



CLUJ-NAPOCA
2009

TEODOR TOADERE

GRAFE teorie, algoritmi si aplicatii

Reeditare

CUPRINS

Editura Albastră

Director editură
Smaranda Derveșteanu

Coperta
SD Special Design

Tipărit
EDITURA ALBASTRĂ

Editura este acreditată de CNCSIS
(Consiliul Național al Cercetării Științifice din Învățământul Superior)
și este recomandată
Consiliului Național de Atestare a Titlurilor și Diplomelor Universitare.

Copyright © 2009

Toate drepturile asupra acestei lucrări aparțin s.c. Casa de Editură Albastră s.r.l.
Reproducerea integrală sau parțială a textului sau a ilustrațiilor din această carte
este posibilă numai cu acordul prealabil în scris al Casei de Editură Albastră.



INTRODUCERE 5

1. NOIUNI DE BAZĂ.....	11
1.1. Noiuni preliminarii.....	11
1.2. Reprezentări ale grafelor.....	32
1.2.1. Reprezentarea geometrică	32
1.2.2. Reprezentarea matriceală	34
1.2.3. Reprezentarea cu tablouri unidimensionale	39
1.3. Numere fundamentale în teoria grafelor	50
1.3.1. Număr de stabilitate internă	50
1.3.2. Număr de stabilitate externă.....	60
1.3.3. Nucleul unui graf	67
1.3.4. Număr cromatic.....	69
1.4. Conexitate	75
1.4.1. Componențe conexe	75
1.4.2. Număr ciclomatic și număr cociclomatic	84
1.4.3. Arbori și păduri	89
1.4.4. Arbore de pondere minimă.....	100
1.5. Grafe particulare	105
1.5.1. Grafe planare	105
1.5.2. Grafe perfecte.....	109
2. SPAȚII LINIARE ASOCIAȚE GRAFELOR.....	112
2.1. Spațiu liniar al părților unei mulțimi	112
2.2. Subspațiuul ciclurilor	114
2.3. Subspațiuul cociclurilor	117
3. DRUMURI ÎN GRAFE.....	121
3.1. Optimizări de drumuri în ipoteza $I(\mu) = \sum_{u \in \mu} I(u)$	123
3.1.1. Valori minime de drumuri de la un vîrf dat	124
3.1.2. Algoritmi matriciali pentru valori minime	135
3.1.3. Determinarea valorilor maxime ale drumurilor	138

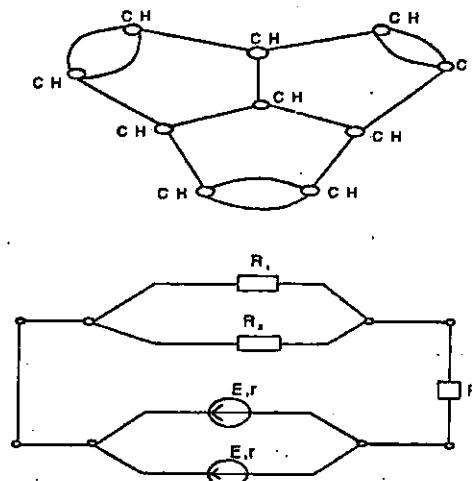
3.2.	Optimizări de drumuri în ipoteza $I(\mu) = \prod_{u \in \mu} I(u)$	142
3.3.	Problema ordonanțării. Drum critic	144
3.3.1.	Graful potențiale-activități	144
3.3.2.	Graful potențiale-etape (PERT)	147
3.4.	Problema comis-voiajorului	148
4.	FLUXURI ȘI REȚELE DE TRANSPORT	151
4.1.	Definiții și proprietăți de bază	151
4.2.	Lema arcelor colorate (Lema lui Minty)	153
4.3.	Problema fluxului maxim într-o rețea de transport	156
4.4.	Tăietură într-o rețea de transport	157
4.5.	Algoritmul lui Ford-Fulkerson	160
4.6.	Modelarea problemei orarului ca flux în grafe	168
4.7.	Extensiile ale algoritmului lui Ford-Fulkerson	173
4.8.	Fluxuri de cost minim	175
5.	CUPLAJE ÎN GRAFE	184
BIBLIOGRAFIE		195

INTRODUCERE

În foarte multe situații obișnuim să descriem unele obiecte (sisteme), pentru a le studia sau prezenta, cu ajutorul unor reprezentări sub forma unor scheme. Asemenea situații se întâlnesc în:

- cartografie;
- studiul unor rețele (căi ferate, rutiere, canalizări, telefonie) economie (diagrame);
- fizică;
- chimie și biochimie (reprezentări de molecule și celule);
- geometrie și topologie;
- proiectarea pe scară largă a circuitelor integrate (VLSI);
- informatică etc.

Aceste modele ale sistemelor (problemelor), de regulă, sunt reprezentări de multigrafuri sau multigrafe. Schemele din figura 1 sunt câteva exemple.



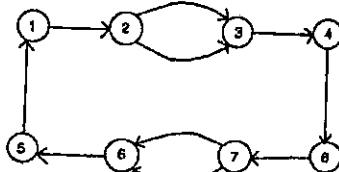


Fig.1.

Precizăm că în limba română pentru substantivul *graf* există în circulație atât pluralul *grafe* cât și cel de *grafuri*. Grafele, fiind multigrafe particulare, sunt utilizate în modelare pentru studiul unor sisteme sau pentru rezolvarea unor probleme din diferite domenii. Pe de altă parte, multigrafele sunt obiecte abstrakte studiate (investigate) cu un aparat matematic propriu, devenind un domeniu al matematicii. Astfel, realități concrete din domenii diferite pot să se reducă la (să fie modelate cu) o aceeași problemă de grafe. Teoria grafelor este o ramură relativ Tânără a matematicii. Ea își are originile în rezolvarea unor probleme aparent neînsemnante pentru dezvoltarea matematicii. Problemele care au condus la apariția și dezvoltarea teoriei grafelor au fost probleme de jocuri și amuzamente matematice menite să testeze mai mult ingeniozitatea rezolvitorilor. Astfel prima problema din teoria grafelor, rezolvată, este problema celor șapte poduri sau problema podurilor din Königsberg.

Prin orașul Königsberg trece râul Pregel care după ce înconjoară insula Kneiphof se desparte în două brațe. Astfel există patru zone de pământ A, B, C, D, care sunt udate de apa râului. Acestea comunică cu ajutorul a șapte poduri numerotate de la 1 la 7 ca în desenul alăturat.

Fiind un loc de promenadă locuitorii și-au pus problema următoare:

Este posibil ca plecând dintr-o zonă oarecare să se parcurgă un traseu astfel ca acesta să treacă peste toate cele șapte poduri, trecând o singură dată peste fiecare pod și să se ajungă în zona de pornire?

Data nașterii teoriei grafelor se consideră a fi anul 1736 când matematicianul Leonhard Euler a scris un articol în care a clarificat această problemă și a dat o metodă pentru rezolvarea altor probleme de același tip [Eul36]. Pentru rezolvarea problemei el și-a construit multigraful din figura 2.

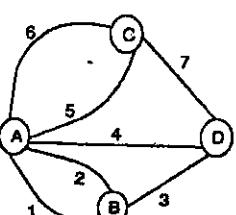
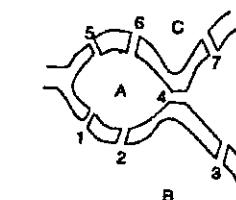


Fig.2.

Probleme înrudite cu problema celor șapte poduri sunt problemele de labirint. O problemă de labirint cere să se determine un traseu, într-un sistem de coridoare, care pornește de la un punct dat (intrarea în labirint) și ajunge la un alt punct dat (iesirea din labirint).

