de execuție, **limitarea căutării**, iar altele pentru **direcționarea căutării**.

Se poate consulta articolul : **The Traveling Salesman Problem: A Case Study in Local Optimization** David S. Johnson Lyle A. McGeoch, *Local Search in Combinatorial Optimization*, E. H. L. Aarts and J. K. Lenstra (eds.), John Wiley and Sons, London, 1997, pp. 215-310.

O altă euristică populară pentru problemele în care funcția de distanță satisface inegalitatea triunghiulară, este dată de *Christofides*.

Spre deosebire de cazul general când am demonstrat că nu se poate spera la o euristică polynomială A cu R_A finită, dacă $\mathbf{P} \neq \mathbf{NP}$, în acest caz se poate demonstra următoarea

Teoremă. (*Christofides, 1973*) Fie CV cu d satisfăcînd

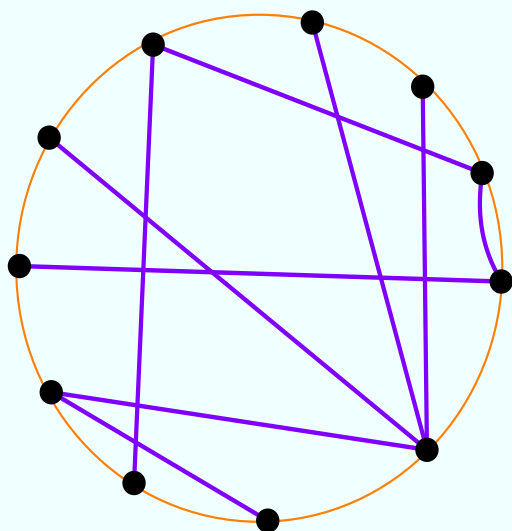
$$\forall v, w, u \in V(K_n) \text{ distincte } d(vw) \leq d(vu) + d(uw).$$

Există un algoritm aproximativ A pentru CV care satisface $R_A = \frac{3}{2}$ și are timp de lucru polynomial.

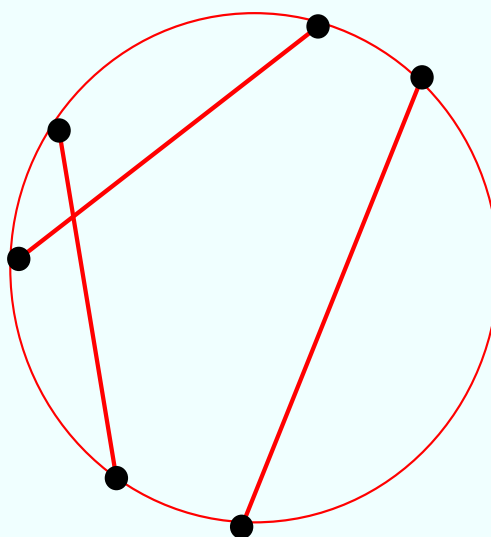
Demonstrație:

Considerăm următorul algoritm A :

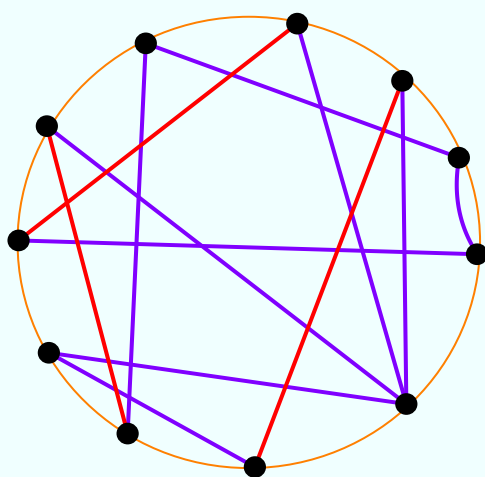
- 1⁰. Se determină T^0 mulțimea muchiilor unui arbore parțial de cost minim în K^n (costul muchiei e fiind $d(e)$). Problema se rezolvă în timp polinomial cu algoritmul lui Prim.
- 2⁰. Se determină M^0 un cuplaj perfect în subgraful indus de vîrfurile de grad impar ale arborelui T^0 și de cost minim. Problema se rezolvă în timpul $O(n^3)$ utilizînd algoritmul lui Edmonds.
- 3⁰. Se consideră multigraful obținut din $\langle T^0 \cup M^0 \rangle_{K_n}$, prin duplicarea muchiilor din $T^0 \cap M^0$. În acest multigraf există un parcurs Eulerian închis, ale cărui vîrfuri sînt $(v_{i_1}, v_{i_2}, \dots, v_{i_1})$. Eliminînd orice apariție multiplă a unui vîrf în acest șir cu excepția primului și ultimului vîrf, se obțin vîrfurile unui circuit hamiltonian H_A în K_n cu muchiile $H_A = (v_{j_1}v_{j_2}, v_{j_2}v_{j_3}, \dots, v_{j_n}v_{j_1})$ (Acest pas necesită $O(n^2)$ operații).



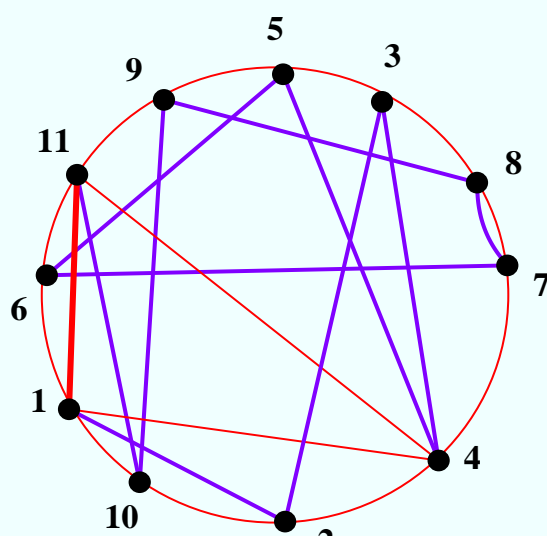
Arborele T^0



Cuplajul M^0



Graf Eulerian



Turul H_A

H_A este soluția aproximativă a problemei CV. Fie $m = \lfloor \frac{n}{2} \rfloor$ și H_0 soluția optimă. Vom arăta (cf. *Cornuejols și Nemhauser*) că

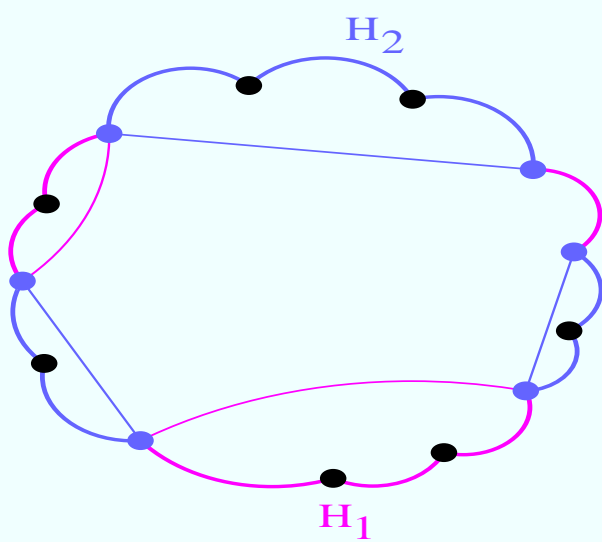
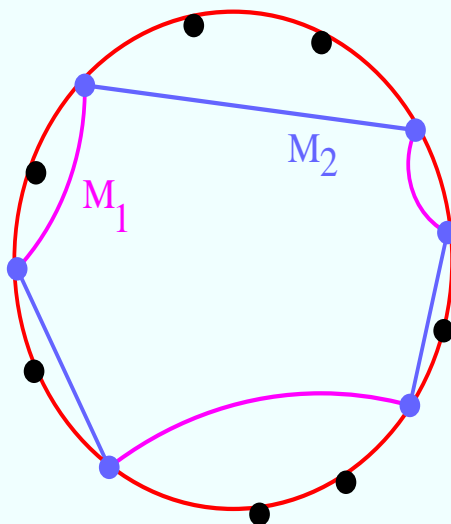
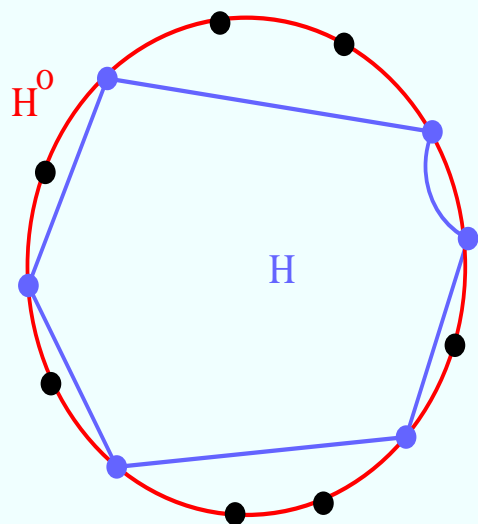
$$\forall n \geq 3 \quad d(H_A) \leq \frac{3m-1}{2m} d(H_0)$$

Presupunem că $H_0 = \{v_1v_2, v_2v_3, \dots, v_nv_1\}$ (eventual, schimbînd numerotarea vîrfurilor). Fie $A = \{v_{i_1}, v_{i_2}, \dots, v_{i_{2k}}\}$ vîrfurile de grad impar ale lui $\langle T^0 \rangle_{K_n}$, $i_1 < i_2 < \dots < i_{2k}$.

Dacă $H = \{v_{i_1}v_{i_2}, v_{i_2}v_{i_3}, \dots, v_{i_{2k-1}}v_{i_{2k}}, v_{i_{2k}}v_{i_1}\}$, avem din inegalitatea triunghiulară, $d(H) \leq d(H_0)$, prin înlocuirea ponderii fiecărei "corzi" $d(v_{i_j}v_{i_{j+1}})$ cu suma ponderilor muchiilor de pe circuitul H_0 subîntinse de $v_{i_j}v_{i_{j+1}}$.

Pe de altă parte, H este circuit de lungime pară, deci este reuniunea a două cuplaje perfecte în $[A]_{K_n}$, $M^1 \cup M^2$.

Presupunem că $d(M^1) \leq d(M^2)$.



Din alegerea lui M^0 , avem $d(M^0) \leq d(M^1) \leq \frac{1}{2}(d(M^1) + d(M^2)) = \frac{1}{2}d(H) \leq \frac{1}{2}d(H_0)$.

Fie $\alpha \in \mathbf{R}_+$ astfel încît $d(M^0) = \alpha d(H_0)$.

Evident, $0 < \alpha \leq \frac{1}{2}$.

Partiționăm H_0 în $H^1 \cup H^2$, punînd în H^i toate muchiile lui H_0 subîntinse de o coardă din M^i ($v_{i_j}v_{i_{j+1}} \in M^i \Rightarrow v_{i_j}v_{i_{j+1}}, \dots, v_{i_{j+1}-1}v_{i_{j+1}} \in H^i$).

Din inegalitatea triunghiulară obținem:

$$d(H^i) \geq d(M^i) \quad i = 1, 2.$$

Măcar unul din H^1 sau H^2 are cel mult $m = \lfloor \frac{n}{2} \rfloor$ muchii.

Presupunem că H^1 . Deoarece $d(H^1) \geq d(M^1) \geq d(M^0) = \alpha d(H_0)$, rezultă că există $e \in H^1 : d(e) \geq \frac{\alpha}{m}d(H_0)$.

Fie T arborele parțial obținut din H_0 prin înlătu-rarea unei muchii de pondere maximă. Avem

$$d(T) = d(H_0) - \max_{e \in H_0} d(e) \leq d(H_0) - \frac{\alpha}{m}d(H_0).$$

Cum T^0 este arbore parțial de cost minim în K_n , rezultă $d(T^0) \leq d(H_0)(1 - \frac{\alpha}{m})$.

Folosind inegalitatea triunghiulară,

$$d(H_A) \leq d(T^0) + d(M^0) \leq d(H_0)(1 - \frac{\alpha}{m}) + \alpha d(H_0) = (1 + \frac{\alpha(m-1)}{m})d(H_0).$$

Cum $\alpha \leq \frac{1}{2}$ se obține $d(H_A) \leq \frac{3m-1}{2m}d(H_0)$ pentru $n \geq 3$.

2. Metode care imită natura

În ultimii 30 de ani s-au dezvoltat numeroase metaeuristici inspirate din comportamentul unor sisteme biologice sau fizice ale lumii reale.

Metaeuristica *simulated annealing*.

Una din metaeuristicile populare utilizate pentru rezolvarea problemelor de optimizare NP-dificile este *călirea simulată*, metodă inspirată din termodinamică, inventată independent de *Kirkpatrick, Gelatt și Vechi* în 1983 și de *Černý* în 1985.

Ca orice metodă Monte Carlo, repetarea algoritmului cu diferiți parametri de start oferă șansa îmbunătățirii soluțiilor găsite, care nu sunt în general, și soluții optime.

În termodinamică , se poate interpreta călirea (intilnită tradițional în prelucrarea metalelor în fierăriile cu forjă , baros și nicovală) ca un proces stohastic ce determină o aranjare a atomilor care minimizează energia totală a unui corp.

La temperaturi înalte, atomii se mișcă liber și se mută cu rapiditate în poziții care cresc energia totală. Pe măsură ce temperatura este scăzută, atomii se apropie gradual de o dispunere laticială regulată și numai ocazional își măresc energia. Aceste creșteri ocazionale de energie joacă un rol crucial în călire : ele permit ieșirea din minimele locale printr-o creștere temporară de energie.

La temperaturi înalte, astfel de salturi apar cu mare probabilitate, iar la temperaturi joase ele apar rar. Temperatura este scăzută lent pentru a menține echilibrul termal. Când atomii sunt în echilibru la temperatura T , probabilitatea ca energia lor totală să fie E este proporțională cu $e^{-\frac{E}{kT}}$, unde k este constanta lui *Bolzm*ann.

În consecință, probabilitatea ca energia să fie $E + dE$ poate fi exprimată,

$$Pr(E + dE) = Pr(E) \cdot e^{-\frac{dE}{kT}},$$

adică probabilitatea de creștere a energiei scade odată cu temperatura.

Călirea simulată este o metodă computațională care imită modul natural de determinare a unei configurații care minimizează energia unui sistem. Atunci cnd se dorește minimizarea unei funcții $f : D \rightarrow \mathbf{R}$, vom interpreta domeniul de definiție al funcției, D , ca fiind mulțimea configurațiilor posibile ale sistemului, iar funcția f ca fiind energia acestuia.

O variabilă fictivă T , asociată procesului de căutare, va juca rolul temperaturii iar constanta lui Boltzmann va fi considerată 1.

Algoritm de călire simulată

1. Se consideră un **plan de călire**:

- temperatura inițială T_{start}
- configurația inițială $x_{start} \in D$
- temperatura finală T_{min}
- o funcție de reducere lentă a temperaturii $decrease(T)$
- nr. maxim de încercări de îmbunătățire a soluției la fiecare prag de temperatură $attempts$
- nr. maxim de schimbări ale soluției la fiecare prag de temperatură $changes$.

2. $T \leftarrow T_{start}; x_{old} \leftarrow x_{start}$

while $T > T_{min}$ **do**

{ $na \leftarrow 0; nc \leftarrow 0$

while $na < attempts$ **and** $nc < changes$ **do**

{ generează o soluție nouă x_{new} ; $na++$;

$\Delta E \leftarrow f(x_{old}) - f(x_{new})$;

If $\Delta E > 0$ **then** { $x_{old} \leftarrow x_{new}$; $nc++$ }

else

{ generează $q \in (0, 1)$ un nr. aleator

if $q < e^{-\frac{\Delta E}{T}}$ **then** { $x_{old} \leftarrow x_{new}$; $nc++$ }

}

}

$decrease(T)$

}

3. **return** x_{old}

Planul de călire se stabilește adeseori prin experimente (*tuning*) asupra clasei de probleme la care se aplică. Modul în care se permite, la un anumit nivel de temperatură, ca unele soluții noi care nu micșorează valoarea funcției să fie considerate (cu scopul părăsirii minimelor locale) poartă denumirea inventatorului : *schema Metropolis*.

Pentru aplicarea călirii simulate pentru rezolvarea problemei comisului voiajor, se pornește cu un tur ales aleator, iar trecerea de la o soluție curentă la o soluție vecină în spațiul soluțiilor se realizează de obicei cu ajutorul unei **2-move**.

Se poate lua $T_{start} = O(\sqrt{n})$, $attempts = 100n$, $changes = 10n$, $decrease(T) = 0.95T$ și $T_{min} = O(1)$.

VIII. GRAFURI PLANARE

1. Proprietăți de bază ale grafurilor planare

Definiție. Fie $G = (V, E)$ un graf și S o suprafață în \mathbb{R}^3 . Spunem că G este **reprezentabil pe S** dacă există $G' = (V', E')$ un graf astfel încât:

- a) $G \cong G'$.
- b) V' e o mulțime de puncte distincte din S .
- c) Orice muchie $e' \in E'$ este o curbă simplă conținută în S care unește cele două extremități.
- d) Orice punct al lui S este sau vîrf al lui G' , sau prin el trece cel mult o muchie a lui G' .

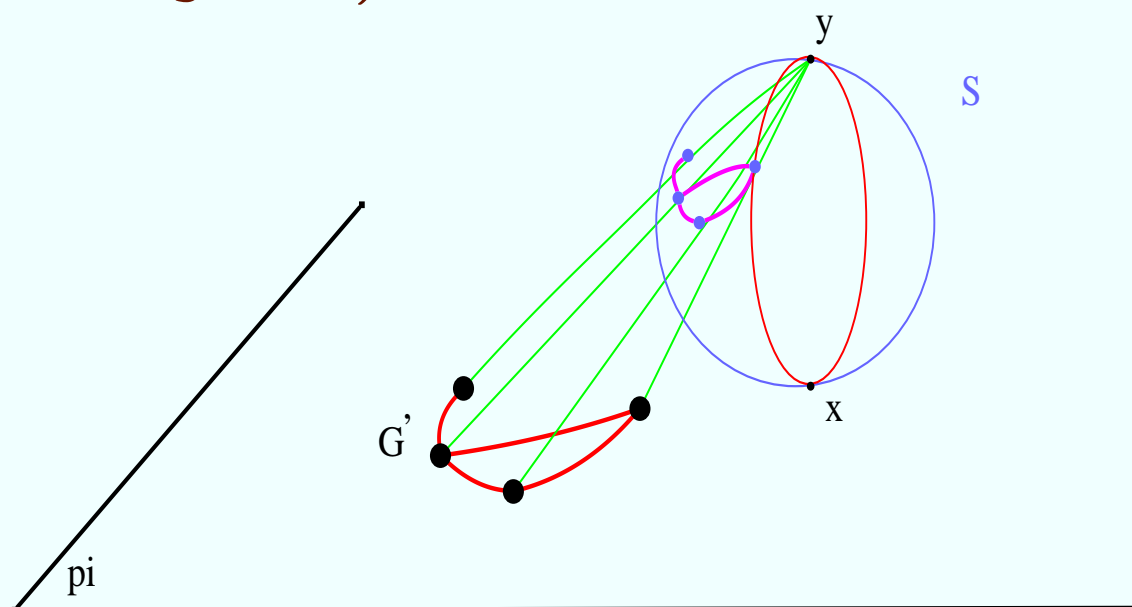
G' se numește **reprezentare a lui G în S** .

Dacă S este un plan atunci G se numește **planar** iar G' o **reprezentare planară** a lui G .

Dacă S este un plan și G' este un graf care satisface b) c) și d) de mai sus atunci G' se numește **graf plan**.

Lemă. 1. *Un graf este planar dacă și numai dacă este reprezentabil pe o sferă.*

Demonstrație. Fie G planar și G' o reprezentare planară a sa în planul π . Considerăm un punct x al lui π și S o sferă tangentă la π în x . Fie y punctul diametral opus al lui x pe sferă. Definim $\phi : \pi \rightarrow S$ considerînd pentru orice punct M al planului π , $\phi(M)$ cel de-al doilea punct de intersecție al dreptei My cu sfera S . Evident ϕ este o bijecție și deci $\phi(G')$ este reprezentarea lui G pe sferă (ϕ este *proiecția stereografică*).



Reciproc, dacă G este reprezentabil pe o sferă S : se alege un punct y pe sferă, se consideră punctul x diametral opus lui y pe sferă, se construiește un plan tangent π sferei S în punctul x și se definește $\psi : S \rightarrow \pi$ considerînd pentru orice punct M al sferei, $\psi(M)$ intersecția dreptei yM cu planul π . Imaginea prin ψ a reprezentării lui G pe sferă va fi o reprezentare planară a lui G .

Definiție. Fie G un graf plan. Dacă îndepărtăm punctele lui G (vîrfurile și muchiile sale) din plan se obține o reuniune de regiuni conexe (orice două puncte se pot uni printr-o curbă simplă conținută în regiune) ale planului, care se numesc **fețele** lui G .

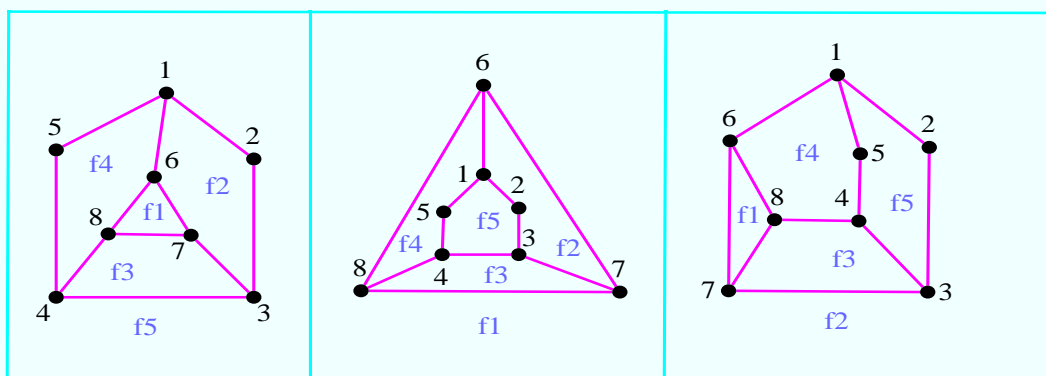
Evident, orice graf plan are un număr finit de fețe, dintre care una singură este nemărginită și se numește **față exterioară** a lui G .

Frontiera unei fețe este mulțimea muchiilor lui G conținute în închiderea acestei fețe.

Orice circuit al lui G împarte punctele planului în două regiuni conexe, deci orice muchie a unui circuit al grafului G se află în frontierele a două fețe.

Un graf plan, împreună cu mulțimea fețelor sale se numește **hartă planară** (uneori, în această definiție se impune ca graful să fie 2-conex; atunci fețele hărții planare se numesc țări și două țări sînt vecine dacă frontierele lor au măcar o muchie comună; aceasta corespunde aspectelor intuitive de la hărțile geografice).

Unui graf planar i se pot asocia reprezentări planare diferite. În figura alăturată sînt desenate trei reprezentări planare diferite ale aceluiași graf:



Un instrument util în caracterizarea și recunoașterea grafurilor planare este următoarea observație:

Lemă. 2. *Orice reprezentare planară a unui graf poate fi transformată într-o reprezentare diferită astfel încât o față specificată a sa să devină față exterioară.*

Demonstrație: Fie G' o reprezentare planară a lui G și F o față a lui G' . Considerăm G^0 o reprezentare pe sferă a lui G' (construită, de exemplu, ca în lema 1) și fie F^0 fața corespunzătoare a lui F (proiecția stereografică transformă orice față tot într-o față). Alegem un punct y din interiorul feței F^0 , considerăm x punctul diametral opus pe sferă lui y și efectuăm proiecția stereografică ϕ pe planul tangent sferei în punctul x . Graful plan G'' astfel obținut va avea față exterioară $\phi(F^0)$ care are aceeași frontieră cu fața F din reprezentarea G' .

Este aproape evident că orice două reprezentări planare ale aceluși graf au același număr de fețe. Acest lucru este mai bine precizat de următoarea teoremă.

Teoremă. 1. (Formula lui Euler) *Fie $G = (V, E)$ un graf plan conex cu n vîrfuri, m muchii și f fețe. Atunci*

$$f = m - n + 2$$

Demonstrație: Inducție după numărul fețelor lui G . Dacă $f = 1$, atunci G nu are circuite și cum este și conex, rezultă că G este arbore, deci $m = n - 1$ și prin urmare teorema are loc.

În pasul inductiv, presupunem teorema adevărată pentru orice graf plan și conex cu mai puțin de $f(\geq 2)$ fețe. Există măcar o muchie e , care nu este punte (altminteri G ar fi arbore și ar avea o singură față). Rezultă că e aparține frontierei a exact două fețe ale lui G . Considerăm $G_1 = G - e$. Din alegerea lui e , G_1 este conex. Evident G_1 este graf plan.

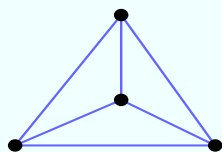
Numărul fețelor lui G_1 este $f_1 = f - 1$ (cele două fețe ale lui G vor genera în G_1 o singură față); în plus G_1 are $n_1 = n$ vîrfuri și $m_1 = m - 1$ muchii.

Pentru G_1 are loc ipoteza inductivă, deci $f_1 = m_1 - n_1 + 2$.

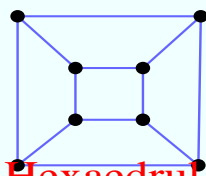
Substituind valorile lui f_1, m_1 și n_1 se obține $f - 1 = (m - 1) - n + 2$, adică $f = m - n + 2$.

Observații: 1^0 Oricărui poliedru convex din \mathbb{R}^3 i se poate asocia un graf planar cu același număr de vîrfuri și muchii (se reprezintă vîrfurile și muchiile poliedrului pe o sferă și se execută proiecția stereografică).

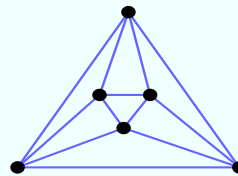
Figura următoare conține grafurile planare asociate poliedrelor platonice:



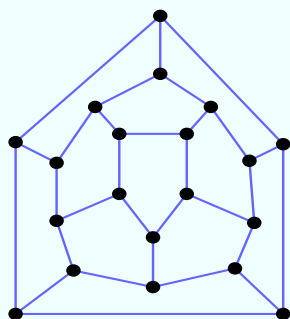
Tetraedrul



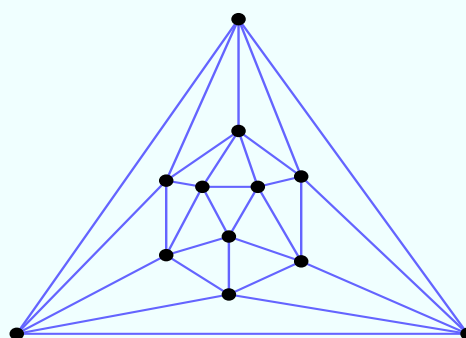
Hexaedrul
(Cubul)



Octoedrul



Dodecaedrul



Icosaedrul

Steinitz și Rademacher (1934) au arătat că un graf planar este graful asociat unui poliedru convex dacă și numai dacă este 3-conex. Relația din teorema 1, este cunoscută ca **formula poliedrală a lui Euler** și precizează numărul fețelor poliedrului convex în funcție de numărul vîrfurilor și muchiilor sale.

2⁰. Din punct de vedere algoritmic, teorema 1 are drept consecință imediată faptul că orice graf planar este "rar", numărul muchiilor este de ordinul numărului de vîrfuri. Va rezulta că orice traversare în ordinul $O(|V| + |E|)$ a lui G este de fapt în $O(|V|)$ operații.

Corolar. 1. *Fie G un graf planar, conex, cu $n(\geq 3)$ vîrfuri și $m > 2$ muchii. Atunci*

$$m \leq 3n - 6.$$

Demonstrație. Fie G' o reprezentare planară a lui G . Dacă G' are o singură față, atunci G' este arbore, $m = n - 1$ (G' are același număr de vîrfuri și muchii ca și G), și pentru $n \geq 3$ inegalitatea are loc. Dacă G' are măcar două fețe, atunci fiecare față F are în frontieră muchiile unui circuit $C(F)$, și fiecare astfel de muchie aparține la exact două fețe. Orice circuit al grafului are măcar 3 muchii, deci

$$2m \geq$$

$$\sum_F \text{față în } G' (\text{nrul muchiilor circ. } C(F)) \geq$$

$$\sum_F 3 = 3f = 3(m - n + 2),$$

inegalitate, evident, echivalentă cu cea din enunț.

Corolar. 2. *Dacă G este un graf bipartit, conex și planar cu $m > 2$ muchii și n vîrfuri, atunci $m \leq 2n - 4$.*

Demonstrație: Același raționament ca mai sus, cu observația că orice circuit are măcar 4 muchii pentru că, graful fiind bipartit, numărul muchiilor oricărui circuit este par.

Corolar. 3. *Dacă G este un graf planar conex, atunci G are un vîrf de grad cel mult 5.*

Demonstrație: Dacă G are mai mult de 2 muchii aplicăm consecința 1 astfel: fie G' o reprezentare planară a lui G cu n vîrfuri și m muchii; notăm cu n_i numărul vîrfurilor de grad i ($1 \leq i \leq n-1$) din G' ; atunci

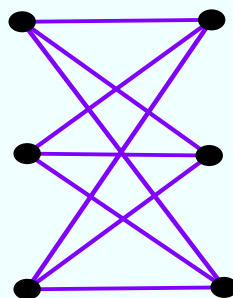
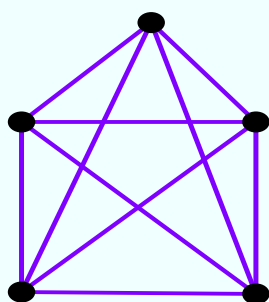
$$\sum_{i=1}^{n-1} i \cdot n_i = 2m \leq 2(3n-6) = 6(\sum_i n_i) - 12,$$

deci

$$\sum_i (i-6)n_i + 12 \leq 0$$

Cum pentru $i \geq 6$ toți termenii sumei sînt nenegativi, rezultă că există i_0 , $1 \leq i_0 \leq 5$, astfel încît $n_{i_0} > 0$, ceea ce trebuia demonstrat.

Observație: Grafurile K_5 și $K_{3,3}$ nu sînt planare.



Primul graf violează inegalitatea din corolarul 1, cel de al doilea pe cea din corolarul 2.

Fie $G = (V, E)$ un graf și $v \in V(G)$ astfel încât $d_G(v) = 2$ și $vw_1, vw_2 \in E$, $w_1 \neq w_2$. Considerăm $h(G) = (V \setminus \{v\}, E \setminus \{vw_1, vw_2\} \cup \{w_1w_2\})$. Se observă că G este planar dacă și numai dacă $h(G)$ este planar.

(Dacă $w_1w_2 \notin E(G)$ atunci pe curba simplă ce unește w_1 și w_2 într-o reprezentare planară a lui $h(G)$ se introduce un nou vîrf v ; dacă $w_1w_2 \in E(G)$, în una din fețele reprezentării planare a lui $h(G)$ se plasează "suficient" de aproape de curba w_1w_2 un vîrf nou v și se "unește" cu w_1 și w_2 . Reciproc, în reprezentarea planară a lui G ștergem punctul v și cele două muchii vw_1 și vw_2 le înlocuim cu reuniunea lor. Dacă $w_1w_2 \in E(G)$ atunci se șterge și curba w_1w_2).

Vom nota cu $h^*(G)$ graful obținut din G aplicîndu-i repetat transformarea h , pînă cînd graful curent nu mai are vîrfuri de grad 2. Din observația anterioară rezultă că G este planar, dacă și numai dacă $h^*(G)$ este planar.

Definiție. Două grafuri G_1 și G_2 se numesc *homeomorfe* dacă și numai dacă $h^*(G_1) \cong h^*(G_2)$.

Teoremă. 2. (Kuratowski 1930) *Un graf este planar dacă și numai dacă nu are subgrafuri homeomorfe cu K_5 sau $K_{3,3}$.*

Necesitatea teoremei este evidentă: dacă un graf este planar atunci orice subgraf al său este planar. Dacă ar exista un subgraf G' al lui G homeomorf cu $K_{3,3}$ sau K_5 , cum aceste două grafuri am arătat că nu sînt planare și cum $h^*(G')$ este planar dacă și numai dacă G' este planar rezultă că G' și deci G nu este planar, contrazicînd alegerea lui G .

Suficiența se demonstrează prin inducție după numărul de muchii.

2. Desenarea unui graf planar

Fie G un graf planar. Se pune problema trasării lui în plan. Este aproape evident că se pot aproxima curbele simple ce unesc vîrfurile, cu linii poligonale, astfel încît să nu se violeze condiția de planaritate. Are loc, totuși, un rezultat mai tare demonstrat de *Fary* în 1948 (și independent de *Wagner* și *Stein*):

Orice graf planar are o reprezentare planară cu toate muchiile segmente de dreaptă (reprezentarea Fary).

Această teoremă a deschis o serie de rezultate referitoare la desenarea grafurilor planare. Pentru un mediu grafic, care permite poziționarea în puncte de coordonate întregi și trasarea de segmente ce unesc aceste puncte, este desigur important să se demonstreze existența unei reprezentări Fary cu **vîrfuri în puncte de coordonate întregi**, și în același timp aria suprafeței ocupate de reprezentare să fie polinomială în raport cu numărul n de vîrfuri ale grafului.

Descriem unul dintre rezultatele de acest tip:

Teoremă. 3. (*Fraysseseix, Pach, Pollack (1988)*)
Orice graf planar cu n vîrfuri are o reprezentare planară cu vîrfuri de coordonate întregi în $[0, 2n - 4] \times [0, n - 2]$ și cu muchii segmente de dreaptă.

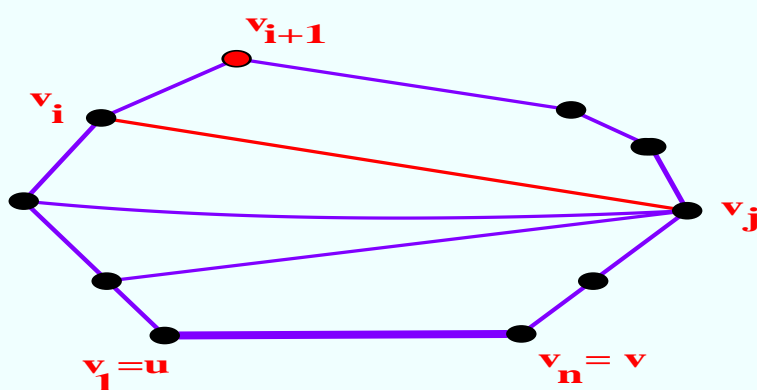
Notăm că demonstrația care urmează poate fi baza unui algoritm de complexitate $O(n \log n)$ pentru obținerea acestei reprezentări.

Lemă. 3. *Fie G un graf planar și G' o reprezentare planară a sa. Dacă C' este un circuit al lui G' ce trece prin muchia $uv \in E(G')$ atunci există $w \in V(C')$ astfel încît $w \neq u, v$ și w nu este extremitatea nici unei corzi interioare a lui C' .*

Demonstrație: Fie v_1, v_2, \dots, v_n vîrfurile circuitului C' într-o parcurgere a sa de la u la v ($v_1 = u, v_n = v$). Dacă C' nu are corzi interioare lema este demonstrată. Altfel, alegem perechea (i, j) astfel încît $v_i v_j$ este coardă interioară a lui C' și

$j-i = \min\{k-l \mid k > l+1, v_k v_l \in E(G'), v_k v_l \text{ coardă interioară} \}$.

Atunci v_{i+1} nu e incident cu nici o coardă interioară ($v_{i+1} v_p$ cu $i+1 < p < j$ nu e coardă interioară căci s-ar contrazice alegerea perechii (i, j) și $v_{i+1} v_l$ cu $l < i$ sau $l > j$, nu este coardă interioară datorită existenței coardei $v_i v_j$ și a planarității).



Vom demonstra teorema 4 în ipoteza suplimentară că G este **maximal planar** : orice muchie i s-ar adăuga se obține un graf neplanar (sau multigraf).

Să observăm că orice față a unui graf maximal planar este un C_3 (altminteri în reprezentarea lui G cu fața exterioară mărginită de un C_n cu $n \geq 4$ se pot introduce muchii fără a pierde planaritatea grafului).

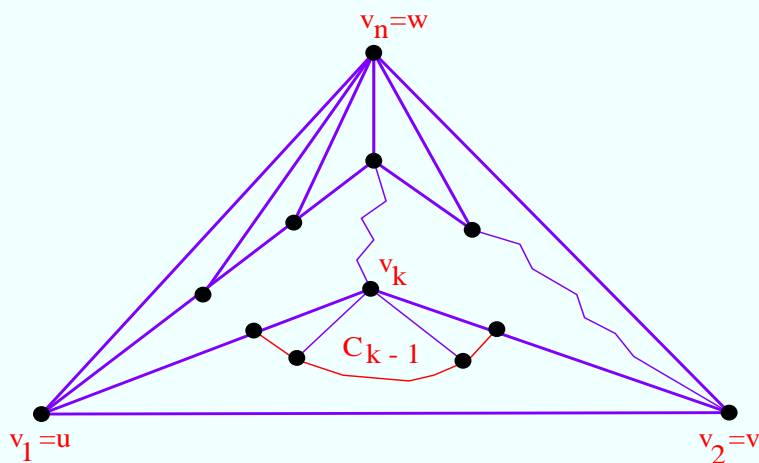
Ipoteza nu este restrictivă: de la o reprezentare a lui G ca o hartă planară (ce se obține aplicînd de exemplu algoritmul de testare a planarității) se trece la o hartă cu toate fețele triunghiului prin inserția în timp liniar de corzi în circuite. La desenarea grafului obținut, muchiile fictive introduse nu se vor trasa.

Lemă. 4. *Fie G maximal planar cu $n \geq 4$ vîrfuri și G' o reprezentare planară a sa cu fața exterioară u, v, w . Există o etichetare a vîrfurilor lui G' : v_1, v_2, \dots, v_n astfel încît $v_1 = u, v_2 = v, v_n = w$ și pentru orice $k, 4 \leq k \leq n$ avem:*

(i) subgraful $G'_{k-1} = [\{v_1, \dots, v_{k-1}\}]_{G'}$, este 2-conex și fața sa exterioară este determinată de circuitul C'_{k-1} ce conține uv .

(ii) în subgraful G'_k , vârful v_k este în fața exterioară a lui G'_{k-1} și $N_{G'_k}(v_k) \cap \{v_1, \dots, v_{k-1}\}$ este un drum de lungime ≥ 1 de pe circuitul $C'_{k-1} \setminus uv$.

Demonstrație: Fie $v_1 = u, v_2 = v, v_n = w, G'_n = G, G'_{n-1} = G - v_n$. Să observăm că $N_{G'_n}(v_n)$ este un circuit ce conține uv (este suficient, pentru a demonstra aceasta să ordonăm $N_{G'_n}(w)$ după abscisele vîrfurilor și să folosim planaritatea maximală).



Rezultă că pentru $k = n$ (i) și (ii) au loc.

Dacă v_k a fost ales ($k \leq n$) atunci în $G'_{k-1} = G' - \{v_n, \dots, v_k\}$ vecinii lui v_k determină un circuit C'_{k-1} ce conține uv și mărginește fața exterioară a lui G'_{k-1} . Din lema 3, rezultă că există v_{k-1} pe C'_{k-1} astfel încât v_{k-1} nu este incident cu o coardă interioară a lui C'_{k-1} .

Din construcție, rezultă că v_{k-1} nu este incident nici cu corzi exterioare lui C'_{k-1} și cu această alegere, se observă că G'_{k-2} va conține un circuit C'_{k-2} cu proprietățile (i) și (ii).