În secolul trecut multe rezultate în teoria grafelor au fost date de Cayley, iar Kirchhoff a studiat circuitele electrice cu ajutorul grafelor. Prima carte de teoria grafelor a apărut abia în anul 1936, editată de D. König, deși articole din acest domeniu au mai fost publicate anterior de o serie de matematicieni. O istorie completă a lucrărilor de teoria grafelor din perioada 1736-1936 este prezentată în [BLW76]. Interesul pentru teoria grafelor a crescut foarte mult în ultimele decenii. Aceasta se datorează aplicațiilor matematicii din diferite domenii cum ar fi: economie, chimie, armată, fizică, etc. Teoria grafelor, printre altele, ne ajută să alegem un traseu optim dintr-o mulțime de variante posibile. Pentru realizarea unui obiectiv în timp scurt, teoria grafelor ne ajută să determinăm momentele de abordare a activităților de realizat (ordonanțare). Pentru stabilirea unor programe de transport optime teoria grafelor ne furnizează algoritmi performanți. În teoria astfel dezvoltată graful este un obiect abstract, dar ale cărui elemente sunt asignate unor realități concrete. Recurgerea la aceste modele fiind utilă datorită algoritmilor cu care se pot rezolva diferitele probleme concrete. Dintre cercetătorii care s-au ocupat de teoria grafelor în ultimele decenii cităm pe: C. Berge, S. Cataranciuc, V. Cepoi, E.Ciurea, C. Croitoru, L.R. Ford, M. Gondran, F.Harary, L. Lovász, O.Ore, E.Olariu, Al. Roșu, V.P.Soltan, I. Tomescu, W.T. Tutte, H.-J. Voss, T.Zamfirescu.

Pentru a putea fi aplicată în domenii variate și datorită faptului că este un domeniu al matematicii teoria grafelor are acum un grad înalt de abstractizare și formalizare, oferind un număr mare de modele și metode general aplicabile. În acest spirit o vom prezenta și noi în cele ce urmează.

Teoria grafelor se poate utiliza în rezolvarea a foarte multe probleme concrete. Aplicarea sa depinde și de capacitatea rezolvitorului de a asocia problemei sale un graf și de a-și formula problema în termenii teoriei grafelor.

Materialul prezentat este structurat în cinci capitulo. Cu excepția capitolului doi, care are un caracter pur teoretic, fiecare capitol prezintă: noțiuni (obiecte ale teoriei grafelor), rezultate teoretice (leme, teoreme și corolarii), probleme și algoritmi de rezolvare a acestora. De asemenea, sunt prezentate și aplicații practice a căror rezolvare se poate face prin modelarea lor cu ajutorul unor elemente din teoria și problematica specifică grafelor.

Capitolul 1 Prezintă principalele definiții referitoare la:

- multigrafe, p -grafe și grafe orientate sau neorientate;

- drum și lanț cu cazurile particulare de circuit respectiv ciclu;
- grafe eulariene și hamiltoniene, conexe și tare conexe, arbori, păduri și arborescențe;
- moduri de reprezentare a grafelor, mulțimi interior și exterior stabilă;
- numere fundamentale în teoria grafelor;
- un spațiu liniar al ciclurilor unui graf;
- grafe planare și perfecte.

Algoritmi din acest capitol se referă la:

- parcursarea grafelor, conversia reprezentării unui graf dintr-o formă în altă formă, determinarea componentelor conexe ale unui graf, determinarea unei păduri maximale sau a unui arbore de pondere minimă pentru un graf dat;
- determinarea de cicluri eulariene sau hamiltoniene;
- determinarea de mulțimi interior sau exterior stabilă și a numerelor de stabilitate.

Este prezentat un tip (o clasă) de *algoritmi backtracking și branch-and-bound* cu exemplificare pentru determinarea de mulțimi și numere de stabilitate internă sau externă.

Capitolul 2 Acest capitol are un caracter doar teoretic și prezintă un mod de a defini spații liniare sau subspații liniare pornind de la un graf dat peste corpul B_2 . Este vorba despre spațiu liniar al submulțimilor unei mulțimi, subspațiul ciclurilor unui graf, subspațiul cociclurilor unui graf dat.

Capitolul 3 În acest capitol prezentăm câteva probleme și algoritmi de rezolvare a acestora pe mulțimile de drumuri ale unui graf dat. Ca aplicații prezentăm problema *ordonanțării (drum citic)* și problema *comis voiajorului*.

Capitolul 4 După prezentarea și studiul noțiunilor: flux într-un graf, rețea, rețea de transport, flux într-o rețea de transport și tăietură, se prezintă problema și algoritmi pentru *determinarea fluxului maxim într-o rețea de transport*. Apoi, sunt date unele aplicații ale acestei probleme (problemele de afectare și cea a orarului). În partea a doua se prezintă problema și algoritmi pentru *determinarea unui program de transport de cost minim* într-o rețea care are doar vârfuri sursă sau vârfuri destinație.

Capitolul 5 După prezentarea principalelor noțiuni (cuplaj, lanț alternant, arbore alternant de rădăcină r) și rezultate teoretice se dă problema *determinării unui cuplaj maxim* și *algoritmul lui Edmonds* de rezolvare a acesteia.

1. NOȚIUNI DE BAZĂ

1.1. Noțiuni preliminarii

Definiția 1.1.1. Se numește *multigraf orientat* orice sistem de forma (X, U) , notat cu G , în care X este o mulțime de elemente oarecare, iar $U \subset X \times X \times \mathbb{N}$.

Elementele mulțimii X le numim *vârfurile* multigrafului, iar elementele mulțimii U le numim *arcele* multigrafului. Într-un multigraf cea de-a treia componentă a unui arc are rol doar de numerotare a arcelor care nu pot fi diferențiate altfel. Numerotarea, convenim, să o facem de la 1 pentru fiecare pereche ordonată de vârfuri. În unele lucrări G este notat cu (V, E) de la cuvintele englezesti *vertex* (vârf) și *edge* (arc).

Dacă mulțimile X și U sunt finite, atunci multigraful G se zice că este *finit* și în acest caz cardinalul mulțimii X , notat cu $n = |X|$, se numește *ordinul multigrafului*. Cardinalul mulțimii U se notează cu m și se numește *dimensiunea multigrafului* G . În cele ce urmează vom considera doar multigrafe finite. Deoarece nu ne interesează natura vârfurilor multigrafulor dacă $|X| = n$ convenim ca X să fie $\{1, 2, \dots, n\}$, ceea ce înseamnă că vârfurile au fost numerotate. Dacă $(i, j, k) \in U$, atunci spunem că arcul respectiv este *incident spre exterior* vârfului i , este *incident spre interior* vârfului j , are sensul de la vârful i la vârful j . Pentru arcul $u = (i, j, k)$ vârful i este extremitatea inițială, iar vârful j este extremitatea finală. Deci în precizarea unui arc contează ordinea vâfurilor. Dacă extremitățile unui arc coincid atunci arcul se numește *bucăță*. Astfel multigraful reprezentat în figura 1 are:

- ca mulțime de vârfuri pe:
$$X = \{1, 2, 3, 4, 5, 6, 7, 8\};$$
- ca mulțime de arce pe:
$$U = \{(1, 2, 1); (2, 3, 1); (2, 3, 2); (3, 4, 1); (4, 8, 1); (8, 7, 1); (7, 6, 1); (7, 6, 2); (6, 5, 1); (5, 1, 1)\}.$$

Un alt exemplu este $G = (X, U)$, unde X este mulțimea oamenilor dintr-o anumită zonă geografică (deci vârf este orice om din X), iar $U = \{(i, j) \mid i, j \in X \text{ și } j \text{ este nepot al lui } i\}$.

Multigraful este un model abstract pentru un sistem oarecare, deci o abstractizare, o închijuire a omului relativă la acel sistem. Pentru o percepere vizuală a grafelor se utilizează reprezentarea lor într-o formă geometrică. În această reprezentare vârfurile le corespund puncte sau cercuri etichetate sau nu din plan, iar arcele se reprezintă prin segmente orientate (vezi figura 1). Există și alte moduri de a reprezenta multigrafele fiecare mod având avantaje și dezavantaje. Adică, influențează eficiența algoritmilor în rezolvarea problemelor. Reprezentarea geometrică este utilă mai mult când dorim să dăm o percepție vizuală a multigrafului, deci se utilizează mai mult pe hârtie sau cu creta pe tablă.