Notăm că etichetarea precizată în lema 4 se poate construi în $O(n)$ considerînd o reprezentare a lui G cu liste de adiacență ordonate circular, prin fixarea vîrfurilor v_1, v_2 și v_n și apoi considerînd pentru fiecare k $k \geq 3$ un vîrf v_k cu proprietatea că arcele cu o extremitate v_k și cealaltă în $\{v_1, \dots, v_{k-1}\}$ formează un segment continuu în lista circulară a lui $A(v_k)$ (existența unui asemenea vîrf fiind asigurată de lema 4).

Demonstrația teoremei 4. Așa cum am mai observat, se poate presupune G maximal planar cu $n(\geq 4)$ vîrfuri. Fie G' o reprezentare planară a lui G , cu vîrfurile v_1, v_2, \dots, v_n , fața exterioară u, v, w și etichetarea vîrfurilor satisface condițiile din lema 4.

Vom construi o reprezentare Fary a lui G cu vîrfurile puncte de coordonate întregi.

În pasul $k(\geq 3)$ al construcției dispunem de o astfel de reprezentare a lui G_k , și în plus sînt satisfăcute condițiile:

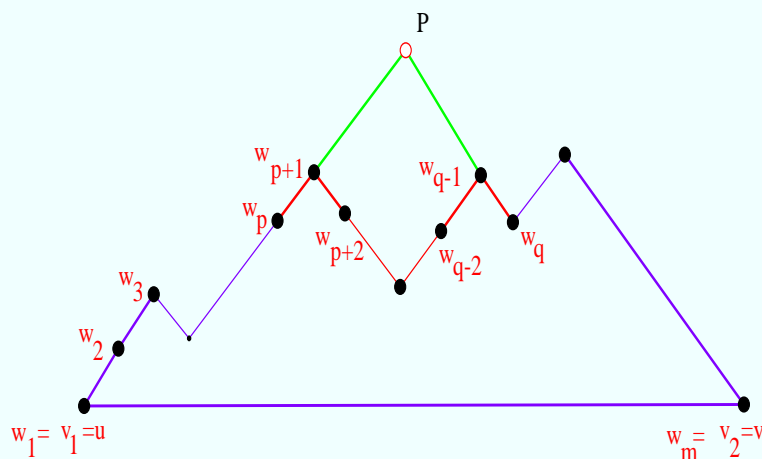
- (1) v_1 are coordonatele $x_{v_1} = 0$ și $y_{v_1} = 0$; v_2 are coordonatele $x_{v_2} = i \leq 2k - 4$, $y_{v_2} = 0$;
- (2) Dacă w_1, w_2, \dots, w_m sînt vîrfurile circuitului ce mărginește fața exterioară a lui G_k în ordinea parcurgerii lor de la v_1 la v_2 ($w_1 = v_1, w_m = v_2$) atunci

$$x_{w_1} < x_{w_2} < \dots < x_{w_m}.$$

(3) Muchiile $w_1w_2, w_2w_3, \dots, w_{m-1}w_m$ sînt segmente de dreaptă paralele cu prima sau a doua bisectoare a axelor de coordonate.

Condiția (3) asigură că $\forall i < j$ dacă prin w_i se duce o paralelă la prima bisectoare și prin w_j o paralelă la a doua bisectoare, **intersecția lor este un punct de coordonate întregi.**

Construim G'_{k+1} . Fie w_p, w_{p+1}, \dots, w_q vecinii lui v_{k+1} în G'_{k+1} (cf. lemei 4) $1 \leq p < q \leq m$.



Paralela prin w_p la prima bisectoare intersectează paralela prin w_q la a doua bisectoare, într-un punct P . Dacă din punctul P se pot trasa segmentele Pw_i $p \leq i \leq q$ astfel încît ele să fie distincte, atunci putem lua $v_{k+1} = P$ și obținem reprezentarea Fary pentru G_{k+1} cu vîrfuri de coordonate întregi, satisfăcînd condițiile (1) (2) și (3).

Dacă segmentul $w_p w_{p+1}$ este paralel cu prima bisectoare, atunci translăm cu o unitate la dreapta toate vîrfurile grafului G_k , care au abscisa $\geq x_{w_{p+1}}$. Efectuăm apoi o translație cu o unitate la dreapta a tuturor vîrfurilor cu abscisa $\geq x_{w_q}$. Să observăm că acum toate segmentele $P'w_i$ cu $p \leq i \leq q$ sînt distincte, segmentele $w_i w_{i+1}$ cu $i = \overline{q, m-1}$ au pantele ± 1 și de asemenea $w_p P'$ și $P'w_q$ (unde P' este punctul obținut prin intersecția paralelelor la prima (respectiv a doua) bisectoare, duse prin w_p și w_q). Luăm $v_{k+1} = P'$ și pasul k al construcției este terminat.

Condițiile (1)- (3) sînt evident satisfăcute. Cu aceasta teorema este demonstrată.

Să observăm că determinarea vîrfurilor care se vor transla în pasul k , poate fi organizată astfel încît să nu necesite în total $O(n^2)$ operații ci numai $O(n \log n)$, utilizînd structuri convenabile de date.

3. Grafuri plane - versiunea combinatorială.

Vom considera în cele ce urmează doar grafuri conexe cu măcar două vârfuri.

În versiunea combinatorială un graf este un triplet $G = (E, \theta, -)$, unde E este o mulțime de cardinal par, $-$ este o *involuție* pe E (permutare de ordin 2) fără puncte fixe, și θ este o permutare pe E .

Elementele lui E sunt gândite ca *arce*; o muchie (neorientată) este reprezentată ca o pereche $e, \bar{e} \in E$ de arce, inverse unul altuia. Aplicația $\bar{}$ inversează direcția.

Se dorește ca aplicația θ să dea o **orientare a muchiilor din jurul unui vârf** (în sens contrar acelor de ceasornic).

Vârfurile sunt *ciclii* permutării θ . (Un ciclu al permutării θ este o submulțime nevidă a lui E închisă în raport cu θ și minimală cu această proprietate).

Dacă notăm cu V mulțimea ciclilor permutării θ atunci definim

$t : E \rightarrow V$ prin $t(e) =$ **unicul ciclu al lui θ ce conține e (extremitatea inițială a arcului e)**

$h : E \rightarrow V$ prin $h(e) =$ **unicul ciclu al lui θ ce conține \bar{e} (extremitatea finală a arcului e)**

Se observă că $\forall e \ t(e) = h(\bar{e})$ și $h(e) = t(\bar{e})$.

Dacă vom considera permutarea $\theta^* : E \rightarrow E$ definită de $\theta^*(e) = \theta(\bar{e})$, atunci o **față** a lui G este un ciclu al permutării θ^* .

Intuitiv, pentru a calcula $\theta^*(e)$, inversăm e pentru a obține \bar{e} și apoi ne rotim (în sensul acelor de ceasornic) în jurul extremității inițiale a lui \bar{e} . Numărul fețelor lui G se notează cu f .

O **componentă conexă** a lui G este o *orbită* a lui E în grupul de permutări generat de θ și $\bar{\cdot}$: **o mulțime nevidă minimală cu proprietatea că este închisă la θ și $\bar{\cdot}$.**

Fie G un graf cu $m = \frac{1}{2}|E|$ muchii (neorientate), $n = |V|$ vârfuri, f fețe, și c componente conexe. **Caracteristica Euler** a lui G se definește ca fiind

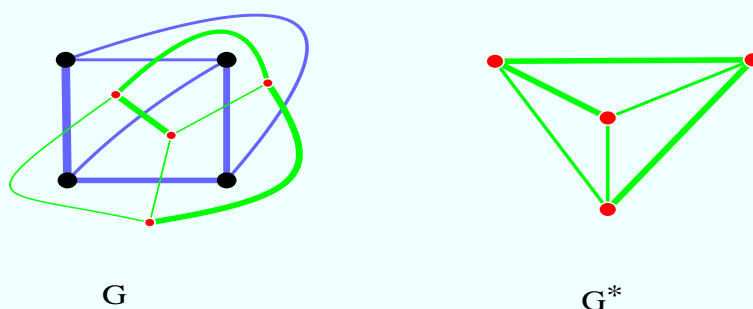
$$\chi(G) = 2c + m - n - f.$$

Un graf G se numește **graf plan** dacă $\chi(G) = 0$.

Se poate demonstra că pentru un graf conex în definiția tradițională, cele două noțiuni de grafuri plane coincid (graful neorientat construit așa cum am descris mai sus atașat unui graf în formă combinatorială este graf plan conform definiției tradiționale și invers, dacă pentru un graf tradițional plan conex se construiește θ conform unei orientări inverse acelor de ceasornic a muchiilor și $\bar{\cdot}$ corespunzătoare, graful combinatorial obținut este plan în noua definiție).

4. Teorema separatorului

Definiție Dacă $G = (V, E)$ este un graf plan maximal (cu toate fețele triunghiuri) atunci **dualul** său este graful $G^* = (V^*, E)$ unde V^* sunt fețele grafului G iar muchiile lui G^* se obțin din muchiile grafului G , asociindu-i fiecărei muchii $e \in E$ cele două fețe în frontiera cărora se află e .



Lemă. 1. Fie $G = (V, E)$ este un graf plan maximal conex , $G^* = (V^*, E)$ dualul său și $E' \subseteq E$. Atunci (V, E') este arbore parțial al lui G dacă și numai dacă $(V^*, E - E')$ este arbore parțial al lui G^* .

Teoremă. 1. (*Tarjan & Lipton, 1979*)

Fie G un graf planar cu n vârfuri. Există o partiție a lui $V(G)$ în clasele disjuncte A, B, S astfel încât:

- 1. S separă A de B în G : $G - S$ nu are muchii cu o extremitate în A și cealaltă în B .*
- 2. $|A| \leq \frac{2}{3}n$, $|B| \leq \frac{2}{3}n$.*
- 3. $|S| \leq 4\sqrt{n}$.*

Această partiție se poate afla în timpul $O(n)$.

Demonstrație. Considerăm graful conex și de asemenea considerăm că dispunem de o reprezentare planară (obținute cu un algoritm liniar).

Alegem un vârf s și executăm o parcurgere bfs din s numerotând vârfurile (în ordinea întâlnirii lor în această parcurgere) și atribuind fiecărui vârf v nivelul său în arborele bfs construit.

Vom nota cu $L(t)$, $0 \leq t \leq l + 1$ mulțimea vârfurilor de pe nivelul t (nivelul $l + 1$ va fi introdus în scopuri tehnice și este vid, ultimul nivel este de fapt l).

Evident, fiecare nivel este un separator în G (avem muchii doar între nivele consecutive).

Fie t_1 nivelul de la mijloc, adică nivelul ce conține vârful numerotat bfs cu numărul de ordine $\frac{n}{2}$. Mulțimea $L(t_1)$ are o parte din proprietățile separatorului pe care îl căutăm:

$$\left| \bigcup_{t < t_1} L(t) \right| < \frac{n}{2} \quad \left| \bigcup_{t > t_1} L(t) \right| < \frac{n}{2}.$$

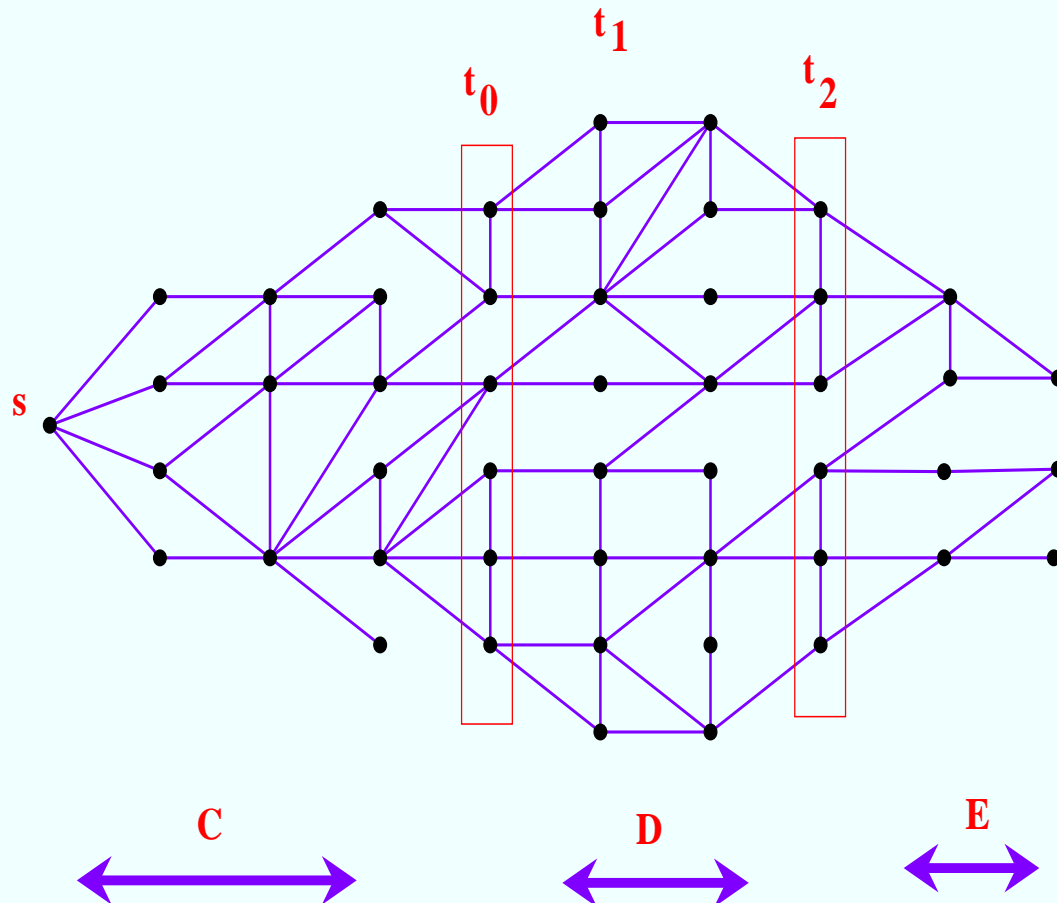
Dacă și $L(t_1) \leq 4\sqrt{n}$, teorema are loc.

Lemă. 2. *Există nivelele $t_0 \leq t_1$ și $t_2 > t_1$ a. încât $|L(t_0)| \leq \sqrt{n}$, $|L(t_2)| \leq \sqrt{n}$ și $t_2 - t_0 \leq \sqrt{n}$.*

Se alege t_0 cel mai mare număr cu proprietățile $t_0 \leq t_1$ și $|L(t_0)| \leq \sqrt{n}$ (există un astfel de nivel pentru că $|L(0)| = 1$). La fel, există t_2 un cel mai mic număr astfel încât $t_2 > t_1$ și $|L(t_2)| \leq \sqrt{n}$ (de aceea s-a luat $|L(l + 1)| = 0$). Orice nivel strict între t_0 și t_2 are mai mult de \sqrt{n} vârfuri deci numărul acestor nivele este mai mic decât \sqrt{n} , altfel am avea mai mult de n vârfuri în graf.

Considerăm

$$C = \cup_{t < t_0} L(t), \quad D = \cup_{t_0 < t < t_2} L(t), \quad E = \cup_{t > t_2} L(t).$$



Dacă $|D| \leq \frac{2}{3}n$ atunci teorema are loc cu $S = L(t_0) \cup L(t_2)$, A mulțimea cu cele mai multe elemente dintre C, D, E și B reuniunea celorlalte două (nu uităm că C și E au cel mult $\frac{n}{2}$ elemente).

Considerăm deci că $n_1 = |D| > \frac{2}{3}n$.

Dacă vom găsi un separator de tipul $\frac{1}{3} \leftrightarrow \frac{2}{3}$ pentru D cu cel mult $2\sqrt{n}$ vârfuri, atunci , atunci îl vom adăuga la $L(t_0) \cup L(t_2)$ pentru a obține un separator de cardinal cel mult $4\sqrt{n}$, reunim mulțimea cu cel mai mare număr de elemente dintre C și E cu partea mică rămasă din D pentru a obține A , iar partea mare rămasă în D o reunim cu cealaltă mulțime (mică) dintre C și E pentru a obține B .

Ne ocupăm de construcția separatorului pentru D . Vom șterge toate vârfurile grafului care nu-s în D cu excepția lui s pe care-l unim cu toate vârfurile de pe nivelul $t_0 + 1$ (primul nivel rămas în D). Graful obținut îl notăm cu D și este evident planar și conex. În plus are un arbore parțial T de diametru cel mult $2\sqrt{n}$ (orice vârf este accesibil din s pe un drum de lungime cel mult \sqrt{n} așa cum am arătat în lemă).

Acest arbore se construiește simplu pornind de la ultimul nivel și adăugând pentru fiecare vârf o muchie incidentă cu el și cu cealaltă extremitate pe nivelul precedent (nivelul 0 conține doar s).

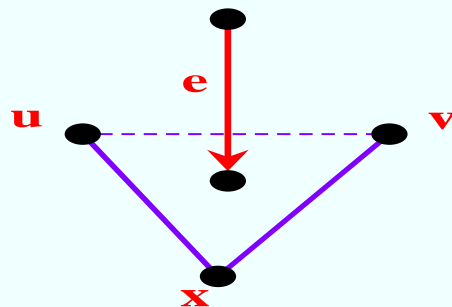
Putem presupune că D este triangulat (altfel se triangularizează în timp liniar). Construim dualul său D^* , tot în timp liniar. Muchiile lui D care nu-s pe arborele T vor fi numite **țepi**. Din lema 1, rezultă că țepii vor secționa un arbore parțial T^* al lui D^* . Considerăm o față a lui D drept rădăcină a lui T^* și considerăm muchiile lui T^* orientate dinspre rădăcină.

Fie $e = uv$ un țep. Există un unic drum de la u la v în T care împreună cu e determină un circuit $c(e)$.

Parcurgem țepii conform unei traversări dfs a lui T^* calculând următoarele informații pentru fiecare țep, recursiv de la frunze în sus:

- $I(e)$ - numărul vârfurilor din interiorul lui $c(e)$.
- $|c(e)|$ - numărul vârfurilor de pe circuitul $c(e)$;
- o reprezentare ca listă a lui $c(e)$.

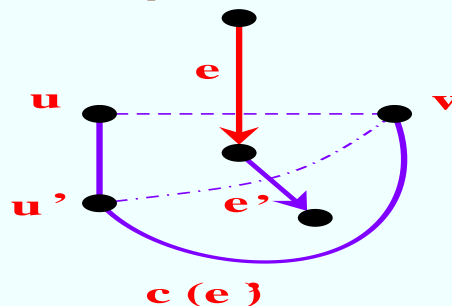
Sunt posibile următoarele patru cazuri (ce corespund înotarcerilor din parcurgerea dfs).



Cazul 1.

Suntem într-o frunză e a lui T^* (ce se determină numărând vecinii). Atunci:

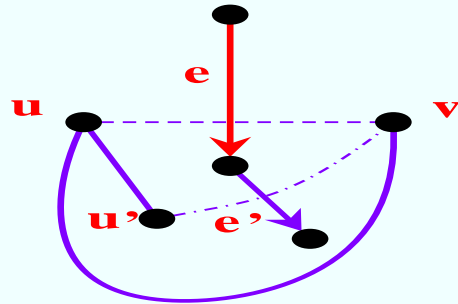
- $I(e) = 0$;
- $|c(e)| = 3$ (D e triangulat);
- $c(e) = [u, x, v]$.



Cazul 2.

Avem calculată informația pentru țepul $e' = u'v$, e este un țep în același triunghi ca și e' și u' este pe circuitul $c(e)$; ac. rezultă testând dacă u nu-i pe lista $c(e')$. Atunci:

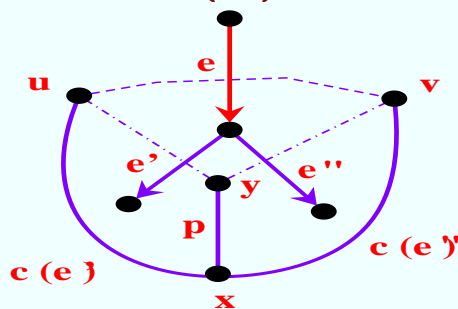
- $I(e) = I(e')$;
- $|c(e)| = c(e') + 1$;
- $c(e) = [u] \cdot c(e')$.



Cazul 3.

Avem calculată informația pentru țepul $e' = u'v$, e este un țep în același triunghi ca și e' și u' nu este pe circuitul $c(e)$; ac. rezultă testând dacă u e pe lista $c(e')$. Atunci:

- $I(e) = I(e') + 1$;
- $|c(e)| = c(e') - 1$;
- $c(e') = [u] \cdot c(e)$ (deci $c(e)$ se obține ștergându-l pe u din lista $c(e')$).



Cazul 4.

Avem calculată informația pentru țepii $e' = uy$ și $e'' = yv$ iar e este un țep în același triunghi ca și e', e'' .

Fie p drumul comun al lui $c(e')$ și $c(e'')$ și fie x celălalt capăt al lui p în afară de y . Atunci:

- $I(e) = I(e') + I(e'') + |p| - 1$ (toate vârfurile lui p cu excepția lui x sunt în interiorul lui $c(e)$);
- $|c(e)| = c(e') + c(e'') - 2|p| + 1$;
- $c(e) = c' \cdot [x] \cdot c''$ unde c' este $c(e')$ cu p șters și c'' este $c(e'')$ cu p șters.

Calculul lui $|p|$ și reprezentarea lui $c(e)$ se obțin scanând $c(e')$ și $c(e'')$ plecând din y până întâlnim ultimul vârf comun care este x . Aceasta nu afectează complexitatea de timp liniară.

Rămâne de arătat că există un țep e astfel încât

$$I(e) \leq \frac{2n_1}{3} \quad \wedge \quad n_1 - (I(e) + c(e)) \leq \frac{2n_1}{3}.$$

Atunci se poate lua $c(e)$ ca separator, vârfurile din interior drept o clasă, iar cele din interior drept cealaltă clasă.

Fie e primul țep întâlnit la întoarcerea din frunzele lui T^* către rădăcină, care satisface proprietatea că $I(e) + |c(e)| \geq \frac{n_1}{3}$.

Atunci mulțimea vârfurilor din exteriorul lui $c(e)$ este de cardinal $n_1 - (I(e) + |c(e)|) \leq \frac{2n_1}{3}$, deci ceea ce rămâne de arătat este că $I(e) \leq \frac{2n_1}{3}$.

Aceasta rezultă analizând cazurile 1-4 de mai sus în care s-a ajuns la e :

1. $I(e) = 0 \leq \frac{2n_1}{3}$.
2. $I(e) + |c(e)| = I(e') + |c(e')| + 1 < \frac{n_1}{3} + 1 \leq \frac{2n_1}{3}$ (pentru $n_1 \geq 3$).
3. $I(e) + c(e) = I(e') + c(e')$, deci e nu-i prima muchie cu proprietatea că $I(e) + |c(e)| \geq \frac{n_1}{3}$.
4. $I(e) + c(e) = I(e') + I(e'') + |p| - 1 + c(e') + c(e'') - 2|p| + 1 < \frac{2n_1}{3} - |p| < \frac{2n_1}{3}$.

Cu aceasta teorema separatorului este complet demonstrată.

Ilustrăm cum poate fi folosită teorema separatorului împreună cu o metodă de *divide & impera* pentru a obține algoritmi exacti cu comportare subexponențială pentru probleme NP-hard pe grafuri planare.

Considerăm problema testării dacă un graf planar dat admite o 3-colorare a vârfurilor (problemă cunoscută ca fiind NP-completă).

Pentru grafuri cu puține vârfuri (un număr constant c) se poate testa în timpul $O(3^c) = O(1)$ dacă graful are o 3-colorare.

Pentru grafuri planare cu numărul n de vârfuri mai mare decât c , construim în timp liniar $O(n)$, așa cum ne asigură teorema separatorului, partiția A, B, C a mulțimii vârfurilor sale cu $|A|, |B| \leq \frac{2n}{3}$ și $|C| \leq 4\sqrt{n}$.

Pentru fiecare din cele $3^{|C|} = 2^{O(\sqrt{n})}$ funcții posibile definite pe C și cu valori în $\{1, 2, 3\}$ se testează dacă este 3-colorare a subgrafului indus de C și dacă poate fi extinsă la o 3-colorare a subgrafului indus de $A \cup C$ în G și la o 3-colorare a subgrafului indus de $B \cup C$ în G (recursiv).

Timpul de lucru al acestui algoritm, $T(n)$, va satisface recurența

$$T(n) = \begin{cases} O(1) & \text{dacă } n \leq c; \\ O(n) + 2^{O(\sqrt{n})}(O(\sqrt{n}) + 2T(\frac{2n}{3})) & \text{dacă } n > c. \end{cases}$$

Se obține $T(n) = 2^{O(\sqrt{n})}$, destul de bun pentru probleme de dimensiuni rezonabile.

Este posibil însă ca notația $O(.)$ să ascundă constante mari !

Există și alte abordări pentru obținerea de algoritmi performanți pentru problemele dificile pe grafuri planare. Descriem în continuare una din ele.

Definiție O **t-descompunere** a unui graf $G = (V(G), E(G))$ este o pereche $(\{X_i | i \in V(T)\}, T)$, unde $\{X_i | i \in V(T)\}$ este o familie de submulțimi ale lui $V(G)$ și T este un arbore, astfel încât:

1. $\cup_{i \in V(T)} X_i = V(G)$;
2. $\forall vw \in E(G) \exists i \in V(T)$ astfel încât $v, w \in X_i$;
3. $\forall v \in V(G)$ mulțimea vârfurilor $\{i | v \in X_i\}$ induce un subarbore în T .

Lățimea unei t-descompuneri $\{X_i | i \in V(T)\}$ se definește ca fiind $\max_{i \in V(T)} (|X_i| - 1)$.

t-lățimea unui graf G este lățimea minimă a unei t-descompuneri a lui G și se notează cu $\text{tw}(G)$.

Definiție O **b-descompunere** a unui graf $G = (V(G), E(G))$ este o pereche (T, τ) , unde T este un arbore cu vârfurile de gradul 1 sau 3 și τ este o bijecție de la mulțimea vârfurilor pendante ale lui T la $E(G)$.

Ordinul unei muchii e a lui T este numărul vârfurilor $v \in V(G)$ astfel încât există vârfurile pendante t_1, t_2 ale lui T în componente conexe diferite ale lui $T - e$ cu $\tau(t_1)$ și $\tau(t_2)$ incidente cu v .

Lațimea b-descompunerii (T, τ) este ordinul maxim al unei muchii a lui T .

b-lățimea unui graf G este lățimea minimă a unei b-descompuneri a lui G și se notează cu $\text{bw}(G)$. (dacă $|E(G)| \leq 1$ se definește b-lățimea lui G ca fiind 0; grafurile nule nu au b-descompuneri; dacă $|E(G)| = 1$ atunci G are o b-descompunere care constă dintr-un arbore cu un nod, a cărui lățime se consideră 0).

Teoremă. 2. (Robertson & Seymour, '90)
Pentru orice graf conex G cu $E(G) \geq 3$ avem

$$\text{bw}(G) \leq \text{tw}(G) + 1 \leq \frac{3}{2}\text{bw}(G).$$

Teoremă. 3. (Fomin & Thilikos, 2003)
Pentru orice graf planar G avem

$$\text{bw}(G) \leq \sqrt{4.5|V(G)|} \leq 2.122\sqrt{|V(G)|}.$$

Teoremă. 4. (Fomin & Thilikos, 2003)
Fie Π o problemă de optimizare care este rezolvabilă pe grafuri cu b -lăţimea cel mult l şi ordinul n în timpul $f(l)g(n)$. Atunci pe grafurile planare de ordin n problema Π este rezolvabilă în timpul $O(f(2.122\sqrt{n})g(n) + n^4)$.

Demonstraţia rezultă utilizând teorema anterioară şi un algoritm de complexitate $O(n^4)$ dat de Robertson & Seymour pentru determinarea unei b -descompuneri optimale a unui graf.

De exemplu, se știe că problema determinării unei mulțimi stabile de cardinal maxim într-un graf cu t-lățimea l se poate rezolva în timpul $O(2^l n)$. Deci, pentru grafuri cu b-lățimea cel mult l se poate rezolva în $O(2^{\frac{3}{2}l} n)$.

Obținem că problema determinării unei mulțimi stabile de cardinal maxim într-un graf planar se poate rezolva în timpul $O(2^{3.182\sqrt{n}} n + n^4)$.

Timpii obținuți sunt mai buni decât metodele bazate pe teoremele de separare !

E N D

Seminarii

Setul de probleme 1

Problema 1. Un graf G se numește *rar* dacă numărul său de muchii m este mai mic decît $\frac{n^2}{\log n}$, unde n reprezintă numărul de virfuri. O justificare este aceea că matricea de adiacență A a grafului, care ocupă n^2 locații de memorie, poate fi întotdeauna reprezentată folosind $O(\frac{n^2}{\log n})$, locații de memorie astfel încît răspunsul la o întrebare " $A(i, j) = 1$? " să se facă în $O(1)$. Descrieți o astfel de schemă de reprezentare. (4 puncte)

Problema 2. Diametrul unui graf este lungimea maximă a unui drum de lungime minimă între două virfuri ale grafului. Două virfuri care sunt extremitățile unui drum minim de lungime maximă în graf se numesc diametral opuse. Demonstrați că următorul algoritm determină o pereche de virfuri diametral opuse într-un arbore T :

- dintr-un virf oarecare se execută o parcurgere BFS a lui T ; fie u ultimul virf vizitat; din virful u se execută o

parcurgere BFS a lui T ; fie v ultimul virf vizitat;

- return u, v .

Este valabil algoritmul pentru un graf conex oarecare ?
(4 puncte)

Problema 3. Fie T un arbore un arbore binar cu rădăcină. Un algoritm simplu de desenare a lui T poate fi descris recursiv după cum urmează.

- Folosim ca suport o grilă (liniatura unui caiet de mate); virfurile se plasează în punctele de intersecție ale grilei.
- Desenăm subarborele stîng; Desenăm subarborele drept.
- Plasăm cele două desene unul lângă altul la distanță pe orizontală doi și cu rădăcinile la aceeași înălțime.
- Plasăm rădăcina cu un nivel mai sus la jumătatea distanței pe orizontală dintre cei doi copii.
- Dacă avem doar un copil plasăm rădăcina cu un nivel mai sus la distanța 1 față de copil (la stînga sau la dreapta după cum este acesta).

Descrieți cum se poate asocia pentru fiecare nod v al arborelui T (folosind algoritmul de mai sus) coordonatele $(x(v), y(v))$ reprezentînd punctul de pe grilă unde va fi desenat. **(3 puncte)**

Problema 4. Intr-o sesiune de examene s-au inscris n studenți care trebuie să susțină examene dintr-o mulțime de m discipline. Intrucit examenele se susțin în scris, se dorește ca toți studenții care dau examen la o disciplină să facă acest lucru simultan. De asemenea, regulamentul de desfășurare a examenelor interzice ca un student să dea două examene în aceeași zi. Pentru fiecare student se dispune de lista disciplinelor la care dorește să fie examinat.

Să se descrie construcția unui graf G care să ofere răspunsul la următoarele două întrebări prin determinarea unor parametri asociați (care se vor preciza):

- care e numărul maxim de examene ce se pot organiza în aceeași zi ?
- care e numărul minim de zile necesare organizării tuturor examenelor? (**3 puncte**)

Setul de probleme 2

Problema 1. Fie $G = (S, T; E)$ un graf bipartit și $X \in \{S, T\}$. Graful G se numește X -lanț dacă vârfurile mulțimii X pot fi ordonate $X = \{x_1, x_2, \dots, x_p\}$ (unde $p = |X|$) astfel încât $N_G(x_1) \supseteq N_G(x_2) \supseteq \dots \supseteq N_G(x_p)$.

a) Demonstrați că G este X -lanț dacă și numai dacă este \overline{X} -lanț, unde $\overline{X} = \{S, T\} - \{X\}$. (2 puncte)

b) Dacă G (bipartit) este reprezentat cu ajutorul listelor de adiacență, are ordinul n și dimensiunea m , descrieți un algoritm cu timpul $O(n + m)$ care să testeze dacă G este S -lanț. (2 puncte)

Problema 2. Un graf G se numește *autocomplementar* dacă este izomorf cu complementul său : $G \simeq \overline{G}$.

a) Demonstrați că un graf autocomplementar este conex și că ordinul său este multiplu de 4 sau multiplu de 4 plus 1. (2 puncte)

b) Demonstrați că pentru orice graf G există un graf autocomplementar H astfel încât G este subgraf indus în H . (2 puncte)

c) Determinați toate grafurile autocomplementare cu cel mult 7 virfuri. (2 puncte)

Problema 3. O echipă de doi programatori L (azy) și T (hinky) primește ca sarcină să determine un drum între 2 noduri date, care să satisfacă anumite cerințe, într-un graf G dat, despre care se știe că este rar : $|E(G)| = O(|G|)$. Programatorul L propune ca soluție generarea (cu backtracking) a tuturor drumurilor dintre cele două noduri și selectarea celui convenabil, motivând că într-un astfel de graf nu pot exista prea multe drumuri între două noduri fixate (sunt puține muchii și deci puține posibilități de ramificare; de ex., într-un arbore există exact un drum între orice două noduri fixate). Programatorul T nu-i de acord și dă următorul contraexemplu: se consideră graful $H = K_2 \times P_{n-1}$ (n un întreg mare); o pereche de virfuri de grad 2 adiacente din H se unește cu un virf nou x , iar cealaltă pereche de virfuri de grad 2 adiacente din H se unește cu un virf nou y ; graful obținut, G , are proprietățile din problema de rezolvat și totuși numărul drumurilor de la x la y în G este prea mare. Ajutați-l pe L să înțeleagă contraexemplul, desenind graful G , arătând că este rar și estimind numărul drumurilor de la x la y . **(2 puncte)**

Problema 4. Presupunem că un *turneu* (digraf cu proprietatea că orice 2 virfuri sunt unite exact printr-un arc) are un circuit C de lungime $n \geq 4$.

Arătați că pentru orice virf x al lui C se pot determina în timpul $O(n)$, încă două virfuri ale lui C y și z astfel încât (x, y, z) este un circuit de lungime 3. **(2 puncte)**

Setul de probleme 3

Problema 1. Fie $G = (V, E)$ un graf cu n virfuri, m muchii și cu matricea de adiacență A . Dintre cele 2^m orientări posibile ale muchiilor sale considerăm una oarecare și cu ajutorul ei construim matricea de incidență virf-arc $Q \in \{0, 1, -1\}^{n \times m}$ definită prin :

$(Q)_{ve} = -1$, dacă v este extremit. inițială a arcului e ,

$(Q)_{ve} = 1$, dacă v este extremitatea finală a arcului e

$(Q)_{ve} = 0$ în toate celelalte cazuri.

Demonstrați că matricea $A + QQ^T$ este o matrice diagonală și precizați semnificația combinatorie a elementelor ei. **(3 puncte)**

Problema 2. Fie G un graf oarecare și notăm cu $b(G)$ graful obținut din G prin inserarea cite unui nou nod pe fiecare muchie. Demonstrați că $b(G)$ este un graf bipartit. **(2 puncte)**

Demonstrați că G și H sunt izomorfe dacă și numai dacă $b(G)$ este izomorf cu $b(H)$. Deduceți că testarea izomorfismului a 2 grafuri oarecare se reduce polinomial la testarea izomorfismului a 2 grafuri bipartite **(2 puncte)**

Problema 3. Graful *paianjen* cu n virfuri este graful care se obține unind unul din virfurile de grad 1 ale grafului P_3 cu toate virfurile unui graf oarecare cu $n - 3$ virfuri, disjunct de P_3 (n este un întreg pozitiv mare).

Dacă G este un graf cu n virfuri reprezentat prin matricea de adiacență, arătați că se poate testa dacă este graf paianjen folosind doar $O(n)$ probe ale matricii de adiacență.

(o probă este un acces la un element oarecare al matricii, fără a-l memora explicit pentru utilizări ulterioare). **(4 puncte)**

Problema 4. Asociem unui arbore binar T de ordin n cu rădăcina r un drum P_{3n} orientat procedind astfel: fiecărui nod v al lui T i se asociază trei noduri cu același nume v pe care le desemnăm prin v_1, v_2, v_3 ; dacă v nu are în T descendent stîng, atunci se introduce arcul v_1v_2 în P_{3n} ; dacă v nu are în T descendent drept, atunci se introduce arcul v_2v_3 în P_{3n} ; dacă descendentul stîng al lui v în T este w , atunci se introduc în P_{3n} arcele v_1w_1 și w_3v_2 ; dacă descendentul drept al lui v în T este w , atunci se introduc în P_{3n} arcele v_2w_1 și w_3v_3 .

Dacă se parcurge drumul P_{3n} de la extremitatea inițială r_1 la extremitatea finală r_3 și se listează numele virfurilor în ordinea parcurgerii lor se obține un șir în care numele fiecărui virf al lui T apare exact de trei ori.

Demonstrați că :

dacă din acest șir se reține doar prima apariție a fiecărui nume se obține parcurgerea *pre-order* a arborelui T ;

dacă din acest șir se reține doar a doua apariție a fiecărui nume se obține parcurgerea *in-order* a arborelui T ;

dacă din acest șir se reține doar a treia apariție a fiecărui nume se obține parcurgerea *post-order* a arborelui T .

(3 puncte)

Setul de probleme 3'

Problema 1. Fie $G = (V, E)$ un graf de ordin n și dimensiune m . O ordonare $V = \{v_{i_1}, \dots, v_{i_n}\}$ a vârfurilor lui G se numește *d-mărginită* dacă în digraful G^{\mapsto} , obținut din G prin înlocuirea fiecărei muchii $\{v_{i_j}, v_{i_k}\}$ cu arcul $(v_{i_{\min\{j,k\}}}, v_{i_{\max\{j,k\}}})$, avem $\forall v \in V \quad d_{G^{\mapsto}}^+(v) \leq d$.

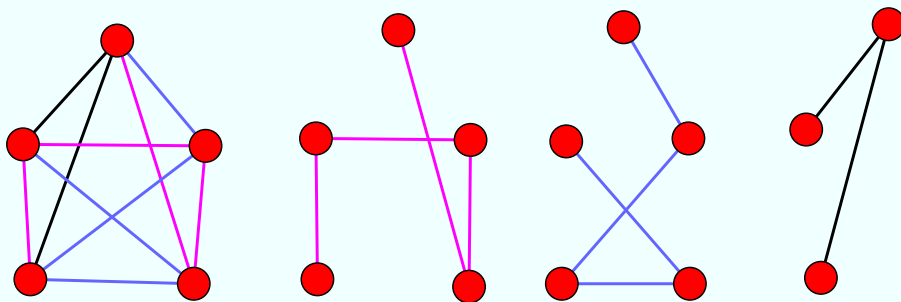
a) Descrieți un algoritm care primind la intrare G reprezentat cu ajutorul listelor de adiacență și $d \in \mathbb{N}^*$, testează în timpul $O(n + m)$ dacă G are o ordonare *d-mărginită* (se vor argumenta corectitudinea și complexitatea). (2 puncte)

b) Utilizați algoritmul de la punctul a) pentru a determina în timpul $O(m \log n)$ parametrul

$o(G) = \min\{d \in \mathbb{N} \mid G \text{ are o ordonare } d\text{-mărginită}\}$. (2 puncte)

c) Arătați că orice graf G admite o colorare a vârfurilor cu $o(G) + 1$ culori. (2 puncte)

Problema 2. Demonstrați algoritmic că mulțimea muchiilor oricărui graf complet K_n ($n \geq 2$) poate fi partiționată în $\lceil \frac{n}{2} \rceil$ submulțimi, fiecare dintre acestea reprezentând mulțimea muchiilor unui arbore (subgraf al lui K_n). Exemplu. K_5 , $\lceil \frac{5}{2} \rceil = 3$:



(4 puncte)

Problema 3. Pentru un graf conex G se execută următorul algoritm:

- Se inițializează o coadă Q cu graful G .
- Cât timp coada Q nu-i vidă:
 - se extrage în H graful din capul cozii,
 - se determină o mulțime de articulație $A \subseteq V(H)$, minimală în raport cu incluziunea (nici o submulțime proprie nu-i mulțime de articulație în H), și dacă V_1, \dots, V_k ($k \geq 2$) sunt mulțimile de vârfuri ale componentelor conexe ale grafului $H - A$, atunci
 - se adaugă la Q grafurile $[A \cup V_1]_H, \dots, [A \cup V_k]_H$.