Dacă între două vârfuri există un arc atunci spunem că *vârfurile sunt adiacente*. De asemenei, două arce distințe sunt *adiacente* dacă au o extremitate comună.

Definiția 1.1.2. Un multigraf G se numește *p-graf* dacă între oricare două vârfuri ale sale există cel mult p arce care să aibă același sens și G are două vârfuri între care există exact p arce care au același sens.

Multigraful reprezentat în figura 1 este un 2-graf.

Definiția 1.1.3. Un 1-graf se numește *graf orientat*.

În acest caz se consideră $U \subset X \times X$.

Să observăm că mulțimea U poate fi privită ca o relație binară pe mulțimea vârfurilor grafului.

Exemplul 1.1.1.:

$$G = (X, U) \text{ cu } X = \{1, 2, 3, 4, 5\} \text{ și} \\ U = \{(1, 2); (2, 1); (2, 3); (3, 5); (3, 4); (4, 3); (4, 5); (5, 1)\}.$$

Vom da alte definiții echivalente pentru noțiunile de multigraf orientat și de graf orientat.

Definiția 1.1.4. Prin *multigraf orientat* înțelegem un sistem $G = (X, E, f)$, în care X este o mulțime de elemente numite *vârfurile* multigrafului, E este o mulțime de elemente numite *arcele* multigrafului cu $E \cap X = \emptyset$, iar $f: E \rightarrow X \times X$ este o funcție.

Deoarece multigraful îl considerăm un obiect abstract, mulțimea E se poate considera că este formată din acele numere naturale cu care s-au numerotat arcele multigrafului. Faptul că $f(k) = (i, j)$ se interpretează astfel:

arcul $k \in E$ ieșe din i și intră în j .

Această definiție a permis introducerea noțiunii de *hipergraf* în care elementele lui U sunt mulțimi de vârfuri nu neapărat de cardinal 2.

Definiția 1.1.5. Prin *graf orientat* înțelegem un sistem $G = (X, \Gamma)$, unde X este o mulțime oarecare, a cărui elemente se numesc *vârfuri ale grafului*, iar Γ este o *aplicație multivocă* $\Gamma: X \rightarrow X$ (adică o funcție de la X la mulțimea $\mathcal{P}(X)$ a părților lui X).

Pentru $i \in X$ și relația $j \in \Gamma i$ interpretăm că de la vârful i la vârful j există un arc în graful G , spunem că j este *succesor* al lui i și că i este *predecesor* al lui j .

Exemplul 1.1.2.:

a. $G = (X, \Gamma)$ cu $X = \{1, 2, 3, 4, 5\}$ și $\Gamma 1 = \{2, 3\}$, $\Gamma 2 = \{1, 3\}$, $\Gamma 3 = \emptyset$, $\Gamma 4 = \{5, 2\}$, $\Gamma 5 = \emptyset$.

b. $G = (X, U)$, unde X este mulțimea celor 64 pătrătele de pe o tablă de șah, iar mulțimea $U = \{(i, j) \mid$ piesă de șah cal de pe i se poate deplasa printr-o mutare pe $j\}$. Să remarcăm acest graf este neorientat.

c. $G = (X, U)$ cu X mulțimea elevilor unei școli și $U = \{(i, j) \mid i$ și j sunt colegi de clasă}. Si acest graf este un graf neorientat ca și graful precedent.

d. $G = (X, \Gamma)$ unde X este mulțimea de activități ce trebuie realizate pentru a realiza un obiectiv (a atinge un scop), iar pentru fiecare i definim $\Gamma i = \{j \mid$ activitatea j poate să înceapă cel mai devreme imediat după realizarea activității $i\}$. Acest graf este un graf orientat.

Observații:

a. Dacă $G = (X, \Gamma)$ atunci graful respectiv se poate preciza și prin (X, U) în care $U = \{(i, j) \mid i \in X$ și $j \in \Gamma i\}$.

b. Analog cazului a dacă $G = (X, U)$, atunci el este definit și de (X, Γ) cu $\Gamma: X \rightarrow X$ și $\Gamma i = \{j \in X \mid (i, j) \in U\}$, $\forall i \in X$.

Folosind observațiile de mai sus indiferent de forma în care se dă un graf vom putea face referire și la cealaltă formă.

Mulțimea Γ^i se numește **mulțimea succesorilor** lui i . Pentru $A \subset X$, ΓA este:

$$\Gamma A = \bigcup_{i \in A} \Gamma^i.$$

Folosind operația de compunere a aplicațiilor, pentru un $n \in \mathbb{N}$, ne putem referi la mulțimea $\Gamma^n i$ definită astfel:

$$\Gamma^n i = \begin{cases} \{i\}, & n = 0 \\ \Gamma(\Gamma^{n-1} i), & n \geq 1 \end{cases}$$

Mulțimea

$$\Gamma^+ i = \bigcup_{n \geq 1} \Gamma^n i,$$

este formată din toți *descendenții* lui $i \in X$, iar aplicația Γ^+ se numește *închiderea tranzitivă* a aplicației Γ .

Definim

$$\hat{\Gamma} i = \bigcup_{n \geq 0} \Gamma^n i = \Gamma^+ i \cup \{i\}$$

și aplicația $\hat{\Gamma}$ o numim *închiderea reflexiv tranzitivă* a aplicației Γ .

Fiind dat un graf oarecare $G = (X, U)$ și un vârf $i \in X$:

- notăm cu $g^+(i)$ numărul succesorilor săi, adică pe $|\Gamma^+ i|$, și-l numim *subgradul exterior* al lui i ;
- notăm cu $g^-(i)$ numărul predecesorilor lui i , adică pe $|\Gamma^{-1} i|$, și-l numim *subgradul interior* al lui i ;
- notăm cu $g(i)$ pe $g^+(i) + g^-(i)$ și-l numim *gradul vârfului* i .

Un vârf i cu $g(i) = 0$ se numește *vârf izolat*.

PARCURGEREA GRAFELOR

Fiind dat un graf G , orientat sau nu, se pune problema de a vizita (parcurge) toate vâfurile grafului. Această vizitare se poate face în diferite moduri. Dintre aceste moduri de parcurgere a vâfurilor mai cunoscute sunt două:

- parcurgerea în lățime (*breadth first*);

- parcurgerea în adâncime (*depth first*).

Există o relativă asemănare între aceste parcurgeri ale vâfurilor unui graf cu parcurgerea elementelor unei matrice. Parcurgerea elementelor unei matrice se poate face "pe linii" sau "pe coloane" rămânând de precizat ordinea de selectare a liniilor respectiv a coloanelor.

Idea parcurgerii în lățime este:

"se vizitează vârful de pornire, se vizitează toți succesorii (vecinii) acestuia, apoi vecinii nevizitați ai acestora și așa mai departe până când se vizitează toate vâfurile grafului".

O descriere mai algoritmizată a parcurgerii în lățime, pornind de la vârful i , este:

```
R:=Φ;
S:={i};
repeat
    vizitează fiecare  $j \in S$ ;
     $S := \Gamma S - R$ ;
     $R := R \cup S$ ;
până când  $S = \Phi$ .
```

Parcurgerea în adâncime intuitiv constă în:

"Se vizitează vârful de pornire, apoi se repetă vizitarea unui succesor nevizitat a ultimului vârf vizitat până când nu există succesiuni nevizitate pentru ultimul vârf vizitat, apoi se revine la predecesorul ultimului vârf vizitat (se consideră că ultimul vârf vizitat ca fiind predecesorul care a condus la vizitarea ultimului vârf vizitat). Procesul de schimbare a ultimului vârf vizitat continuă până când se ajunge la vârful de pornire și acesta nu are succesiuni nevizitate".

O descriere a algoritmului de parcurgere în adâncime a unui graf este:

```
vizitează vârful de pornire,
repeat
    căt timp
        există succesor nevizitat al ultimului vârf vizitat vizitează-l,
        sf;
        revenire la predecesorul vârfului ce nu mai are succesiuni nevizitate;
până când vârful de pornire nu mai are succesiuni nevizitate.
```

Dăm în continuare o rafinare a descrierii algoritmilor de parcurgere în lățime și în adâncime a grafelor care construiește ordinea respectivă de vizitare a vâfurilor grafului în sirul *vizit* și folosește ca variabilă de lucru lista liniară x .

Algoritmul 1.1.1.

```

Pentru  $i=1, n$  execută  $vizit[i]:=0$ ;  $sfp$ ;
 $p:=1$ ;  $x[p]:=vârful de pornire$ ;  $vizit[vârful de pornire]:=1$ ;  $i:=1$ ;
Repetă
    Pentru  $j$  succesor al lui  $x[p]$  execută
        Dacă  $vizit[j]=0$  atunci
             $i:=i+1$ ;  $x[i]:=j$ ;
            Vizitează vârful  $j$ ;  $vizit[j]:=1$ ;
        Sfida;
    Sfrepeta;
     $p:=p+1$ ;
până când  $p=n$ .

```

Algoritmul 1.1.2.

```

Pentru  $i=1, n$  execută  $vizit[i]:=0$ ;  $sfp$ ;
 $p:=1$ ;  $x[p]:=vârful de pornire$ ;  $vizit[vârful de pornire]:=1$ ;
Repetă
    Cât timp există  $j$  succesor al lui  $x[p]$  cu  $vizit[j]=0$  execută
         $p:=p+1$ ;  $x[p]:=j$ ;
        Vizitează vârful  $j$ ;  $vizit[j]:=1$ ;
    Sfătimp;
     $p:=p-1$ ;
până când  $p=0$ .

```

Să remarcăm că în cei doi algoritmi lista x este o coadă (listă FIFO) pentru parcurgerea în lățime respectiv o stivă (listă LIFO) pentru parcurgerea în adâncime a grafelor.

Definiția 1.1.6. Multigrafele $G_1 = (X_1, U_1)$ și $G_2 = (X_2, U_2)$ sunt *izomorfe* dacă există bijecțiile $f: X_1 \rightarrow X_2$ și $g: N \rightarrow N$ astfel ca $(i, j, k) \in U_1 \Leftrightarrow (f(i), f(j), g(k)) \in U_2$.

Pentru ca două grafe să fie izomorfe este suficientă doar existența unei funcții $f: X_1 \rightarrow X_2$ cu proprietatea $(i, j) \in U_1 \Leftrightarrow (f(i), f(j)) \in U_2$.

Teorema 1.1.1. Relația de izomorfism în mulțimea grafelor este o relație de echivalență.

Demonstrație:

Deoarece funcția identitate 1_X este bijectivă înseamnă că orice graf G este izomorf cu el însuși. Deci relația de izomorfism este reflexivă.

Dacă grafele G_1 și G_2 sunt izomorfe atunci există o bijecție între mulțimile de vârfuri ale celor două grafe cu proprietatea de conservare a relației de adiacență

a vârfurilor. Știm că orice funcție bijectivă admite o funcție inversă care este tot bijectivă. Funcția inversă conservă de asemenea relația de adiacență, deci și grafele G_2 și G_1 sunt izomorfe. Prin urmare relația de izomorfism a grafelor este simetrică.

Fie perechile de grafe G_1 , G_2 și G_3 , G_3 izomorfe, prin urmare există funcțiile $f_1: X_1 \rightarrow X_2$ și $f_2: X_2 \rightarrow X_3$ bijective cu proprietățile precizate pentru grafe izomorfe. Cum compusa a două bijecții este tot o bijecție și funcția $f_1 \circ f_2: X_1 \rightarrow X_3$ are proprietatea cerută pentru izomorfismul de grafe înseamnă că grafele G_1 și G_3 sunt izomorfe. Deci relația de izomorfism a grafelor este și tranzitivă.

Prin urmare, relația de izomorfism a grafelor este reflexivă, simetrică și tranzitivă, deci este relație de echivalență. \square

Definițiile 1.1.7. Grafe particulare:

1. Un graf este *simplu* dacă nu are bucle.

Observație:

Orice graf care are bucle poate fi transformat într-un graf simplu prin adăugarea de către un vârf k_i pentru fiecare buclă (i, i) și înlocuind bucla cu arcele (i, k_i) și (k_i, i) . Prin urmare, putem considera că toate grafele considerate sunt simple.

2. Un graf orientat $G = (X, U)$ este *simetric* dacă pentru orice vârfuri i și j cu $(i, j) \in U$ atunci și $(j, i) \in U$.

Dacă $G = (X, \Gamma)$ atunci G este simetric dacă și numai dacă $\forall i, j \in X: i \in \Gamma j \Leftrightarrow j \in \Gamma i$.

Observație:

De remarcat că într-un graf simetric nu în mod necesar orice pereche de vârfuri este arc.

3. Un graf orientat este *antisimetric* dacă pentru orice vârfuri i și j cu $(i, j) \in U$ atunci $(j, i) \notin U$.

Observație:

Să observăm că un graf care nu este simetric nu înseamnă că este antisimetric.

4. Un graf orientat este *turnir* dacă este antisimetric și are număr maxim de arce.

Observație:

Altfel spus $\forall i, j \in X (i, j) \in U \Leftrightarrow (j, i) \notin U$.

5. Graful **complementar** al unui graf dat $G = (X, U)$ este graful (X, V) cu $V = X \times X - U - \{(i, i) \mid i \in X\}$.
6. Graful (A, V) este **subgraf** al grafului (X, U) dacă $A \subset X$ și $V = U \cap (A \times A)$. Un asemenea graf se mai numește și **graful generat** de mulțimea A notat $G(A)$.
7. Graful (X, V) este **graf parțial** al grafului (X, U) dacă $V \subset U$.

Observație:

De remarcat că un subgraf este format dintr-o parte de vârfuri ale grafului dat și toate arcele grafului dat ce au ambele extremități în această mulțime de vârfuri, în timp ce un graf parțial are aceeași mulțime de vârfuri ca graful dat și doar o parte dintre arcele grafului dat.

8. Graful (A, V) este **subgraf parțial** al grafului (X, U) dacă și numai dacă $A \subset X$ și $V \subset U \cap (A \times A)$.
9. Graful (X, U) este **bipartit** dacă $\exists A \subset X$ astfel încât $U \subset A \times (X - A)$.

Definiția 1.1.8. Prin **graf neorientat** se înțelege un sistem $G = (X, U)$ cu X o mulțime de elemente numite **vârfurile grafului** și U o mulțime de perechi neordonate de vârfuri.

Un element din U se numește **muchie**. În unele lucrări muchia dintre vârfurile i și j se notează prin $[i, j]$ și poate fi privită ca fiind o submulțime de vârfuri cu două elemente.

Reprezentarea geometrică a muchiilor se deosebește de cea a arcelor prin faptul că lipsește orientarea de la arce. Un exemplu este cel din figura 2.

Analog cazului grafelor orientate, în cazul grafelor neorientate despre o muchie spunem că este incidentă vârfurilor care sunt extremități ale sale. Vârfurile între care există o muchie se numesc **adiacente**. Două muchii care au o extremitate comună se numesc **adiacente**. **Gradul** unui vârf i al unui graf neorientat este numărul de muchii incidente lui și se notează tot cu $g(i)$.

Un graf neorientat se numește **complet** dacă are număr maxim de muchii. Un graf complet de ordin n se notează cu K_n .

Graful bipartit complet (X, U) cu $U = A \times (X - A)$, $A \subset X$, $|A| = p$ și $|X - A| = q$ se notează cu $K_{p,q}$.

Un subgraf parțial complet al unui graf se numește **clică**.

Problema 1.1.1. Fie X o mulțime cu n elemente. Să se determine:

- a. numărul grafelor neorientate, fără bucle, care au ca mulțime de vârfuri pe X ;
- b. numărul grafelor orientate, simple, care au ca mulțime de vârfuri pe X ;
- c. numărul grafelor orientate, simple și antisimetrice, ce au pe X ca mulțime de vârfuri;
- d. numărul turnirurilor ce au pe X ca mulțime de vârfuri;
- e. numărul grafelor simple care au gradele vâfurilor distincte.