Se observă că dacă graful curent este complet atunci nu se adaugă nimic în coada Q .

a) Arătați că fiecare graf introdus în coadă este conex. **(2 puncte)**

b) Demonstrați că numărul total al grafurilor introduse în coada Q nu depășește $|G|^2$. **(2 puncte)**

Setul de probleme 3''

Problema 1. Fie \mathcal{C} clasa grafurilor G cu proprietatea că orice arbore dfs al lui G este un drum (pentru orice ordonare a vârfurilor lui G și orice ordonare a listelor de adiacență asociate acestor vârfuri, orice aplicare a unui dfs generează un drum hamiltonian în G).

Demonstrați că

$$\mathcal{C} = \{K_1, K_2\} \cup \bigcup_{n \geq 3} \{K_n, C_n, K_{n,n}\}.$$

(1+3 puncte)

Problema 2. Fie $D = (V, E)$ un digraf (fără bucle) de ordin n cu mulțimea vârfurilor $V = \{1, 2, \dots, n\}$. Considerăm următorul algoritm:

1. $SK \leftarrow \emptyset$;
 for $i = 1$ **to** n **do** $\backslash \backslash$ stânga \rightarrow dreapta
 if $(\nexists j \in SK \text{ astfel încât } ji \in E)$ **then** $SK \leftarrow SK \cup \{i\}$;
2. **for** $i = n$ **to** 1 **do** $\backslash \backslash$ dreapta \rightarrow stânga
 if $i \in SK \wedge (\exists j \in SK \text{ astfel încât } ji \in E)$ **then** $SK \leftarrow SK \setminus \{i\}$;
3. **output** SK .

Demonstrați că SK este un *seminucleu* în D : SK este nevidă, stabilă în $G[D]$ (graful suport al digrafului D) și orice vârf din $v \in V \setminus SK$ e accesibil în D , dintr-un vârf al lui SK , pe un drum de lungime cel mult 2.

Indicați structurile de date și modul de folosire a acestora pentru o implementare a algoritmului de mai sus în timpul $\mathcal{O}(n + m)$ (m fiind $|E|$).

(2+2 puncte)

Problema 3. Arătați că dacă $G = (S, T; E)$ este un graf bipartit cu următoarele proprietăți:

- $|S| = n; |T| = m$ ($n, m \in \mathbb{N}^*$);
- $\forall t \in T \ |N_G(t)| > k > 0$; (pentru un k oarecare — mai mic decât n);
- $\forall t_1, t_2 \in T$ dacă $t_1 \neq t_2$ atunci $N_G(t_1) \neq N_G(t_2)$;
- $\forall t_1, t_2 \in T$ dacă $t_1 \neq t_2$ atunci $|N_G(t_1) \cap N_G(t_2)| = k$,
atunci are loc inegalitatea $m \leq n$.

(2 puncte)

Problema 4.

Pentru $n \in \mathbb{N}^*$ definim graful $G_n = (V, E)$ astfel:

- $V = \{(i, j) | 1 \leq i \leq n, 1 \leq j \leq n\}$,
- $(i, j)(k, l) \in E$ (pentru două vârfuri (i, j) și (k, l) distincte din V) dacă și numai dacă $i = l$ sau $j = k$.

Demonstrați că G_n este *universal pentru familia arborilor de ordin n* :

oricare ar fi T un arbore de ordin n există $A \subset V$ astfel încât $T \cong [A]_{G_n}$.

(2+2 puncte)

Setul de probleme 4

Problema 1. Prezentați (pe cel mult o pagină) o problemă interesantă din domeniul IT care să necesite rezolvarea eficientă a unei probleme de drum minim într-un digraf asociat problemei inițiale. (3 puncte)

Problema 2. Fie $G = (V, E)$ un graf, $s \in V$ un virf oarecare al lui G iar t un alt virf, accesibil în G printr-un drum din s . O mulțime A de muchii se numește *st-inevitabilă* dacă există $S \subset V$ astfel încât $s \in S$, $t \notin S$ și $A = \{e \in E \mid e = uv, u \in S, v \notin S\}$. Arătați că numărul maxim de mulțimi *st-inevitabile* disjuncte două câte două este egal cu distanța în G de la s la t și că se poate determina o familie de astfel de mulțimi cu ajutorul unui bfs a lui G din s . (3 puncte)

Problema 3. Fie $G = (V, E)$ un graf conex și v un virf al său cu proprietatea că $N_G(v) \neq V - \{v\}$. Dacă pentru $A \subset V$ notăm cu $N_G(A) = \cup_{a \in A} N_G(a) - A$, se observă că există mulțimi de virfuri A care satisfac proprietățile : $v \in A$, $[A]_G$ este conex, $N = N_G(A) \neq \emptyset$ și $R = V - (A \cup N) \neq \emptyset$ (de exemplu, $A = \{v\}$).

- a) Demonstrați că dacă se consideră o mulțime A maximală (în raport cu incluziunea) satisfăcând proprietățile enunțate, atunci orice virf din R este adiacent cu orice virf din N . **(2 puncte)**
- b) Dacă, în plus, graful G este $\{C_k\}_{k \geq 4}$ -free, atunci mulțimea N de la punctul a) are proprietatea că este clică în graful G . **(2 puncte)**
- c) Deduceți că singurele grafuri $\{C_k\}_{k \geq 4}$ -free, regulate și conexe sunt grafurile complete. **(2 puncte)**

Problema 4. Arătați că se poate utiliza o parcurgere dfs pentru a determina un circuit par într-un graf 3-regulat oarecare. **(2 puncte)**

Setul de probleme 4'

Problema 1. Fie $T = (V, E)$ un arbore cu măcar două vârfuri reprezentat cu ajutorul listelor de adiacență. Se declară un vârf oarecare $r \in V$ rădăcină și se notează cu $d(v)$ (pentru orice vârf $v \in V$) lista descendenților imediați ai lui v în parcurgerea *bfs* din r (vârfurilor pendante le corespund liste vide). Considerăm următorul algoritm:

1. Se construiesc tablourile de întregi $a[v]$ și $b[v]$ ($v \in V$) astfel: dacă $d(v)$ este vidă, atunci $a[v] \leftarrow 1$ și $b[v] \leftarrow 0$; dacă $d(v)$ este nevidă și toate vârfurile u din $d(v)$ au $a[u]$ și $b[u]$ calculate, atunci $a[v] \leftarrow 1 + \sum_{u \in d(v)} b[u]$ și $b[v] \leftarrow \sum_{u \in d(v)} \max(a[u], b[u])$.

2. Se returnează $x \leftarrow \max(a[r], b[r])$.

Descrieți în pseudocod algoritmul de mai sus, argumentați complexitatea timp de $O(|V|)$ și demonstrați că valoarea returnată x este numărul de stabilitate $\alpha(T)$ al arborelui T .
(1+1+2 = 4 puncte)

Problema 2. Fie $G = (V, E)$ un graf d -regulat de ordin n care satisface următoarea proprietate:

există $\alpha > 0$ astfel încât pentru orice mulțime de vârfuri $S \subset V$ cu proprietatea că $|S| \leq \frac{n}{2}$, numărul muchiilor cu o extremitate în S și cealaltă în $V - S$ este cel puțin $\alpha|S|$.

- a) Grafurile complete au proprietatea de mai sus ?
(argumentați răspunsul)
- b) Dacă α și d sunt constante (nu depind de n), demonstrați că diametrul lui G satisface

$$d(G) = \mathcal{O}(\log n).$$

(1+3 = 4 puncte)

Problema 3. Muchiile grafului conex $G = (V, E)$ se colorează arbitrar roșu și albastru.

a) Demonstrați că există în G un parcurs Euleian închis fără muchii consecutive de aceeași culoare dacă și numai dacă pentru fiecare vârf $v \in V$ al lui G numărul muchiilor roșii incidente cu v este egal cu numărul muchiilor albastre incidente cu v .

b) Dacă graful G este complet și x, y, z sunt trei vârfuri distincte ale sale demonstrați că *dacă există în G un drum fără muchii consecutive de aceeași culoare de la x la y trecând prin z , atunci există un drum cu aceeași proprietate care are ca primă muchie pe xz sau ca ultimă muchie pe zy .*

(1+1+2 = 4 puncte)

Problema 4. Fie G un graf și $\delta(G)$ gradul minim al unui vârf al său. Descrieți un algoritm care, pentru un arbore dat T cu $k \leq \delta(G)$ muchii, să construiască (în timp polinomial) un subgraf H al lui G astfel încât $H \cong T$.

(2 puncte)

Setul de probleme 5

Problema 1. Să se arate că un graf G este bipartit dacă și numai dacă orice subgraf indus H al lui G satisface proprietatea $2\alpha(H) \geq |H|$ (**3 puncte**)

Problema 2. Demonstrați că într-un graf bipartit G cu n virfuri și m muchii avem inegalitatea $4m \leq n^2$. (**2 puncte**)

Descrieți un algoritm care să testeze dacă un graf cu n virfuri și m muchii este complementarul unui graf bipartit în timpul $O(n + m)$ (**3 puncte**)

Problema 3. Arătați că orice graf G cu m muchii are un graf partial H bipartit și cu cel puțin $\frac{m}{2}$ muchii. (**3 puncte**)

Problema 4. Demonstrați că în orice graf conex $G = (V, E)$ există o mulțime stabilă S astfel încât graful bipartit $H = (S, V - S; E')$ este conex, unde $E' = E - \mathcal{P}_2(V - S)$. Deduceți că $\alpha(G) \geq \frac{|G| - 1}{\Delta(G)}$ pentru orice graf conex G . (**3 puncte**)

Setul de probleme 6

Problema 1. Pentru $d \in \mathbb{N}^*$ se consideră graful $G_d = \underbrace{K_2 \times K_2 \times \dots \times K_2}_{d \text{ factori}}$.

Să se determine ordinul, dimensiunea și diametrul lui G_d .
(2 puncte)

Să se arate că G_d este bipartit și să se determine $\alpha(G_d)$.
(2 puncte)

Problema 2. Un graf cu cel puțin trei virfuri se numește *confidențial conex* dacă pentru orice trei virfuri distincte a, b, c ale grafului există un drum de la a la b astfel încât niciunul dintre virfurile interne ale acestui drum (dacă există astfel de virfuri) nu este c sau un vecin al lui c . Un exemplu banal de graf confidențial conex este graful K_n cu $n \geq 3$.

Demonstrați că un graf conex $G = (V, E)$, cu cel puțin trei virfuri și care nu-i complet, este confidențial conex dacă și numai dacă au loc următoarele două condiții :

1. Pentru orice virf v mulțimea $\overline{N}(v) = \{w \in V \mid w \neq v, vw \notin E\}$ este nevidă și induce un graf conex.
2. Orice muchie a grafului este conținută într-un C_4 indus în graf sau este muchia din mijlocul unui P_4 indus în graf.

(4 puncte)

Problema 3. În Problema 2-SAT se dau : o mulțime de variabile boolene $U = \{x_1, x_2, \dots, x_n\}$ și o mulțime de clauze $C = \{C_1, C_2, \dots, C_m\}$, unde fiecare clauză C_i este disjuncția a doi literali $C_i = v_i \vee w_i$, literalii reprezentând variabile sau negațiile acestora. Problemei i se asociază un digraf G cu $V(G) = \{x_1, x_2, \dots, x_n, \overline{x_1}, \overline{x_2}, \dots, \overline{x_n}\}$ (adică toți literalii posibili) și în care pentru fiecare clauză $C_i = v_i \vee w_i$ se adaugă arcele $\overline{v_i}w_i$ și $\overline{w_i}v_i$ (folosind, evident, convenția referitoare la dubla negare). Demonstrați că există o atribuire a valorilor de adevăr și fals pentru variabilele booleene, astfel încât fiecare clauză să fie adevărată, dacă și numai dacă digraful G are proprietatea că pentru orice $i \in \{1, \dots, n\}$ $\overline{x_i}$ și x_i aparțin la componente tari conexe diferite. **(4 puncte)**

Argumentați complexitatea timp de $O(n + m)$ pentru testarea proprietății de mai sus. **(2 puncte)**

Setul de probleme 7

Problema 1. Gossip Problem. Intr-un grup de n "doamne", fiecare cunoaște o parte dintr-o birfă pe care celelalte $n-1$ o cunosc. Ele comunică prin telefon și orice apel telefonic între orice două doamne are ca efect faptul că fiecare din ele va afla tot ce cunoaște cealaltă.

(a) Descrieți o schemă de a da telefoanele astfel încât într-un număr minim $f(n)$ de apeluri telefonice, fiecare "doamnă" va afla tot ce știe celelalte.

Indicație: Arătați că $f(2) = 1, f(3) = 3, f(4) = 4$ și pentru $n > 4$ $f(n) = 2n - 4$ (ușor, indicind scheme de telefonare cu aceste numere de apeluri). Incercați să argumentați că $2n - 4$ este chiar numărul minim.

(2 puncte pentru descrierea schemei, **1 punct** pentru demonstrarea optimalității)

(b) Modelați problema în limbajul teoriei grafurilor: schemei de telefonare îi va corespunde un șir de muchii iar cunoașterea comună se va exprima printr-o condiție referitoare la existența unor drumuri speciale cu elemente din șirul considerat **(1 punct)**

Problema 2. Fie D un digraf și două funcții definite pe mulțimea arcelor sale, $a : E(D) \longrightarrow R_+$ și $b : E(D) \longrightarrow R_+^*$. Descrieți un algoritm eficient pentru determinarea unui circuit C^* în D astfel încît

$$\frac{a(C^*)}{b(C^*)} = \min \left\{ \frac{a(C)}{b(C)}; C \text{ circuit în } D \right\}$$

(4 puncte)

Problema 3. Fie A_1, A_2, \dots, A_n submulțimi distincte ale unei mulțimi de n elemente S . Demonstrați că există un element x în mulțimea S astfel încît $A_1 - \{x\}, A_2 - \{x\}, \dots, A_n - \{x\}$ să fie și ele distincte. **(2 puncte)**

Problema 4. Fie G un graf și $c : E(G) \longrightarrow R_+$ o funcție de capacitate a muchiilor. Oricărui drum din graf cu măcar o muchie i se asociază *locul îngust* ca fiind muchia sa de capacitate minimă. Descrieți un algoritm eficient care să determine pentru două virfuri s și t distincte ale grafului drumul cu locul îngust cel mai mare (dintre toate drumurile de la s la t în graful G).

(4 puncte)

Setul de probleme 7'

Problema 1. Fie $G = (V, E)$ un digraf de ordin n , $a : E \rightarrow \mathbf{R}_+$ o funcție de cost nenegativă, și $s \neq t$ două vârfuri fixate. Pentru rezolvarea problemei **P1** (a determinării unui drum de cost a minim de la s la t în G) se propune următorul algoritm:

1. **for** each $i \in V$ **do** $p_i \leftarrow 0$;
 $i \leftarrow s; \text{înainte}(s) \leftarrow s$;
2. **while** $i \neq t$ **do**
 if $\exists j \in V$ astfel încât $p_i - p_j = a_{ij}$ **then**
 $\{ \text{înainte}(j) \leftarrow i; i \leftarrow j; \}$
 else
 $\{ p_i \leftarrow \min_{ij \in E} (a_{ij} + p_j); i \leftarrow \text{înainte}(i) \}$;
3. Costul unui drum de cost minim de la s la t este $p_s - p_t$ și un drum de cost minim se obține din:
 $t, \text{înainte}(t), \text{înainte}(\text{înainte}(t)), \dots, s$.

a) Demonstrați că dacă pasul 2 se termină atunci afirmațiile din pasul 3 sunt corecte. (2 puncte)

b) Stabiliți complexitatea timp a algoritmului (2 puncte)

Problema 2. Fie $T = (V, E)$ un arbore și $w : V \rightarrow \mathbf{R}_+$ o funcție de pondere nenegativă. Pentru orice subarbore T' al lui T se definește ponderea sa, $w(T')$, ca fiind suma ponderilor vârfurilor sale.

Arătați că există un vârf $v_0 \in V$ astfel încât nici unul din subarborii lui $T - v_0$ nu are ponderea mai mare decât $\frac{1}{2}w(T)$. (1 punct)

Descrieți un algoritm cu timpul $O(|V|)$ pentru găsirea lui v_0 . (2 puncte)

Problema 3. Dacă G și H sunt două grafuri, notația $G \rightarrow H$ semnifică faptul că există $f : V(G) \rightarrow V(H)$ astfel încât $\forall uv \in E(G)$ avem că $f(u)f(v) \in E(H)$ (există un morfism de grafuri de la G la H).

Justificați corectitudinea unui algoritm care să răspundă în timpul $O(1)$ la întrebarea: "Are loc $C_n \rightarrow C_m$?" ($n, m \in \mathbb{N}, n, m \geq 3$; C_k este graful circuit de ordin k). **(3 puncte)**

Problema 4. Dacă H este un graf, atunci $q(H)$ notează numărul componentelor conexe de ordin impar ale lui H , iar $\nu(H)$ cardinalul maxim al unui cuplaj al lui H . Demonstrați că pentru orice graf G are loc relația:

$$\max_{S \subset V(G)} (q(G - S) - |S|) = |V(G)| - 2\nu(G) .$$

(Se presupune cunoscută teorema lui Tutte) **(4 puncte)**

Setul de probleme 7''

Problema 1. Determinați numărul cuplajelor perfecte ale grafului:



(3 puncte)

Problema 2.

a) Fie $D = (V, E)$ un digraf aciclic cu n vârfuri și m arce și $A, B \subset V$ două mulțimi disjuncte, stabile în $G(D)$ (graful suport al digrafului). Fie $d(A, B) := \min\{d(a, b) \mid a \in A, b \in B\}$ ($d(x, y)$ = distanța în D de la x la y = lungimea celui mai scurt drum dintre x și y , dacă acesta există). Descrieți un algoritm de complexitate $\mathcal{O}(n + m)$ pentru aflarea unei mulțimi maximale \mathcal{P} de drumuri disjuncte (cu mulțimile de vârfuri disjuncte) de la A la B , fiecare de lungime $d(A, B)$ (maximalitatea lui \mathcal{P} este în raport cu incluziunea, adică nu mai există un alt drum de la A la B care să aibă lungimea $d(A, B)$ și să fie disjunct de orice drum din \mathcal{P}).

b) Arătați cum poate fi folosit algoritmul de la a) pentru implementarea algoritmului lui Hopcroft & Karp de aflare a unui cuplaj de cardinal maxim într-un graf bipartit.

(3+2 puncte)

Problema 3. Fie $D = (V, E)$ un digraf cu mulțimea de vârfuri $V = \{1, \dots, n\}$ și mulțimea arcelor $E = \{e_1, \dots, e_m\}$.

Fie $A = (a_{ij}) \in \mathcal{M}_{n \times m}(\{-1, 0, 1\})$ matricea de incidență a lui D (dacă arcul e_j iese din i atunci $a_{ij} = 1$, dacă arcul e_j intră în i atunci $a_{ij} = -1$, altfel $a_{ij} = 0$). Arătați că pentru orice submatrice pătrată B a lui A are loc:

$$\det(B) \in \{-1, 0, 1\}.$$

(2 puncte)

Problema 4. Într-un graf fără vârfuri izolate se construiește un drum P astfel: se pleacă dintr-un vârf oarecare de start și apoi, din vârful curent în care ne aflăm, alegem un vecin diferit de vârfurile deja vizitate. Atunci când nu mai este posibilă nici o alegere, construcția lui P se încheie. Evident, lungimea drumului P este cel puțin 1 și ea depinde de structura grafului și de alegerile făcute. Proprietarul grafului solicită o plată pentru folosirea acestuia în procesul de construcție a drumului P . Această plată se poate face înaintea fiecărei alegeri și, dacă se plătește 1 RON se obține dreptul de a face această alegere, iar dacă se plătesc $T \gg 1$ RONi atunci se obține dreptul de a face gratuit toate alegerile următoare. După terminarea construcției se poate compara suma plătită, $Apriori(P)$, cu cea care s-ar fi făcut dacă s-ar fi cunoscut drumul P , notată $Posteriori(P)$. Găsiți o strategie de plată astfel încât pentru orice graf și orice drum construit P să avem $Apriori(P) \leq (2 - 1/T)Posteriori(P)$.