Rezolvare:

Cum X este dată (fixă) numărul grafelor $G = (X, U)$ este egal cu numărul mulțimilor U care fac ca G să aibă proprietatea cerută.

- a. Pentru grafele neorientate, U este mulțime de muchii adică de perechi neordonate de vârfuri sau de submulțimi cu 2 elemente ale lui X . Prin urmare mulțimea tuturor muchiilor peste X , pe care o notăm cu M , are cardinalul egal cu numărul combinărilor de $n = |X|$ luate câte 2 = C_n^2 . Cum numărul grafelor neorientate este egal cu numărul submulțimilor lui M , deducem că acest număr este 2^b , cu $b = C_n^2$.
- b. Pentru grafele orientate, U este mulțime de arce adică de perechi ordonate de vârfuri. Prin urmare mulțimea tuturor arcelor peste X , pe care o notăm cu A , este $X \times X - \{(p, p) \mid p \in X\}$ numărul arcelor fiind, deci egal cu numărul aranjamentelor de $n = |X|$ luate câte 2 = A_n^2 . Cum numărul grafelor orientate este egal cu numărul submulțimilor lui A , deducem că acest număr este egal cu 2^a , cu $a = A_n^2$.
- c. Cum grafele orientate și antisimetrice se pot obține din cele neorientate prin orientarea muchiilor (transformarea muchiilor în arce). Descriem un algoritm care să facă această transformare și care ne permite să determinăm numărul grafelor orientate și simetrice bazându-ne pe numerele precizate la punctul a. Deoarece orice pereche ordonată (arc) se poate obține dintr-o mulțime de două elemente $\{t, s\}$ (muchie) prin scrierea ei fie (t, s) fie (s, t) . Cum natura vârfurilor nu contează vom considera $X = \{1, 2, \dots, n\}$ și transformarea muchiei în arc este echivalentă cu scrierea vârfurilor t și s fie în ordine crescătoare fie în ordine descrescătoare. Din fiecare graf neorientat cu k muchii se aleg p muchii pe care le orientăm, depinzând de valoarea acestora, de la vârful

mai mic spre vârful mai mare, iar pe celelalte $k - p$ muchii le orientăm invers (de la vârful mai mare spre vârful mai mic). Cum orice graf antisimetru (orientat) cu k arce se poate obține cu acest procedeu din graful neorientat cu cele k arce considerate ca muchii, deducem că numărul de grafe orientate antisimetrice ce se pot obține dintr-un graf neorientat cu k muchii este $C_k^0 + C_k^1 + \dots + C_k^k = 2^k$ pentru că valoarea lui p poate fi $0, 1, \dots, k$. Dacă notăm cu $b = C_n^2$ numărul maxim de muchii, deducem că numărul de grafe cu k muchii este C_b^k . Din acestea se obțin $C_b^k 2^k$ grafe antisimetrice (toate cu k arce) și pentru că valoarea lui k poate fi $0, 1, \dots, b$ numărul grafelor antisimetrice este

$$C_b^0 2^0 + C_b^1 2^1 + \dots + C_b^b 2^b = (2+1)^b = 3^b.$$

- d. Cum turnurile sunt grafe antisimetrice cu număr maxim de arce din rezolvarea de la punctul b deducem că numărul turnurilor este $C_b^b 2^b = 2^b$ adică este egal cu numărul grafelor neorientate. Această egalitate se deduce și din faptul că între mulțimea turnurilor și mulțimea grafelor orientate se pot defini bijecții. O asemenea bijecție se definește astfel:

pentru fiecare turn se elimină arcele orientate de un vârf de valoare mică spre un vârf de valoare mare, iar celelalte arce se transformă în muchii.

- e. Pentru că X este cu n elemente deducem $0 \leq g(i) \leq n - 1$ pentru $\forall i \in X$. Pentru că numerele $g(i)$ sunt distințe și sunt n valori din domeniul $0..n-1$, rezultă că $\exists p, q \in X$ astfel încât $g(p) = 0$ și $g(q) = n - 1$. Prin urmare vârful p nu este adjacent cu nici un vârf al grafului (inclusiv nici cu q) și vârful q este adjacent cu toate celelalte vârfuri ale grafului (inclusiv cu p). Această afirmație contradictorie ne conduce la concluzia că nu există grafe cu gradele vâfurilor distincte sau în orice graf există cel puțin două vârfuri care au același grad sau funcția ce asociază fiecărui vârf gradul său într-un graf nu este injectivă.

Alte rezolvări ale problemei de mai sus se găsesc în [TCI94].

Definiția 1.1.9. Fiind dat un graf orientat $G = (X, U)$, prin *drum* în graful G înțelegem o succesiune de arce cu proprietatea că extremitatea terminală a unui arc al drumului coincide cu extremitatea inițială a arcului următor din drum.

Prin drum convenim să înțelegem și succesiunea de vârfuri care sunt extremități ale arcelor ce compun drumul. Deci, un drum μ este fie $\{u_1, u_2, \dots, u_q\}$ fie $\{i_0, i_1, \dots, i_q\}$ cu proprietatea că $u_j = (i_{j-1}, i_j) \in U$ pentru $j = 1, 2, \dots, q$. Spunem că drumul μ are ca extremitate inițială pe i_0 și ca extremitate finală pe i_q sau drumul μ este de la i_0 la i_q .

Definiția 1.1.10. Lungimea unui drum este numărul de arce care compun drumul respectiv.

Definiția 1.1.11. Un drum într-un graf este:

- *simplu* dacă nu folosește de două ori un același arc;
- *compus* dacă nu este simplu;
- *elementar* dacă nu conține (trece) de două ori un același vârf;
- *circuit* dacă extremitatea inițială a drumului coincide cu cea finală;
- *eulerian* dacă este simplu și trece prin toate arcele grafului;
- *hamiltonian* dacă este elementar și trece prin toate vâfurile grafului.

Corespunzător noțiunilor de drum și circuit în grafele neorientate sunt noțiunile de *lanț* respectiv *ciclu*. Deci, *lanț* este o succesiune de muchii cu proprietatea că oricare muchie are o extremitate comună cu muchia precedentă și cealaltă extremitate este comună cu muchia următoare. Dacă extremitățile lanțului coincid, atunci lanțul se numește *ciclu*.

Între două drumuri (lanțuri) μ_1 și μ_2 cu proprietatea că extremitatea terminală a lui μ_1 coincide cu extremitatea inițială a lui μ_2 se definește operația de *componere* a lor prin care se obține drumul $\mu_1 \circ \mu_2$ format prin concatenarea celor două succesiuni de arce (muchii).

Definiția 1.1.12. Un graf orientat este *tare conex* dacă între oricare două vârfuri ale grafului există un drum.

Deci, într-un graf tare conex prin oricare două vârfuri trece cel puțin un circuit.

Definiția 1.1.13. Un graf neorientat este *conex* dacă între oricare două vârfuri ale grafului există un lanț.

Convenție:

Dacă relativ la un graf orientat ne referim la o noțiune definită pentru grafele neorientate atunci se consideră că pentru precizarea ei se neglijăază orientarea arcelor. Dacă relativ la un graf neorientat ne referim la o noțiune definită pentru un graf orientat atunci se consideră că fiecare muchie este înlocuită cu cele două arce, de sensuri opuse și care au aceleași extremități ca și muchia înlocuită. Astfel, graful din fig.1.1.1. nu este tare conex, dar este conex. Acesta pentru că de la vârful 3 la vârful 1 nu există drum (deci nici circuit care să treacă prin ele), dar succesiunea de vârfuri 4,3,2,1,4 este un ciclu hamiltonian și deci între oricare două vârfuri există lanț.

Graful din figura 1.1.1. are, de altfel, o infinitate de cicluri (de exemplu 343 sau 1234321) toate de lungime pară, chiar dacă numai unul dintre acestea este ciclu simplu (și hamiltonian).

De aceea programele care rezolvă probleme ce se pun pentru un anumit tip de graf (orientat sau neorientat) după procedura de citire a datelor este bine să conțină o procedură de transformare a grafului citit în graful corespunzător pe (in) care se va rezolva problema pusă.