(4 puncte)

Setul de probleme 8

Problema 1. Fie G un graf conex și o funcție de cost $c : E(G) \rightarrow R$. Vom numi *tăietură* orice mulțime A de muchii ale lui G cu proprietatea că există o bipartiție $(S, V(G) - S)$ a mulțimii virfurilor lui G astfel încât A este mulțimea muchiilor lui G cu extremitățile în clase diferite ale bipartiției.

- a) Arătați că dacă funcția de cost are proprietatea că orice tăietură are o unică muchie de cost minim, atunci există un unic arbore parțial de cost minim. **(2 puncte)**
- b) Deduceți că, dacă funcția de cost c este injectivă, atunci G are un unic arbore parțial de cost minim. **(1 punct)**
- c) Sunt adevărate reciprocele afirmațiilor a) și b) ? **(1 punct)**

Problema 2. Considerăm o numerotare fixată a celor $m > 0$ muchii ale unui graf conex $G = (V, E)$ de ordin n . Pentru orice submulțime de muchii A considerăm $x^A \in GF^m$ vectorul m -dimensional cu elemente 0,1 definit prin $x_i^A = 1 \Leftrightarrow e_i \in A$ (vect. caracteristic). GF^m este spațiul vectorial peste corpul GF (cu elem. 0 și 1, și operațiile de adunare și înmulțire modulo 2).

a) Demonstrați că mulțimea vectorilor caracteristici ai tuturor tăieturilor grafului G , la care adăugăm și vectorul nul, formează un subspațiu vectorial X al lui GF^m . **(1 punct)**

b) Demonstrați că vectorii caracteristici ai mulțimilor muchiilor circuitelor grafului G generează un subspațiu vectorial U al lui GF^m ortogonal pe X . **(1 punct)**

c) Arătați că $\dim(X) \geq n - 1$ **(1 punct)**

d) Arătați că $\dim(U) \geq m - n + 1$ **(1 punct)**

e) Deduceți că $\dim(X) = n - 1$ și că $\dim(U) = m - n + 1$. **(1 punct)**

Problema 3. Arătați că orice arbore cu gradul maxim $t > 0$ are cel puțin t virfuri pendante. **(2 puncte)**

Problema 4. Fie $T = (V, E)$ un arbore cu rădăcina r (un virf oarecare) și cu $\text{parent}(v)$ părintele nodului $v \in V$, $v \neq r$. Un cuplaj M al lui T se numește *propriu* dacă orice virf expus v (relativ la M) în T are un frate w (două virfuri sunt frați dacă au același părinte) astfel încât $w \text{ parent}(v) \in M$. a) Demonstrați că orice cuplaj propriu este de cardinal maxim. **(1 punct)**

b) Arătați că pentru orice arbore cu n virfuri, dat prin listele de adiacență, se poate construi în timpul $O(n)$ un cuplaj propriu. **(2 puncte)**

Setul de probleme 8'

Problema 1.

O procedura naivă de creare a unei rețele sociale $G = (V, E)$ este următoarea:

Inițializare

$G = (\{v\}, \emptyset)$ // v este creatorul rețelei

Aderarea la G (a unui nou membru $v \notin V(G)$)

$V \leftarrow V(G) \cup \{v\}; E \leftarrow E(G)$

if v has a friend in $V(G)$ **then** $E \leftarrow E \cup \{vw | w \in V(G)\}$
 $G \leftarrow (V, E)$

a) Descrieți un algoritm eficient care, pentru un graf dat G , să decidă dacă este o rețea socială naivă (a fost creat cu algoritmul de mai sus).

b) Administratorul rețelei a observat că pentru orice doi utilizatori v și w ai rețelei $G = (V, E)$ ($v \neq w \in V$), se poate răspunde în timp constant la o întrebare $vw \in E$? astfel: se asociază grafului o valoare întreagă pozitivă $x(G)$ și câte o valoare întreagă pozitivă $y(v)$ pentru fiecare vârf v , astfel încât pentru orice două vârfuri distincte v și w are loc $vw \in E \Leftrightarrow y(v) + y(w) > x(G)$. Modificați algoritmul de mai sus pentru a realiza această nouă reprezentare a grafului (se întrețin doar mulțimea vârfurilor V , $x(G)$ și lista $(y(v))_{v \in V}$).

(2+2 = 4 puncte)

Problema 2. Considerăm următoarea euristică pentru determinarea unui cuplaj de cardinal maxim într-un graf bipartit $G = (S, T; E)$:

```
function  $MMB(G = (S, T; E))$   
     $M = \emptyset$   
    while  $E \neq \emptyset$  do {  
        if  $\exists u$  varf de grad 1 then  $e \leftarrow$  muchia incidentă cu  $u$   
        else  $e \leftarrow$  o muchie incidentă cu un vârf de grad maxim  
         $M \leftarrow M \cup \{e\}$   
        se șterg extremitățile muchiei  $e$  din  $G$   
    }  
    return  $M$ 
```

a) Demonstrați că algoritmul returnează cuplajul de cardinal maxim dacă G este arbore.

b) Demonstrați că dacă G are un unic cuplaj perfect M_0 , atunci algoritmul returnează M_0 .

c) Arătați că pentru cuplajul M returnat are loc

$$|M| \geq \frac{\nu(G)}{2}. \quad (2+2+2 = 6 \text{ puncte})$$

Problema 3. Fie $G = (V, E)$ un graf conex și $c : E \rightarrow \mathbb{R}$ o funcție de cost cu valori reale asociate muchiilor.

a) Fie T_0 un arbore parțial al lui G cu proprietatea că pentru orice muchie $e \in E(T_0)$ există un arbore parțial de cost minim T^* al lui G astfel încât $e \in E(T^*)$. Rezultă că T_0 este arbore parțial de cost minim al lui G ? (argumentați răspunsul)

(b) Pentru $T \in \mathcal{T}_G$ definim $b(T) = \max_{e \in E(T)} c(e)$. Se dorește aflarea unui arbore $T_0 \in \mathcal{T}_G$ astfel încât $b(T_0) = \min_{T \in \mathcal{T}_G} b(T)$. Dacă aplicăm algoritmul lui Prim pentru aflarea unui arbore parțial de cost minim T^* , este acesta o soluție pentru problema dată? (argumentați răspunsul)

(2+2 = 4 puncte)

Setul de probleme 9

Problema 1. Fie $G = (S, T; E)$ un graf bipartit. Utilizați teorema lui Hall pe un graf convenabil pentru a demonstra că pentru orice întreg k , cu $0 \leq k \leq |S|$, graful G are un cuplaj de cardinal cel puțin $|S| - k$ dacă și numai dacă $\forall A \subseteq S \ |N_G(A)| \geq |A| - k$. **(2 puncte)**

Problema 2. Numim cuplaj *de grad maxim* în graful G , un cuplaj M cu suma gradelor virfurilor saturate de M maximă printre toate cuplajele grafului.

a) Arătați că un cuplaj de grad maxim este de cardinal maxim **(2 puncte)**

b) Dem. că există în graful G un cuplaj care saturează toate virfurile de grad maxim dacă și numai dacă orice cuplaj de grad maxim are aceeași proprietate. **(2 puncte)**

c) Demonstrați că dacă mulțimea virfurilor de grad maxim ale grafului G induce un graf bipartit, atunci G are un cuplaj care saturează toate virfurile de grad maxim. **(2 puncte)**

d) Deduceți că mulțimea muchiilor unui graf bipartit G poate fi partiționată în $\Delta(G)$ cuplaje. **(2 puncte)**

Problema 3. Considerăm următoarea problemă de decizie:

Instanță: $G = (V, E)$ un graf, $k \in \mathbb{N}$, $b \in \mathbb{N}^*$.

Intrebare : Există în G un subgraf H cu b muchii, fără virfuri izolate și cu ordinul lui H cel puțin k ?

Arătați că problema se poate rezolva în timp polinomial. **(2 puncte)**

Problema 4. Arătați, utilizând teorema lui Tutte, că orice graf 2-muchie conex 3-regulat are un cuplaj perfect. **(2 puncte)**

Setul de probleme 10

Problema 1. Fie G un graf conex cu n virfuri și \mathcal{T}_G familia arborilor săi parțiali. Se consideră graful $H = (\mathcal{T}_G, E(H))$ unde $T_1 T_2 \in E(H) \iff |E(T_1) \Delta E(T_2)| = 2$.

a) Demonstrați că H este conex și are diametrul cel mult $n - 1$. **(2 puncte)**

b) Demonstrați că pentru orice funcție de cost c pe mulțimea muchiilor grafului G , mulțimea arborilor parțiali de cost c minim induce un subgraf conex în H . **(2 puncte)**

Problema 2. Fie $H = (V, E)$ un digraf și $ts \in E$ un arc fixat al său. Se colorează toate arcele lui H cu galben, roșu și verde arbitrar, cu singura condiție ca arcul ts să fie galben (se poate întâmpla ca să nu avem arce roșii sau verzi). Demonstrați algoritmic că are loc exact una din următoarele situații:

i) există un circuit în graful $G(H)$ (nu se ține seama de orientare) cu arce galbene sau verzi care conține arcul ts și toate arcele galbene ale sale au aceeași orientare.

ii) există o partiție (S, T) a lui V astfel încât $s \in S, t \in T$, toate arcele de la S la T sunt roșii și toate arcele de la T la S sunt roșii sau galbene.

(2 puncte)

Problema 3. Fie $G = (V, E)$ un graf. O mulțime de virfuri $A \subseteq V$ se numește *m-independentă* dacă există un cuplaj M al lui G astfel încât $A \subseteq S(M)$. Demonstrați că dacă A și B sunt mulțimi m-independente și $|A| < |B|$, atunci $\exists b \in B - A : A \cup \{b\}$ este m-independentă (mulțimile m-independente maximale au același cardinal). **(4 puncte)**

Problema 4. Cuplaje stabile in grafuri bipartite
Fie graful complet bipartit $K_{n,n} = (B, F; E)$, unde $B = \{b_1, b_2, \dots, b_n\}$ și $F = \{f_1, f_2, \dots, f_n\}$. Dacă M este un cuplaj perfect în $K_{n,n}$ (fiecare b este cuplat cu exact un f), vom folosi notația : $b_i f_j \in M \iff f_j = M(b_i) \iff b_i = M(f_j)$. Vom presupune că

$\forall b \in B$ are o ordonare a preferințelor sale pe F :

$f_{i_1} <_b f_{i_2} <_b \dots <_b f_{i_n}$ și

$\forall f \in F$ are o ordonare a preferințelor sale pe B :

$b_{i_1} <_f b_{i_2} <_f \dots <_f b_{i_n}$.

Un cuplaj perfect M al lui $K_{n,n}$ se numește *stabil* dacă :

$\forall b \in B$ dacă $f <_b M(b)$, atunci $M(f) <_f b$ și, de asemenea,

$\forall f \in F$ dacă $b <_f M(f)$, atunci $M(b) <_b f$.

Să se arate că pentru orice ordonări ale preferințelor există un cuplaj stabil și să se construiască unul în $\mathcal{O}(n^3)$.

(4 puncte)

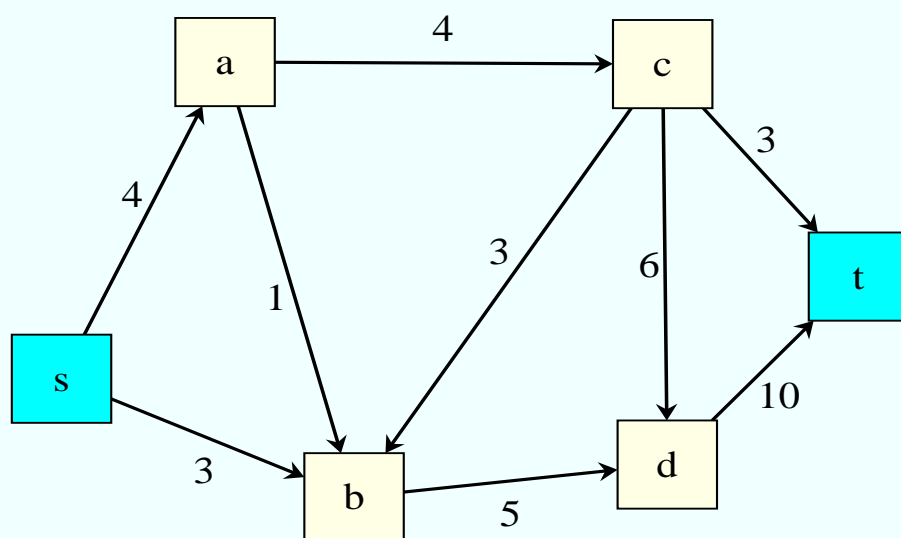
Setul de probleme 11

Problema 1. Se dispune de un algoritm care primind la intrare un graf G și o funcție de pondere nenegativă pe mulțimea muchiilor acestuia, returnează un cuplaj perfect în graful G de pondere minimă (printre toate cuplajele perfecte ale grafului; dacă G nu are cuplaj perfect se anunță acest lucru). Arătați că se poate utiliza acest algoritm pentru determinarea eficientă a cuplajului de cardinal maxim într-un graf oarecare. **(3 puncte)**

Problema 2. Arătați că se poate determina, într-o matrice cu elemente 0 și 1 dată, o mulțime de cardinal maxim de elemente egale cu 0 și care să nu se găsească pe aceeași linie sau coloană, cu ajutorul unui algoritm de flux maxim (pe o rețea convenabil definită). **(3 puncte)**

Problema 3. Digraful $G = (V, E)$ descrie topologia interconectării într-o rețea de procesoare. Pentru fiecare procesor $v \in V$ se cunoaște încărcarea sa $load(v) \in \mathbb{R}^+$. Se cere să se determine (cu ajutorul unei probleme de flux maxim) un plan de *echilibrare statică* a încărcării procesoarelor : se va indica pentru fiecare procesor ce cantitate de încărcare va trimite și la ce procesor astfel încât, în final, toate procesoarele să aibă aceeași încărcare. **(4 puncte)**

Problema 4. Să se determine fluxul de valoare maximă în rețeaua din figura de mai jos (explicind funcționarea algoritmului lui Edmonds-Karp):



(Etichetele arcelor reprezintă capacitățile)

(4 puncte)

Setul de probleme 11'

Problema 1. Considerăm următoarele probleme de decizie:

PERF *Instanță:* G un graf.
 Intrebare: Are G un cuplaj perfect?

3PERF *Instanță:* G un graf cu gradul fiecărui vârf ≤ 3 .
 Intrebare: Are G un cuplaj perfect?

Demonstrați că **PERF** se reduce polinomial la **3PERF** (4 puncte)

Problema 2. Demonstrați că nu există nici o permutare e_1, e_2, \dots, e_{10} a muchiilor grafului complet K_5 , astfel încât pentru orice $i \in \{1, \dots, 9\}$ muchiile e_i și e_{i+1} nu sunt adiacente în K_5 și, de asemenea, e_1 și e_{10} nu sunt adiacente în K_5 . (2 puncte)

Problema 3. Un organizator al unei conferințe trebuie să asigure fețe de masă (curate) pentru fiecare din cele D zile cât durează conferința.

Se cunoaște numărul M_i al meselor de care e nevoie în ziua i a conferinței ($i = 1, D$). Se consideră că toate cele M_i fețe de masă se murdăresc la sfârșitul zilei i ($i = 1, D$).

Organizatorul are de ales între a cumpăra fețe de masă noi, la prețul unitar p , sau, în dimineața zilei i , să trimită la curățat fețe de masă murdare (din zilele precedente; $i \geq 2$). Curățătoria are două tipuri de servicii: *serviciul rapid*, prin care se returnează fețele de masă curate la începutul zilei $i + 1$ la un cost unitar c_1 , și *serviciul lent* prin care returnează fețele de masă curate la începutul zilei $i + 2$ la un cost unitar c_2 . Desigur, $p > c_1 > c_2$.

Problema pe care și-o pune organizatorul este de a face o planificare a modului de cumpărare și trimitere la curățătorie a fețelor de masă, astfel încât să satisfacă toate cererile pe durata conferinței, la un preț minim.

(Se presupune că nu există fețe de mese în stoc, la începutul conferinței, și că valoarea acestora după terminarea conferinței e neglijabilă).

Să se formuleze problema organizatorului ca o problemă de flux de cost minim (justificare). (**4 puncte**)

Problema 4. O euristică naturală pentru colorarea vârfurilor unui graf $G = (V, E)$ este următoarea:

a) Se alege o D -ordonare a lui G , adică o ordonare $V = \{v_{i_1}, v_{i_2}, \dots, v_{i_n}\}$ astfel încât $d_G(v_{i_1}) \geq d_G(v_{i_2}) \geq \dots \geq d_G(v_{i_n})$.

b) Se colorează *greedy* vârfurile: lui v_{i_1} i se atribuie culoarea 1 și apoi pentru fiecare vârf v_{i_j} , cu $j = 2, \dots, n$, se atribuie cea mai mică culoare posibilă (cel mai mic număr natural p cu proprietatea că nu a fost atribuit drept culoare unuia dintre vecinii săi deja colorați).

Considerăm următoarea problemă de decizie:

3GCOL *Instanță:* G un graf.
Intrebare: Există o D -ordonare a vârfurilor lui G astfel încât
euristica de mai sus dă o 3-colorare a lui G ?

Demonstrați că problema

3COL *Instanță:* G un graf.
Intrebare: Admite G o 3-colorare ?

se reduce polinomial la **3GCOL**. (4 puncte)

Setul de probleme 11''

Problema 1. Fie $R = (G, s, t, c)$ o rețea (G digraful suport, $s \in V(G)$ intrarea, $t \in V(G)$, $t \neq s$ ieșirea și $c : E(G) \rightarrow \mathbf{R}_+$ funcția de capacitate). Presupunem (fără a restrânge generalitatea !) că st și ts nu sunt arce în G . Se dispune și de o funcție de *mărginire inferioară* $m : E(G) \rightarrow \mathbf{R}_+$, satisfăcând $m(e) \leq c(e)$ pe orice arc e al lui G . Numim *flux legal în R* orice flux x în R cu proprietatea că $x(e) \geq m(e) \forall e \in E(G)$.

a) Demonstrați că pentru orice flux legal x și orice secțiune (S, T) în R are loc

$$v(x) \leq \sum_{i \in S, j \in T, ij \in E(G)} c(ij) - \sum_{i \in S, j \in T, ji \in E(G)} m(ji).$$

b) Se construiește din R rețeaua \overline{R} astfel:

- se adaugă la G o intrare nouă \overline{s} și o ieșire nouă \overline{t} ;
- pentru $\forall v \in V(G)$ se adaugă arcul $\overline{s}v$ de capacitate $\overline{c}(\overline{s}v) = \sum_{uv \in E(G)} m(uv)$;
- pentru $\forall v \in V(G)$ se adaugă arcul $v\overline{t}$ de capacitate $\overline{c}(v\overline{t}) = \sum_{vu \in E(G)} m(vu)$;
- se adaugă arcele st și ts de capacitate $\overline{c}(st) = \overline{c}(ts) = \infty$;
- se definește \overline{c} pe arcele ij ale lui G ca fiind $\overline{c}(ij) = c(ij) - m(ij)$.

Demonstrați că există un flux legal în rețeaua R dacă și numai dacă există un flux de valoare $M = \sum_{e \in E(G)} m(e)$ în rețeaua $\bar{R} = (\bar{G}, \bar{s}, \bar{t}, \bar{c})$ (\bar{G} este digraful construit mai sus, \bar{c} este funcția de capacitate definită mai sus).

c) Utilizând un flux legal de start (care se poate obține ca la b)), indicați cum se poate adapta algoritmul lui Ford & Fulkerson pentru a obține un flux legal de valoare maximă într-o rețea în care pe fiecare arc este precizată capacitatea și marginea inferioară.

(2+2+2 puncte)

Problema 2. Dacă H este un graf conex, $A \subseteq V(H)$ o mulțime nevidă de vârfuri ale sale și $w : E(H) \rightarrow \mathcal{R}_+$, atunci se numește *arbore Steiner* corespunzător tripletei (H, A, w) un arbore $T(H, A, w) = (V_T, E_T)$, subgraf al lui H , cu proprietatea că $A \subseteq V_T$ și suma costurilor muchiilor sale, $s[T(H, A, w)] = \sum_{e \in E_T} w(e)$, este minimă printre toți arborii subgrafuri ale lui H care conțin A .

a) Justificați că determinarea lui $T(H, A, w)$ se poate face în timp polinomial pentru cazul când $A = V(H)$ sau $|A| \leq 2$.

b) Fie $G = (V, E)$ un graf conex cu mulțimea de vârfuri $V = \{1, \dots, n\}$ și $A \subseteq V$. Pe mulțimea muchiilor lui G este dată o funcție de cost $c : E \rightarrow \mathcal{R}_+$.

Considerăm și graful complet K_n cu mulțimea de vârfuri V și cu funcția de cost $\bar{c} : E(K_n) \rightarrow \mathcal{R}_+$ dată de $\bar{c}(ij) = \min_{[P \text{ drum în } G \text{ de la } i \text{ la } j]} c(P)$ pentru orice $ij \in E(K_n)$. Demonstrați că $s[T(G, A, c)] = s[T(K_n, A, \bar{c})]$ și că din orice arbore Steiner $T(K_n, A, \bar{c})$ se poate construi un arbore Steiner $T(G, A, c)$.

c) Arătați că există un arbore Steiner $T(K_n, A, \bar{c})$ cu proprietatea că vârfurile sale care nu-s din A au gradul cel puțin 3. Deduceți (folosind această proprietate) că există întotdeauna un arbore Steiner $T(K_n, A, \bar{c})$ cu cel mult $2|A| - 2$ vârfuri.

$((1+1)+(1+2)+(2+1)$ puncte)

Setul de probleme 12

Problema 1. Fie v valoarea fluxului maxim în rețeaua $R = (G, c, s, t)$. Demonstrați că există k st -drumuri în G , P_1, \dots, P_k ($0 \leq k \leq |E(G)|$), și numerele reale nenegative v_1, \dots, v_k , astfel încât $x : E(G) \rightarrow \mathbb{R}$, definit pentru orice arc ij prin $x_{ij} = 0 + \sum_{t:ij \in P_t} v_t$, este flux în R de valoare maximă v . **(4 puncte)**

Problema 2. Numim *GP-descompunere* a grafului K_n orice mulțime $\mathcal{A} = \{B_1, \dots, B_{k(\mathcal{A})}\}$, unde : fiecare B_i este un subgraf bipartit complet al lui K_n , orice două grafuri B_i și B_j au mulțimile de muchii disjuncte și $\cup_{i=1, k(\mathcal{A})} E(B_i) = E(K_n)$. Arătați că orice GP-descompunere \mathcal{A} a lui K_n satisface inegalitatea $k(\mathcal{A}) \geq n - 1$. **(4 puncte)**

Problema 3. Fie $G = (V, E)$ un graf și $f : V \rightarrow V$ cu proprietatea că $\forall uv \in E : f(u)f(v) \in E$. Demonstrați că $\omega(G) \leq |f(V)|$. Este adevărat că pentru orice graf $G = (V, E)$ există funcții f cu proprietatea de mai sus și astfel încât $|f(V)| \leq \Delta(G) + 1$? **(4 puncte)**

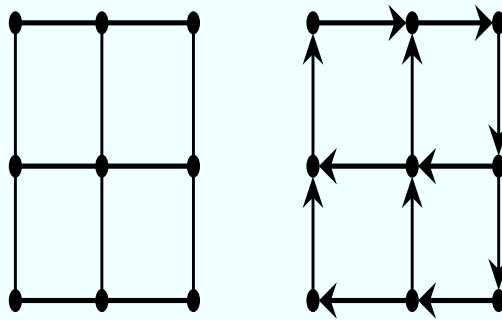
Problema 4. Fie $G = (V, E)$ un graf. Numim *partiție specială* orice bipartiție (S, T) a lui V astfel încât subgraful indus de T în G este neconex și subgraful indus de S în complementarul grafului G este neconex.

Arătați că graful circuit C_n ($n \geq 3$) nu are partiții speciale.

Descrieți un algoritm polinomial care să testeze dacă un graf dat are partiții speciale. **(2 puncte)**

Setul de probleme 13

Problema 1. Consiliul municipal al unui orașel a decis să elimine blocajele de circulație de pe străzile acestuia prin introducerea sensului unic pe fiecare stradă. Desigur, va trebui ca între orice două locații să existe (măcar) un drum de acces după această decizie. Dacă reprezentăm graful străzilor folosind noduri pentru intersecțiile stradale și muchii conectând aceste noduri corespunzător străzilor, o situație simplificată este dată de următoarea figură :



Desigur, graful real al străzilor este mult mai complicat și de aceea a fost angajat un expert (student la info) care, analizându-l, a observat că are proprietatea că este 2-conex (orice intersecție s-ar bloca, singurele locații afectate sunt cele de pe străzile din acea intersecție) și a propus următorul algoritm de orientare a muchiilor (fixarea sensului unic pentru fiecare stradă):

1. Se alege un nod oarecare și se execută o parcurgere **dfs** etichetând nodurile de la 0 la $n-1$ (n e numărul de noduri din graf) în ordinea întâlnirii lor;
2. Fiecare muchie (stradă) este orientată de la nodul cu etichetă mai mică la nodul cu etichetă mai mare dacă acea muchie face parte din arborele **dfs** construit, și de la nodul cu etichetă mai mare la nodul cu etichetă mai mică, în caz contrar.

Arătați că algoritmul funcționează corect pe graful din figura de mai sus și apoi demonstrați că algoritmul este corect (digraful obținut este tare conex) pentru orice graf 2-conex. **(1+3 puncte)**

Problema 2. Demonstrați că orice graf G conex are un subgraf indus H astfel încât :

1. H este conex;
2. $|H| = 2|S| - 1$, unde S este o mulțime stabilă a grafului H ;
3. $\forall v \in V(G) - V(H) \exists w \in V(H)$ astfel încât $vw \in E(G)$.

(2 puncte)

Problema 3. La sfârșitul unei zile de lucru, laborantul a observat că a disparut un mouse din laborator, deși la verificarea de dimineață nu lipsea. Din registrul sălii a rezultat că în acea zi au intrat în laborator doar următorii șase studenți: Ana, Barbu, Costică, Dan, Elena și Ion. De asemenea, se știe că fiecare din ei a stat în laborator un interval de timp și apoi a plecat (dar nu se cunosc orele de venire sau plecare și nici ordinea în care cei șase au accesat laboratorul). Decanul i-a anchetat, și a obținut următoarele mărturii (sub jurământ):

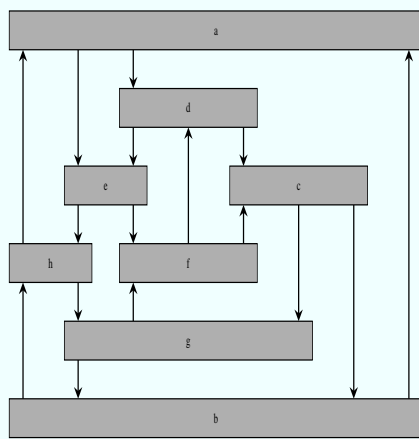
1. *Ana a spus că i-a văzut în laborator pe Barbu și Elena;*
2. *Barbu a spus că i-a văzut în laborator pe Ana și Ion;*
3. *Costică a spus că i-a văzut în laborator pe Dan și Ion;*
4. *Dan a spus că i-a văzut în laborator pe Ana și Ion;*
5. *Elena a spus că i-a văzut în laborator pe Barbu și Costică;*
6. *Ion a spus că i-a văzut în laborator pe Costică și Elena.*

Analizând răspunsurile, decanul (care știa teoria grafurilor) a intuit că exact unul (hoțul) dintre cei șase studenți a mințit, apoi, folosind deducția logică, l-a identificat și i-a cerut să aducă de urgență mouse-ul înapoi pentru a nu-l exmatricula.

Știind că decanul a asociat fiecărui student un interval de timp (notat, A, B, \dots , după numele lor) și că dintr-o mărturie " $X... spune că i-a văzut pe Y... și Z...$ " a dedus că intervalul X se intersectează cu Y și Z , evidențiați inconsistența din graful asociat intersecțiilor acestor intervale și cum se poate depista hoțul, în ipoteza că exact unul dintre cei șase studenți a mințit.

(2+2 puncte)

Problema 4. Un grup de polițiști își desfășoară activitatea în 8 locații a, b, \dots, h conectate prin străzi cu sens unic de circulație, așa cum este descris în digraful de mai jos. Se observă că în fiecare nod (locație) intră două arce și din fiecare ies două arce. De asemenea se observă că digraful corespunzător este tare conex.



Șeful polițiștilor a hotărât să vopsească străzile roșu și oranj astfel încât din fiecare nod să plece un arc roșu și un arc oranj. Scopul acestei decizii a fost ca atunci când într-o locație oarecare se întâmplă o infracțiune, să-i adune operativ pe toți polițiștii în acel nod al digrafului transmițându-le tuturor prin sistemul lor de radio-recepție mesajul "*Adunarea*" și un cuvânt din $\{r, o\}^*$ (cuvânt ce depinde doar de nodul în care loc infracțiunea).

La primirea mesajului fiecare polițist își notează cuvântul primit și îl folosește drept algoritm de deplasare astfel:

din nodul în care se află pleacă pe strada roșie sau pe strada oranj, după cum e prima literă (r sau o) din cuvântul primit. Apoi, dacă n-a ajuns la locul de adunare (pe care-l recunoaște după prezența șefului), alege strada indicată de a doua literă a cuvântului și așa mai departe.

Descrieți (într-un pseudocod prietenos) un algoritm care să depisteze o vopsire roșu-oranj a arcelor digrafului și, corespunzător acesteia, a câte unui cuvânt de rutare pentru fiecare nod de adunare (sau să decidă că nu există soluție).

Pentru digraful din figură există soluții! Descrieți una din ele și argumentați că merge (se poate implementa algoritmul descris, sau se poate folosi o abordare *try and error*).

(2+2 puncte)

Setul de probleme 14

Problema 1. În problema P2 - a determinării drumurilor de cost minim de la un vârf dat, s , la toate celelalte ale unui digraf $G = (V, E)$ - se știe că funcția de cost asociată arcelor satisface $a : E \rightarrow \{0, 1, \dots, C\}$, unde C este constantă întreagă (adică nu depinde de $n = |V|$ sau de $m = |E|$).

Să se adapteze algoritmul lui Dijkstra pentru această situație, astfel încât complexitatea timp să fie $O(n + m)$. Se vor descrie structurile de date folosite și modul (argumentare !) în care se obține complexitatea liniară.

(2+2 puncte)

Problema 2. Pentru o instanță \mathcal{C} a problemei 2SAT construim multigraful $G_{\mathcal{C}} = (V; E)$ ale cărui muchii sunt colorate R(oșu) și B(leu), astfel:

1. $V \leftarrow \emptyset; R \leftarrow \emptyset; B \leftarrow \emptyset;$
2. **for** $C \in \mathcal{C}$ **do** {
 if $C = x_{\alpha} \vee x_{\beta}$ **then** $\{V \leftarrow V \cup \{x_{\alpha}, x_{\beta}\}; R \leftarrow R \cup \{x_{\alpha}x_{\beta}\}\};$
 if $C = \bar{x}_{\alpha} \vee \bar{x}_{\beta}$ **then** $\{V \leftarrow V \cup \{x_{\alpha}, x_{\beta}\}; B \leftarrow B \cup \{x_{\alpha}x_{\beta}\}\};$
 if $C = \bar{x}_{\alpha} \vee x_{\beta}$ **then**
 $\{V \leftarrow V \cup \{x_{\alpha}, x_{\beta}, x_C\}; R \leftarrow R \cup \{x_{\beta}x_C\}; B \leftarrow B \cup \{x_Cx_{\alpha}\}\};$
 if $C = x_{\alpha} \vee \bar{x}_{\beta}$ **then**
 $\{V \leftarrow V \cup \{x_{\alpha}, x_{\beta}, x_C\}; R \leftarrow R \cup \{x_{\alpha}x_C\}; B \leftarrow B \cup \{x_Cx_{\beta}\}\};$
 }
3. $E \leftarrow R \sqcup B$; **output** $G_{\mathcal{C}} = (V; E)$.

Observăm că V conține mulțimea X a variabilelor booleene care apar în 2-clauzele lui \mathcal{C} și pentru fiecare 2-clauză $C \in \mathcal{C}$ compusă dintr-un literal pozitiv și unul negativ (clauză mixtă) se adaugă un vârf nou la V . Notăția $R \sqcup B$ semnifică faptul că muchiile $e \in R \cap B$ au multiplicitate 2 [extremitățile lui e sunt unite printr-o muchie R (oșie) și una B (leu)].

Demonstrați că \mathcal{C} este satisfiabilă dacă și numai dacă există $S, T \subseteq V$ astfel încât: $S \cap T = \emptyset$, $S \cup T = V$, nu există muchii R (oșii) cu ambele extremități în S și nu există muchii B (leu) cu ambele extremități în T .

(2 +2 puncte)

Problema 3. Fie $G = (V, E)$ un graf conex și $c : E \rightarrow \mathbf{R}$. Pentru un arbore parțial oarecare $T = (V, E')$ al lui G , și două vârfuri oarecare $v, w \in V$, se notează cu $v \xrightarrow{T} w$ unicul drum de la v la w în T și cu $E(v \xrightarrow{T} w)$ mulțimea muchiilor acestuia. Demonstrați că arborele parțial $T^* = (V, E^*)$ este arbore parțial de cost minim dacă și numai dacă

$$\forall e = vw \in E \setminus E^*, \forall e' \in E(v \xrightarrow{T^*} w) \text{ are loc } c(e) \geq c(e').$$

(1+1 puncte)

Problema 4. Fie $G = (V, E)$ un graf conex fără punți și $c : E \rightarrow \mathbb{R}$. Fie $T = (V, E')$ un arbore parțial al lui G de cost minim și $e \in E'$ o muchie oarecare a sa. $T - e$ are exact două componente conexe cu mulțimile de vârfuri V_1 și V_2 . Muchia de cost minim (diferită de e) dintre toate muchiile lui G cu o extremitate în V_1 și cealaltă în V_2 se notează cu $rep_T(e)$ (deoarece e nu e punte în G , $rep_T(e)$ există !).

a) Demonstrați că dacă T^* e arbore parțial de cost minim în G și e este o muchie oarecare a lui T^* atunci $T_1^* = T^* - e + rep_{T^*}(e)$ este arbore parțial de cost minim în $G - e$ (T_1^* se obține din T^* scoțând muchia e și adăugând muchia $rep_{T^*}(e)$).

b) Fie $k \in \mathbb{N}$ astfel încât $2 \leq k < |V|$ și T_1, \dots, T_k arbori parțiali ai lui G astfel încât $c(T_1) \leq c(T_2) \leq \dots \leq c(T_k)$ și pentru orice alt arbore parțial T al lui G avem $c(T) \geq c(T_k)$ (T_1, \dots, T_k sunt primii cei mai mici arbori parțiali ai lui G în raport cu costul c).

Fie T^* un arbore parțial de cost minim în G . Pentru fiecare muchie e a lui T^* considerăm ponderea $w(e) = c(rep_{T^*}(e)) - c(e)$. Sortăm cele $|V| - 1$ muchii ale lui T^* crescător în raport cu ponderile w . Fie S mulțimea formată din ultimile $|V| - k$ muchii din acest șir.

Demonstrați că $S \subseteq E(T_i) \forall i \in \{1, \dots, k\}$.

(2+2 puncte)

Setul de probleme 15

Problema 1. *Demonstrați că numărul cuplajelor perfecte ale unui arbore este 0 sau 1 și că un arbore are un cuplaj perfect dacă și numai dacă prin îndepărtarea oricărui vârf se obține o pădure cu exact un arbore de ordin impar.* (2+2 puncte)

Problema 2. Fie $R = (G, s, t, c)$ o rețea și (S_i, T_i) ($i = 1, 2$) secțiuni de capacitate minimă ale ei. *Demonstrați că și $(S_1 \cup S_2, T_1 \cap T_2)$ și $(S_1 \cap S_2, T_1 \cup T_2)$ sunt secțiuni de capacitate minimă în R .* (3+1 puncte)

Problema 3. Fie $G = (S, T; E)$ un graf bipartit și M un cuplaj de cardinal maxim în G . Considerăm următoarele mulțimi de vârfuri:

1. $P = \{v \in S \cup T \mid \exists w \in E(M) \text{ și un drum alternat (relativ la } M \text{ în } G) \text{ de lungime pară de la } w \text{ la } v\};$
2. $I = \{v \in S \cup T \mid \exists w \in E(M) \text{ și un drum alternat (relativ la } M \text{ în } G) \text{ de lungime impară de la } w \text{ la } v\};$
3. $N = \{v \in S \cup T \mid \text{nu } \exists w \in E(M) \text{ și un drum alternat (relativ la } M \text{ în } G) \text{ de la } w \text{ la } v\}.$

Demonstrați că :

a) Mulțimile P, I și N sunt disjuncte două câte două și sunt aceleași pentru orice cuplaj de cardinal maxim M .

b) În orice cuplaj de cardinal maxim al lui G fiecare vârf din I este cuplat cu un vârf din P și fiecare vârf din N este cuplat cu un alt vârf din N . Cardinalul maxim al unui cuplaj al grafului este $|I| + \frac{|U|}{2}$.

(2+2 puncte)

Problema 4. O firmă de soft dispune de n programatori, P_1, P_2, \dots, P_n , pentru executarea a m lucrări, L_1, L_2, \dots, L_m . Se cunoaște pentru fiecare programator P_i lista \mathcal{L}_i de lucrări pe care le poate executa și numărul s_i al lucrărilor din \mathcal{L}_i pe care le poate termina într-o săptămână ($s_i \leq |\mathcal{L}_i|$). Fiecare lucrare poate fi executată de măcar un programator.

Să se descrie cum se poate determina numărul minim de săptămâni în care se pot termina toate lucrările, folosind fluxurile în rețele.

(2 puncte)

Setul de probleme 16

Problema 1. Fie $G = (V, E)$ un graf cu n vârfuri $\{v_1, \dots, v_n\}$ și $c : E \rightarrow \mathbf{R}_+$ o funcție de capacitate nenegativă, care asociază fiecărei muchii e capacitatea $c(e)$. Se numește secțiune în G orice partiție cu două clase (S, T) a lui V . Capacitatea secțiunii (S, T) este $c(S, T) = \sum_{e \in E, |e \cap S|=1} c(e)$. O secțiune minimă în G este o secțiune (S_0, T_0) astfel încât

$$c(S_0, T_0) = \min_{(S, T) \text{ secțiune în } G} c(S, T).$$

a) Să se arate că se poate determina în timp polinomial o secțiune minimă în graful G rezolvând un număr polinomial de probleme de flux maxim pe rețele convenabil alese.

b) Arătați că dacă $G = C_n$ (graful circuit de ordin $n \geq 3$) cu toate muchiile de capacitate 1, atunci există $\frac{n(n-1)}{2}$ secțiuni de capacitate minimă. (2+2 = 4 puncte)

Problema 2. În continuarea notațiilor de la problema 1, definim pentru orice pereche $i, j \in \{1, \dots, n\}$, $i \neq j$,

$$c'_{i,j} = \min_{(S, T) \text{ secțiune în } G \text{ cu } v_i \in S \text{ și } v_j \in T} c(S, T).$$

a) Demonstrați că pentru orice șir i_1, i_2, \dots, i_k de $k \geq 3$ elemente distincte din $\{1, \dots, n\}$ are loc

$$c'_{i_1, i_k} \geq \min\{c'_{i_1, i_2}, c'_{i_2, i_3}, \dots, c'_{i_{k-1}, i_k}\}.$$

b) Se consideră graful K_n cu mulțimea de vârfuri $\{1, \dots, n\}$ și funcția de pondere pe muchiile sale c' definită mai sus (notăm că $c'_{i,j} = c'_{j,i}$). Fie T^* un arbore parțial de pondere maximă al lui K_n (în raport cu ponderea c'). *Demonstrați că $\forall i, j \in \{1, \dots, n\}, i \neq j$, dacă P este unicul drum de la i la j în T^* , atunci*

$$c'_{i,j} = \min_{e \in P} c'(e).$$

Observație: Rezultă că există un arbore cu mulțimea de vârfuri V , cu ponderi pe muchii astfel încât, pentru a determina capacitatea minimă a unei secțiuni în graful G care separă două virfuri, determinăm muchia de pondere minimă de pe drumul ce unește cele două vârfuri în arbore. **(2+3 = 5 puncte)**

Problema 3.