Dacă $G = (X, U)$ este un graf neorientat, atunci în forma $\bar{G} = (X, \Gamma)$ el este dat de aplicația multivocă $\Gamma : X \rightarrow X$ cu $\Gamma i = \Gamma^i = \{(i, j) \mid (i, j) \in U\}$, $\forall i \in X$. Deci, pentru grafele neorientate $g^+(i) = g^-(i) = g(i)$ și $g(1) + g(2) + \dots + g(n) = 2m$.

Definiția 1.1.14. *Graful linie* asociat grafului $G = (X, U)$ este graful $L(G) = (U, V)$ cu $V = \{(u, v) \mid u, v \in X$ și u, v sunt adiacente}.

Deci, în graful linie $L(G)$ vârfuri sunt muchiile lui G și între două asemenea vârfuri se definește muchie în $L(G)$ dacă și numai dacă în G ele sunt muchii adiacente. Să observăm că $L(L(G))$ este izomorf cu G , dar că și grafe neetichetate $L(L(G)) = G$.

Problema podurilor din Königsberg cere în termenii teoriei grafelor să se determine, dacă există, un lanț eulerian în graful atașat ei (figura 2).

Un rezultat important pentru problemele de acest tip este:

Teorema 1.1.2. *Un graf conex (tare conex) admite un ciclu (respectiv un circuit) eulerian dacă și numai dacă $g(i)$ este par (respectiv $g^+(i) = g^-(i)$), $\forall i \in X$.*

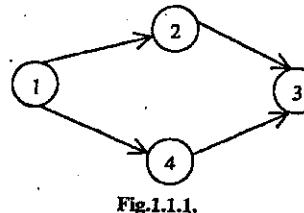


Fig.1.1.1.

Demonstrație:

Demonstrația acestei echivalențe o facem în paragraful de conexitate 1.4.

Corolar 1.1.1. *Un graf conex admite un lanț eulerian (care nu este ciclu) dacă și numai dacă toate vârfurile cu excepția a două vârfuri au gradele pare. Vârfurile cu gradele impare sunt extremitățile lanțului.*

Corolar 1.1.2. *Un graf orientat admite un drum eulerian (care nu este circuit) dacă și numai dacă există vârfurile i_0 și j_0 astfel încât*

$$g^+(i) = g^-(i), \forall i \in X - \{i_0, j_0\},$$

$$g^+(i_0) = g^-(i_0) + 1 \text{ și } g^+(j_0) = g^-(j_0) - 1.$$

Vârful i_0 fiind extremitatea inițială, iar j_0 este extremitatea finală a drumului eulerian.

Determinarea unui drum, circuit, lanț sau ciclu eulerian se poate face prin a parcurge vârfurile grafului, pornind de la extremitatea inițială respectiv de la un vârf oarecare, folosind din fiecare vârf un arc respectiv o muchie care nu a mai fost utilizat(ă).

Problema 1.1.2. Se dă un sistem de străzi care trebuie asfaltate. Din motiv de protecție a asfaltului, utilajele cu care se asfaltează nu trebuie să treacă pe o șosea asfaltată. Se cere să se determine ordinea de asfaltare a străzilor.

Rezolvare:

A rezolva această problemă înseamnă să determinăm un lanț eulerian în graful ce reprezintă sistemul de străzi ce trebuie asfaltate. Dacă un asemenea lanț nu există atunci se determină un drum eulerian în graf orientat construit conform convenției de mai sus. Asfaltarea unei străzi se va face la ultima trecere a utilajelor pe acea stradă.

Definiția 1.1.15. *Graful orientat (neorientat) care are cel puțin un ciclu eulerian sau hamiltonian se numește **graf eulerian** respectiv **graf hamiltonian**.*

Acstea noțiuni sunt duale în sensul următor:

graful G este eulerian dacă și numai dacă graful linie $L(G)$ corespunzător este hamiltonian.

În ceea ce privesc grafele hamiltoniene dăm următoarele teoreme (probleme), care sunt rezolvate și în [TCI94].

Teorema 1.1.3. În K_n , graful complet de ordin $n \geq 2$ există:

- a. $(n - 1)!/2$ cicluri hamiltoniene distincte;
- b. $2^{h-1}(n - h - 1)!$ cicluri hamiltoniene distincte care trec prin h muchii neadiacente fixate (date).

Demonstrație:

a. Graful fiind complet între oricare două vârfuri există muchie și deci orice succesiune de vârfuri este un lanț. Prin urmare ciclul hamiltonian este orice succesiune în care apar toate vârfurile grafului o singură dată la care se adaugă primul vârf din acea succesiune. Adică, orice permutare circulară (și numai acestea) este un ciclu hamiltonian. Prin urmare, vom rezolva problema prin a determina (a număra) câte permutări circulare se pot forma dintr-o mulțime cu n elemente. Știm că numărul permutărilor (liniare) a unei mulțimi cu n elemente este $n!$. Cum dintr-un permutare circulară (cerc pe care sunt plasate n puncte) se pot obține $2n$ permutări liniare (prin "ruperea" cercului pe arcul de după: punctul 1 sau punctul 2 sau ... sau punctul n și apoi "întinderea" sa astfel ca un capăt să fie în stânga sau acel capăt să fie în dreapta) deducem că numărul permutărilor circulare este de $2n$ ori mai mic decât al celor liniare. Prin urmare numărul ciclurilor hamiltoniene distincte din K_n (permutele circulare ale unei mulțimi cu n elemente) este $n!/(2n) = (n - 1)!/2$.

b. Să notăm cele h muchii date prin (i_{2j-1}, i_{2j}) cu $j = 1, h$. Deoarece, în ciclurile hamiltoniene (permutele circulare) care au proprietatea dată vârfurilor i_{2j+1} sunt fie precedate fie urmate de vârfurile i_{2j} , vom considera graful K_{n-h} ale cărui vârfuri sunt de două categorii. Astfel, h vârfuri corespund celor h muchii ale grafului K_n , date în problemă, și pe care le vom numi pseudovârfuri, iar celelalte $n - 2h$ sunt vârfurile din K_n neadiacente nici una dintre cele h muchii date, pe care le putem numi vârfuri veritabile. Fiecare ciclu hamiltonian ω al grafului K_{n-h} generează prin "spargerea" celor h pseudovârfuri, 2^h cicluri hamiltoniene din K_n care trec prin cele h muchii date și invers, adică din 2^h cicluri hamiltoniene din K_n care trec prin cele h muchii date se obține un același ciclu hamiltonian din K_{n-h} , prin înlocuirea celor h muchii cu căte un pseudovârf. Aceasta rezultă din faptul că o "spargere" de pseudovârfuri înseamnă: se aleg cele h muchii ale lui K_n ca fiind un element dintre cele 2^h ale mulțimii $\{(i_1, i_2); (i_2, i_1)\} \times \{(i_3, i_4); (i_4, i_3)\} \times \dots \times \{(i_{2h-1}, i_{2h}); (i_{2h}, i_{2h-1})\}$, de perechi ordonate de vârfuri ale lui K_n , apoi se înlocuiește fiecare pseudovârf al lui ω cu perechea (muchia) corespunzătoare. Pentru că K_{n-h} are $(n - h - 1)!/2$

cicluri hamiltoniene (conform cazului a), deducem că în K_n trec prin cele h muchii date $2^h(n - h - 1)!/2 = 2^{h-1}(n - h - 1)!$ cicluri hamiltoniene. \square

Teorema 1.1.4. Dacă $G = (X, U)$ este un graf de ordin $n \geq 3$ și pentru orice pereche de vârfuri p și q neadiacente are loc $g(p) + g(q) \geq n$, atunci G este hamiltonian. [Ore60]

Demonstrație:

Fără a restrânge generalitatea putem considera că G este un graf saturat cu proprietatea din enunțul teoremei, adică orice muchie s-ar adăuga la U în G să ar forma un ciclu hamiltonian. Altfel spus dacă demonstrăm că graful saturat (cu număr maxim de muchii) ce nu conține ciclu hamiltonian are o pereche de vârfuri neadiacente i, j cu $g(i) + g(j) < n$ atunci orice alt graf fără ciclu hamiltonian are o asemenea pereche de vârfuri.