Se consideră o competiție sportivă între n echipe $\{e_1, \dots, e_n\}$, în care fiecare echipă dispută $a \geq 1$ meciuri cu fiecare dintre celelalte $n - 1$ echipe (deci, fiecare echipă va juca $a(n - 1)$ meciuri în total). Orice meci se termină cu victoria uneia dintre cele două echipe participante (nu există remize). Se dorește să se decidă dacă este posibil ca, la finalul competiției, fiecare echipă e_i să câștige un număr de c_i meciuri (vectorul de întregi $c[1 \dots n]$ este intrarea problemei de decizie).

Arătați ca problema se poate rezolva în timp polinomial cu ajutorul fluxurilor pe o rețea convenabil definită. **(3 puncte)**

Problema 4. În rețeaua $R = (G, s, t, c)$, toate capacitățile nenule sunt numere întregi pozitive pare.

Demonstrați că există un flux x de valoare maximă cu proprietatea că pe orice arc, dacă fluxul este nenul atunci el este un număr pozitiv par. **(2 puncte)**

Setul de probleme 17

Problema 1. Fie $G = (S, T; E)$ un graf bipartit cu $n = |V(G)|$ vârfuri și $m = |E|$ muchii.

- a) Demonstrați că $m \leq \frac{n^2}{4}$.
- b) Demonstrați că, dacă $B_{n \times m}$ este matricea de incidență a lui G , atunci orice submatrice pătrată C a lui B are proprietatea că $\det(C) \in \{-1, 0, 1\}$.
- c) Se orientează arbitrar muchiile lui G și se obține digraful \vec{G} . Demonstrați că există $K \subseteq V(G) = S \cup T$ astfel încât K e mulțime stabilă în \vec{G} și oricare ar fi $v \in V(G) - K$, există $u \in K$ cu proprietatea că $uv \in E(\vec{G})$.

(2+2+2 = 6 puncte)

Problema 2. Fie $G = (V, E)$ un graf de ordin n și $s, t \in V$ astfel încât $d_G(s, t) > \frac{n}{2}$. Demonstrați că există $v \in V - \{s, t\}$ cu proprietatea că orice drum de la s la t în graful G trece prin v . Descrieți un algoritm de complexitate timp $O(n + |E|)$ care să determine acest vârf v .

(2+2 = 4 puncte)

Problema 3. a) Modificați algoritmul BFS astfel încât pentru un graf $G = (V, E)$ dat și $s \in V$ să determine pentru orice vârf $v \in V$ numărul drumurilor de lungime minimă de la s la v , în timpul $O(|V| + |E|)$.
b) Aceeași problemă pentru cazul în care G este digraf!

(2+2 = 4 puncte)

Setul de probleme 18

Problema 1. Fie $G = (V, E)$ un graf conex cu n vârfuri și m muchii și fie $c : E \rightarrow \mathbf{R}_{>0}$ o funcție de cost pe muchiile sale.

a) Fie $T = (V, E_T)$ un arbore parțial al lui G cu proprietatea că pentru orice muchie $e \in E_T$ există un arbore parțial de cost minim $T^* = (V, E_{T^*})$ astfel încât $e \in E_{T^*}$. Adevărat sau fals? : " T este arbore parțial de cost minim" (pentru răspunsul adevărat dați o demonstrație, pentru răspunsul fals dați un contraexemplu).

b) Fie R și S doi arbori parțiali ai lui G , $R \neq S$. Cum se poate construi un șir de lungime minimă de arbori parțiali T_0, T_1, \dots, T_k astfel încât $T_0 = R$, $T_k = S$ și fiecare arbore T_i ($i \geq 1$) se obține din precedentul, T_{i-1} , prin ștergerea unei muchii și adăugarea alteia? Care este complexitatea timp a construcției?

c) Pentru orice arbore parțial $T = (V, E_T)$ al lui G se definește "costul" său ca fiind produsul costurilor muchiilor sale: $c(T) = \prod_{e \in E_T} c(e)$. Descrieți un algoritm cât mai eficient care să determine T^* , arbore parțial al lui G , astfel încât

$$c(T^*) = \max_{T \text{ arbore parțial al lui } G} c(T).$$

(2+2+2 = 6 puncte)

Problema 2. Considerăm următoarea problemă de decizie:

AGM **Input:** $G = (V, E)$ graf, $k \in \mathbb{N}$.

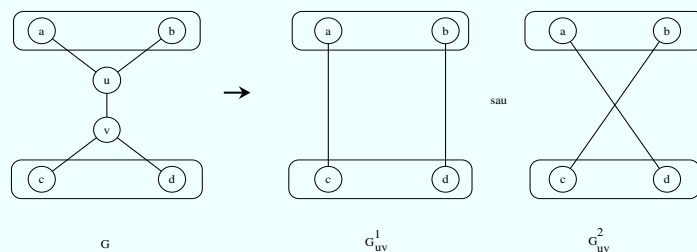
Question: Există un arbore parțial al T al lui G cu $\Delta(T) \geq k$?

Arătați că **AGM** \in P.

(2 puncte)

Problema 3. Fie $G = (V, E)$ un graf 3-regulat, conex și fără punți.

a) Fie $uv \in E$ o muchie oarecare a lui G ca în figura de mai jos. Se elimină cele două vârfuri u și v , iar vecinii lor se conectează prin muchii așa cum este indicat în figură. Demonstrați că măcar unul din grafurile G_{uv}^1 sau G_{uv}^2 este 3-regulat, conex și fără punți. Se notează cu G' acest graf.



b) Arătați că dacă G' are un cuplaj perfect M , atunci M se poate transforma într-un cuplaj perfect al lui G .

c) Deduceți că pentru orice graf 3-regulat, conex și fără punți se poate construi un cuplaj perfect. Ce complexitate timp are construcția, dacă G are n vârfuri ?

(2+2+2 = 6 puncte)

Setul de probleme 19

Problema 1. Se consideră rețeaua $R = (G, c, s, t)$ cu digraful $G = (V, E)$ având n vârfuri și m arce, $c : E \rightarrow \mathbf{Z}_+$ și $C \in \mathbf{Z}_+$, $C = \max_{e \in E} c(e)$.

a) Demonstrați că valoarea maximă a unui flux în rețeaua R este cel mult $m \cdot C$.

b) Arătați că $\forall x$ flux în R , dat $K \in \mathbf{Z}_+$, se poate depista un drum de creștere P de capacitate reziduală $\delta(P)$ cel puțin K (dacă el există), în timpul $O(m)$.

c) Considerăm următorul algoritm

SC-MAX-FLOW(R)

$C = \max_{e \in E} c(e)$

$x \leftarrow 0$ // x este fluxul curent din rețea

$K \leftarrow 2^{1+\lfloor \log C \rfloor}$

while $K \geq 1$ **do** {

while x are un dr. de creșt. P cu $\delta(P) \geq K$ **do** {

$x \leftarrow x \otimes \delta(P)$ }

$K \leftarrow K/2$ }

return x

1. Demonstrați că algoritmul **SC-MAX-FLOW**(R) de mai sus construiește un flux x de valoare maximă în R .

2. Demonstrați că, după fiecare iterație a buclei while exterioare, valoarea maximă a unui flux în R este cel mult $v(x) + m \cdot K$.

3. Demonstrați că pentru fiecare valoare a lui K , numărul iterațiilor buclei while interioare nu depășește $2m$. Deduceți că algoritmul are complexitatea $O(m^2 \log C)$.

(2+2+1+2+2 = 9 puncte)

Problema 2. Considerăm următoarea problemă de decizie:

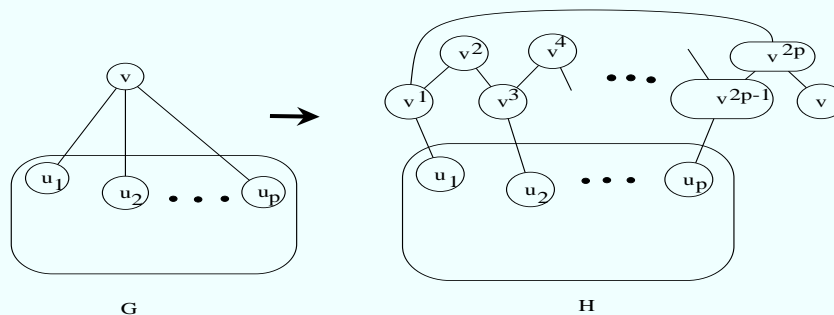
MIN-SECȚIUNE-UNICĂ

Input: $R = (G, c, s, t)$, rețea.

Question: Există în R o unică secțiune de capacitate minimă?

Arătați că **MIN-SECȚIUNE-UNICĂ** $\in P$. (2 puncte)

Problema 3. a) Fie $G = (V, E)$ un graf și $v \in V$ un vârf cu gradul $d_G(v) = p \geq 4$. Fie $N_G(v) = \{u_1, \dots, u_p\}$. Construim graful H astfel: (1) se șterg din G muchiile vu_1, \dots, vu_p ; (2) se adaugă la G circuitul C_{2p} cu vârfurile v^1, \dots, v^{2p} ; (3) se adaugă la graful obținut muchiile $v^{2i-1}u_i$ pentru $i \in \{1, \dots, p\}$, și muchia vv^{2p} (vezi figura de mai jos). Demonstrați că $\alpha(H) = \alpha(G) + p$.



b) Fie **SM3** problema de decizie obținută din **SM** (vezi pag. 283 din curs) prin restricționarea instanței la un graf cu gradul maxim cel mult 3. Demonstrați că **SM** \propto **SM3**.

(1+2 = 3 puncte)

Setul de probleme 20

Problema 1. Fie $G = (V, E)$ un graf cu $V = \{1, \dots, n\}$. Considerăm funcția $f : V \rightarrow V$ definită de

$$f(v) = \min\{u \mid u \in V, d_G(v, u) \leq 2\}.$$

- a) Demonstrați că dacă există $vw \in E(G)$ astfel încât $f(v) \neq f(w)$ atunci G conține graful P_4 ca subgraf indus.
b) Demonstrați că dacă G este P_4 -free atunci două vârfuri oarecare u și v sunt în aceeași componentă conexă dacă și numai dacă $f(u) = f(v)$. **(2+2 = 4 puncte)**

Problema 2. Numim **graf bicolor** perechea (G, m) , unde $G = (V(G), E(G))$ este un graf, iar $m : E(G) \rightarrow \{alb, verde\}$ este o funcție care asociază fiecărei muchii $e \in E(G)$ o culoare $m(e) \in \{alb, verde\}$. Graful bicolor (G, m) este **verzui** dacă în orice circuit al lui G numărul muchiilor verzi este par.

- a) Demonstrați că dacă (K_n, m) este verzui atunci există mulțimile A și B de vârfuri astfel încât $V(K_n) = A \cup B$ (reuniune disjunctă) și pentru orice muchie $e = uv \in E(K_n)$ avem:

$$m(e) = \begin{cases} alb & \text{dacă } u, v \in A \text{ sau } u, v \in B \\ verde & \text{altfel.} \end{cases}$$

b) Demonstrați că dacă (G, m) este verzui și $n = |G|$ atunci se pot adăuga muchii grafului G , extinzând colorarea m la aceste noi muchii, astfel încât să obținem graful bicolor verzui (K_n, m) .

c) Descrieți un algoritm care primind la intrare graful bicolor (G, m) determină în timpul $O(|V(G)| + |E(G)|)$ dacă este sau nu verzui.

(2+2+2 = 6 puncte)

Problema 3. Există aplicații în care se dă un digraf $G = (V, E)$ și $a : E(G) \rightarrow \mathbf{R}_+$ și trebuie să răspundem repetat la întrebări de tipul: *Care este drumul minim dintre s și t ?* ($s, t \in V(G), s \neq t$)

Pentru situația când G este un digraf mare, se propune aplicarea algoritmului lui Dijkstra (costurile fiind nenegative) dar **bidirecțional**: se construiește G' inversul lui G , în care orice arc $(i, j) \in E(G')$ are costul $a'_{ij} = a_{ji}$; se aplică apoi succesiv câte un pas al algoritmului lui Dijkstra pentru G, s și funcția de cost a și apoi un pas pentru G', t și funcția de cost a' ; la depistarea unui vârf u marcat definitiv (introdus în S) de cele două instanțe ale algoritmului lui Dijkstra, se transformă drumul de la t la u din G' depistat de a doua instanță într-un drum D_1 în G de la u la t , și se returnează drumul de la s la u depistat de prima instanță și drumul D_1 .

Dați un contraexemplu pentru algoritmul propus!

(2 puncte)

Problema 4.

Un *graf interval* este definit de o mulțime finită și nevidă de intervale de pe axa reală. Pentru fiecare interval avem un vârf în graful interval și două vârfuri sunt adiacente dacă intervalele corespunzătoare lor au intersecția nevidă.

Descrieți un algoritm care primind la intrare n intervale (închise) cu extremitățile numere întregi din intervalul $[1..2n]$ determină în timpul $O(n)$ componentele conexe ale grafului interval definit de acestea.

(2 puncte)

Setul de probleme 21

Problema 1. Fie $G = (V, E)$ un graf conex cu n vârfuri și m muchii. Fiecare muchie $e \in E$ are asociat un cost real $c(e)$, iar costurile muchiilor sunt distincte.

- a) Descrieți un algoritm care să decidă în timpul $O(n + m)$ dacă, pentru o muchie dată $e_0 \in E$, există un arbore parțial de cost minim al lui G care să conțină muchia e_0 .
b) Argumentați corectitudinea algoritmului propus. (2+2=4 puncte)

Problema 2. Pentru graful conex $G = (V, E)$ cu n vârfuri, m muchii și funcția de cost $c : E \rightarrow \mathbf{R}$, se cunoaște lista muchiilor (cu costurile aferente) și un arbore parțial de cost minim $T = (V, E_T)$. Arborele T este reprezentat de vectorul *parent* cu n componente, în care pentru orice $v \in V - \{s\}$ avem $\text{parent}[v] =$ vârful dinaintea lui v de pe unicul drum din T de la vârful s la v , iar $\text{parent}[s] = s$ (s este un vârf oarecare, fixat). Se dorește să se actualizeze lista muchiilor și arborele T la operațiile de adăugare și de ștergere a unei muchii. *Se cere să se proiecteze algoritmi de complexitate $O(n + m)$ pentru fiecare din aceste operații (cu justificarea corectitudinii și a complexității timp).*

Mai precis, acești algoritmi au:

- a) La intrare o muchie $e = uv$ nouă și costul ei, iar la ieșire noua listă de muchii ($E \cup \{e\}$) și vectorul *parent* reprezentând un arbore parțial de cost minim în $G + e$.

b) La intrare o muchie $e = uv$ existentă în graful G , iar la ieșire noua listă de muchii $(E - \{e\})$ și vectorul *parent* reprezentând un arbore parțial de cost minim în $G - e$ (dacă acest arbore parțial mai există; altfel se returnează mesajul că graful nu mai este conex).

(2+2 = 4 puncte)

Problema 3. Fie T un arbore cu un număr par de vârfuri care nu are cuplaj perfect. Demonstrați că există în T un vârf v_0 cu proprietatea că stergând v_0 din arborele T (și toate muchiile incidente cu v_0) se obține o pădure cu măcar doi arbori cu un număr impar de vârfuri.

(2 puncte)

Problema 4. Un cuplaj M al unui graf G se numește **cuplaj tare** dacă subgraful indus în G de mulțimea vârfurilor saturate de M este graful $|M|K_2$. Proiectați un algoritm de complexitate $O(n)$ pentru găsirea unui cuplaj tare de cardinal maxim într-un arbore cu n vârfuri. Argumentați corectitudinea acestui algoritm.

(2+2= 4 puncte)

Setul de probleme 22

Problema 1. Fie G un graf cu proprietatea că nu are circuite impare disjuncte. Demonstrați că G este 5-colorabil. (2 puncte)

Problema 2. Graful $G = (V, E)$ cu n vârfuri și m muchii modelează o rețea de comunicație bidirecțională: în fiecare vârf este plasat un utilizator care schimbă mesaje cu ceilalți utilizatori folosind muchiile incidente cu vârful în care este localizat (un sistem de rutare asociat rețelei va asigura ajungerea mesajelor la destinatari). Într-un vârf special $s \in V$ se află un utilizator care trimite spamuri. Pentru fiecare muchie $\{u, v\} \in E$ a grafului se cunoaște costul $c(\{u, v\})$ al instalării unui filtru care să blocheze orice spam ce s-ar putea transmite pe muchia $\{u, v\}$ (de la u la v sau de la v la u). Se dorește ca o mulțime $A \subseteq V - \{s\}$ de utilizatori să fie protejată de spamurile trimise de utilizatorul din s .

Se cere să se proiecteze un algoritm cu timp de lucru polinomial care să determine o mulțime de muchii $E' \subseteq E$ pe care să se plaseze filtrele care să protejeze utilizatorii din A și să aibă cost minim (suma costurilor muchiilor din E' să fie minimă printre toate mulțimile ce pot proteja A). Justificare.

(2+2 = 4 puncte)

Problema 3. Un *nucleu* într-un digraf $G = (V, E)$ este o mulțime S de vârfuri cu proprietatea că S nu conține vârfuri adiacente și pentru orice vârf $u \in V - S$ există $v \in S$ astfel încât $(v, u) \in E$. Fie problema de decizie:

NUCLEU INSTANȚĂ: $G = (V, E)$ digraf.
 ÎNTREBARE: Are G un nucleu?

Demonstrați că următoarea construcție oferă o reducere polinomială a problemei **SAT** la problema **NUCLEU** ($SAT \propto NUCLEU$).

Dată o instanță F a lui **SAT** construim digraful G astfel:

- pentru fiecare clauză C a lui F adăugăm la G un 3-circuit $v_C^1, (v_C^1, v_C^2), v_C^2, (v_C^2, v_C^3), v_C^3, (v_C^3, v_C^1), v_C^1$;
 - pentru fiecare variabilă x care apare în formula F adăugăm la G un 2-circuit $v_x, (v_x, v_{\bar{x}}), v_{\bar{x}}, (v_{\bar{x}}, v_x), v_x$;
 - pentru fiecare pereche (l, C) unde l este un literal (x sau \bar{x}) care apare în clauza C adăugăm la G arcele $(v_l, v_C^1), (v_l, v_C^2), (v_l, v_C^3)$.
- (2+2=4 puncte)**

Problema 4. Fiecare student dintr-o mulțime S de $n > 0$ studenți optează pentru o submulțime de 4 cursuri opționale dintr-o mulțime C de $k > 4$ cursuri opționale. Se cere să se descrie un algoritm care să determine în timp polinomial o alocare a studenților la cursurile opționale astfel încât fiecare student să fie alocat la exact 3 cursuri opționale (din cele patru pentru care a optat) și fiecare curs opțional să aibă repartizat cel mult $\lceil \frac{3n}{k} \rceil$ studenți (dacă o astfel de alocare există).

Justificare. **(2+2= 4 puncte)**