Fie (i, j) o pereche de vârfuri între care nu există muchie în G . Atunci prin adăugarea muchiei dintre aceste vârfuri se va obține cel puțin un ciclu hamiltonian. Deci în graful G există între i și j un lanț hamiltonian. Notăm cu $\omega = (i = x_1, x_2, \dots, x_h = j)$ acest lanț și cu k pe $g(i)$. Deci vârful i are k vârfuri adiacente, considerăm că aceste vârfuri au în lanțul ω indicii $2 = i_1 < i_2 < \dots < i_k \leq n$. Să demonstrăm că vârful j nu este adiacent cu nici unul dintre vârfurile indicii $i_s - 1$, unde $1 \leq s \leq k$, din sirul x . Dacă j ar fi adiacent cu vârful cu indicele $i_s - 1$ atunci graful G conține ciclul hamiltonian următor ($i = x_1, \dots, x_{i_s-1}, j = x_n, x_{n-1}, \dots, x_{i_s}, x_1 = i$).

Prin urmare, gradul vârfului j verifică $g(j) \leq n - 1 - k = n - 1 - g(i)$ ceea ce este o contradicție cu ipoteza din care se deduce că $g(i) + g(j) \geq n$. \square

Urmărind ideea demonstrației teoremei de mai sus să se rezolve problema următoare.

Problema 1.1.2. Fie $G = (X, U)$ un graf cu $n \geq 3$ vârfuri, iar p și q două vârfuri (date) neadiacente pentru care $g(p) + g(q) \geq n$. Să se demonstreze că $G' = (X, U')$, cu $U' = U \cup \{(p, q)\}$ este hamiltonian dacă și numai dacă G este hamiltonian. [BLW76]

Observație:

Operația de obținere a grafului G' se mai notează cu $G' = G + pq$, unde pq este notație pentru muchia (p, q) .

Rezolvare:

Dacă G este hamiltonian prin adăugarea de muchii și noul graf este hamiltonian, pentru că ciclul hamiltonian din G este ciclu hamiltonian și în noul graf.

Fie p și q două vârfuri pentru care $g(p) + g(q) \geq n$ și $G + pq$ este hamiltonian. Deci, există un ciclu hamiltonian în $G + pq$. Dacă acest ciclu nu conține muchia pq atunci el este ciclu hamiltonian și în G , dacă el conține muchia pq printre-un raționament ca la teorema 1.1.4 se ajunge la contradicția $g(p) + g(q) < n$. Deci, acest caz nu este posibil și prin urmare ciclul hamiltonian din $G + pq$ nu conține muchia pq , adică este ciclu hamiltonian și în G .

Precizăm că dacă în teorema 1.1.4, inegalitatea cu suma gradelor trebuie să aibă loc pentru toate perechile de vârfuri neadiacente, în problema 1.1.2, această inegalitate este cerută pentru o singură asemenea pereche de vârfuri. Dar, se cere o condiție suplimentară $G + pq$ să fie hamiltonian. Alte condiții de suficiență pentru ca un graf să fie hamiltonian le-am grupat în teorema următoare.

Teorema 1.1.5. Fie $G = (X, U)$ un graf de ordin $n \geq 3$ pentru care gradele vârfurilor verifică inegalitățile $d_1 \leq d_2 \leq \dots \leq d_n$. Graful G conține un ciclu hamiltonian dacă este verificată cel puțin una dintre condițiile:

- $d_1 \geq n/2$ (Dirac) [Dir52];
- $\exists k \in X: d_k \leq k \leq n/2 \Rightarrow d_{n-k} \geq n - k$ (Chvátal);
- $\forall p, q : d_p \leq p \text{ și } d_q \leq q \Rightarrow d_p + d_q \geq n$ (Bondy).

Demonstrație:

a. Din $d_1 \geq n/2$ și faptul că șirul d este ordonat crescător, deducem că pentru orice pereche de vârfuri p și q (nu numai pentru cele neadiacente) are loc $d_p + d_q \geq n$ și conform teoremei 1.1.4 graful G este hamiltonian.

b. Vom demonstra afirmația prin metoda reducerii la absurd. Fie G un graf cu n vârfuri, saturat în multimea grafelor care nu conțin ciclu hamiltonian. Alegem vârfurile i, j neadiacente cu proprietatea $i < j$ și $i + j$ este maxim. Din maximitatea lui $i + j$ deducem că:

- vârful i este adiacent vârfurilor: $j+1, \dots, n$ și
- vârful j este adiacent vârfurilor: $i+1, \dots, j-1, j+1, \dots, n$.

De aici

$$1. \quad d_i \geq n - j;$$

$$2. \quad d_j \geq n - i - 1;$$

G fiind graf saturat rezultă că $G + ij$ conține un ciclu hamiltonian și urmărind demonstrația teoremei 1.1.4, se deduce:

$$3. \quad d_i \leq n - 1 - d_j, \text{ adică } d_i + d_j \leq n - 1.$$

$$\text{Din (2) și (3) obținem } d_i \leq n - 1 - d_j \leq n - 1 - (n - i - 1) = i.$$

Notăm cu $k = d_i \leq i$ și conform ipotezei $d_k \leq d_i = k$.

Pe de altă parte $i < j$ implică $d_i \leq d_j$ care împreună cu (3) conduce la $2k = 2d_i \leq n - 1 < n$.

Deci $d_k \leq k < n/2$ și conform ipotezei

$$4. \quad d_{n-k} \geq n - k = n - d_i \geq d_j + 1$$

Dar $n - k > j$, pentru că altfel din $n - k \leq j$ obținem $d_{n-k} \leq d_j$, ceea ce e în contradicție cu (4).

Deci $n - d_i = n - k > j$ sau $d_i < n - j$, ceea ce este în contradicție cu (1) și demonstrează că presupunerea făcută este falsă.

c. Demonstrăm că în acest caz este verificată condiția b.

Fie k un vârf cu $d_k \leq k < n/2$. Notăm $l = n - k$.

Presupunem că $d_l < l$. Din ipoteză $d_k + d_l \geq n$ și deci $d_l \geq n - d_k \geq n - k = l$, ceea ce implică $d_l \geq l$. Prin urmare este verificată condiția b, din care deducem că graful G în care are loc condiția c conține un ciclu hamiltonian. \square

Pentru determinarea de drumuri, lanțuri, circuite sau cicluri hamiltoniene se poate utiliza metoda backtracking (pentru cicluri hamiltoniene metoda este descrisă în paragraful 1.3.1) și reprezentarea grafului cu lista succesorilor (vezi paragraful 1.2.3.).

Pentru a demonstra însă că un graf $G = (X, U)$, de exemplu cel din figura 1.1.2., nu este hamiltonian se poate folosi următoarea metodă.

Mai întâi să constatăm că orice ciclu hamiltonian este ciclu elementar și are un număr de muchii egal cu numărul de vârfuri ale grafului, în cazul nostru 16. Pe de altă parte, dintre muchiile oricărui ciclu elementar (deci și ale ciclurilor hamiltoniene) exact câte două sunt incidente fiecărui vârf. Prin urmare dintre muchiile incidente oricărui vârf un ciclu elementar utilizează două, celelalte nu sunt folosite. Vom determina câte muchii ale grafului nu pot fi utilizate de orice ciclu elementar și prin diferență calculăm numărul maxim de muchii dintr-un ciclu elementar al lui G . Dacă acest număr este mai mic decât numărul de vârfuri atunci graful nu conține nici un ciclu hamiltonian.

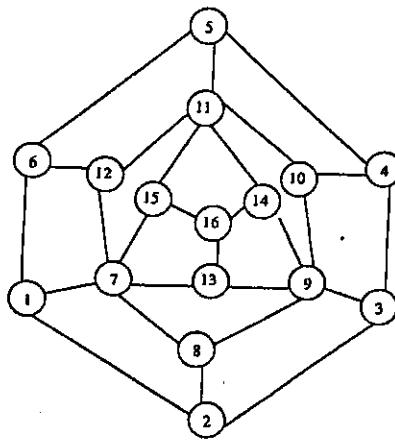


Fig. 1.1.2.

Pentru a aplica metoda descrisă mai sus considerăm pe rând submulțimile de vârfuri $S_1 = \{2, 4, 6\}$, $S_2 = \{7, 9, 11\}$ și $S_3 = \{16\}$ ale grafului G . Mai întâi să observăm că oricare două vârfuri ale oricărora dintre aceste mulțimi nu sunt adiacente și că $\Gamma S_1 \cap (S_2 \cup S_3) = \emptyset$, $\Gamma S_2 \cap (S_1 \cup S_3) = \emptyset$ și $\Gamma S_3 \cap (S_1 \cup S_2) = \emptyset$. Pentru că numărul muchiilor grafului incidente vârfurilor din aceste trei mulțimi este 9, 15 respectiv 3, iar orice ciclu are exact două muchii incidente fiecărui vârf, deducem că dintre aceste muchii orice ciclu nu va folosi 3, 9 respectiv una, un total de 13 muchii. Deci dintre cele 27 de muchii ale grafului orice ciclu nu va utiliza 13 muchii. Așadar, numărul muchiilor oricărui ciclu elementar este cel mult $27 - 13 = 14$ și un asemenea ciclu elementar neavând 16 muchii nu este ciclu hamiltonian. Pentru demonstrație era suficientă considerarea doar a mulțimilor S_1 și S_2 , dar pentru o înțelegere mai clară a demonstrației am utilizat și mulțimea S_3 .

Precizăm că cele trei submulțimi de vârfuri considerate sunt mulțimi interior stabilă (vezi definiția 1.3.1.1.) și că cele trei mulțimi de muchii incidente acestora sunt disjuncte două câte două.

Pentru grafele bipartite se poate utiliza și următoarea teoremă.

Teorema 1.1.6. *Un graf bipartit $G = (X, U)$, cu mulțimile de vârfuri A_1 și $A_2 = X - A_1$, de cardinală p respectiv q , poate conține un ciclu hamiltonian numai dacă $p = q$; și poate conține un lanț hamiltonian numai dacă $|p - q| \leq 1$.*

Demonstrație:

Să presupunem că G conține ciclul hamiltonian:

$x_1, x_2, \dots, x_n, x_1$.

Cum G este graf bipartit cu mulțimile de vârfuri A_1 și $A_2 = X - A_1$, se deduce că:

$A_1 = \{x_1, x_3, \dots, x_{n-1}\}$ și $A_2 = \{x_2, x_4, \dots, x_n\}$.

De aici deducem că $p = q = n/2$. \square

Dacă G conține lanțul hamiltonian:

x_1, x_2, \dots, x_n .

Cum G este graf bipartit, cu mulțimile de vârfuri A_1 și $A_2 = X - A_1$, se deduce că:

$A_1 = \{x_1, x_3, \dots\}$ și $A_2 = \{x_2, x_4, \dots\}$.

De aici, pentru că n poate fi par sau impar, deducem că $|p - q| \leq 1$. \square

Rezultate cu privire la circuite și cicluri euleriene precum caracterizarea grafelor euleriene se găsesc în paragraful 1.4.

Definiția 1.1.16. *Arbore* este un graf conex și fără cicluri.

Definiția 1.1.17. Un graf se numește *aciclic* dacă nu conține cicluri.

Graful aciclic se mai numește și *pădure*. Așa cum se deduce ușor, o pădure este un graf care este format din cel puțin un arbore și numărul arborilor dintr-o pădure este egal cu numărul componentelor conexe ale acelui graf.

Aici am prezentat doar noțiunile de arbore și de pădure. O abordare a acestora o facem în paragraful 1.4.3.

Definiția 1.1.18. *Arborescență* este orice graf $G = (X, \Gamma)$ cu proprietatea $\exists r \in X$ astfel încât $\Gamma^{-1}r = \Phi$ și $\forall i \in X - \{r\}$ există un singur drum de la r la i .

Vârful r este un vârf special și se numește *rădăcina arborescenței*. Alte vârfuri speciale ale unei arborescențe sunt vârfurile care nu au succesiști sau $\Gamma i = \Phi$ cu $i \in X$, pe care le numim *vârfuri frunză* sau *vârfuri terminale*. Să observăm că într-o arborescență fiecare vârf cu excepția rădăcinii are un singur

precedesor numit și vîrf părinte al respectivului vîrf. Adică, $\forall i \in X - \{r\}$ $|\Gamma^{-1}i| = 1$ și $j \in \Gamma^{-1}i$ este părintele lui $i \in X$. Precizăm de asemenea că, dintr-un arbore (X, U) cu n vârfuri se pot obține n arborescențe conform cu alegerea rădăcinii r din X și orientând muchiile astfel încât toate lanțurile dintre r și orice alt vîrf i să devină drumuri de la r la i . În unele lucrări în loc de arborescență se utilizează termenul de arbore de rădăcină r , noi am preferat termenul de arborescență pentru a accentua faptul că este un graf orientat spre deosebire de arbore care este un graf neorientat. De asemenea, chiar și reprezentarea geometrică a arborescențelor se face în forma grafului neorientat prin plasarea vâfurilor pe nivele. La nivelul 0 se află rădăcina și muchiile au orientarea de la nivelul superior către nivelul inferior (de la rădăcină spre frunze).

Notății:

Fiind dat un graf $G = (X, U)$, o mulțime de vârfuri $A \subset X$ și o mulțime de arce (muchii) $V \subset U$ convenim următoarele notații:

- $G(A)$ pentru subgraful corespunzător mulțimii A , adică $(A, (A \times A) \cap U)$ numit *subgraful generat* de A .
- $G(V)$ pentru subgraful parțial corespunzător mulțimii de arce (muchii) V . Deci $G(V)$ are ca mulțime de vârfuri mulțimea vâfurilor adiacente arcelor (muchiilor) din V , iar ca mulțime de arce (muchii) pe V .
- $X(G)$ pentru mulțimea vâfurilor grafului G .
- $U(G)$ pentru mulțimea arcelor (muchiilor) grafului G .
- $G - A$ pentru sugraful $G(X - A)$.
- $G - V$ pentru subgraful parțial $G(U - V)$.
- dacă $V \subset X \times X$ prin $G + V$ înțelegem graful $(X, U \cup V)$, adică perechile de vârfuri din V se transformă în muchii sau imprecis spus muchiile din V se adaugă grafului G .

Pentru un graf $G = (X, U)$ orientat și un vîrf i sau o mulțime de vârfuri A din X notăm cu:

- $\omega^+(i)$ mulțimea tuturor arcelor incidente spre exterior vârfului i , adică mulțimea arcelor ce "ies" din i sau

$$\omega^+(i) = \{(i, j) \in U \mid j \in X\} = \{(i, j) \mid j \in \Gamma^+i\} = \{i\} \times \Gamma^+i;$$
- $\omega^-(i)$ mulțimea arcelor incidente spre interior vârfului i , adică mulțimea arcelor ce "intră" în i sau

$$\omega^-(i) = \{(j, i) \in U \mid j \in X\} = \{(j, i) \mid j \in \Gamma^-i\} = \Gamma^-i \times \{i\};$$

- $\omega^+(A)$ mulțimea arcelor incidente spre exterior mulțimii A , adică $\omega^+(A) = \{(i, j) \in U \mid i \in A \text{ și } j \in X - A\}$ (mulțimea arcelor ce "ies" din A și "intră" în $X - A$);
- $\omega^-(A)$ mulțimea arcelor incidente spre interior mulțimii A , adică $\omega^-(A) = \{(i, j) \in U \mid i \in X - A \text{ și } j \in A\}$ (mulțimea arcelor ce "intră" în A și "ies" din $X - A$);
- $\omega(i) = \omega^+(i) \cup \omega^-(i)$ și $\omega(A) = \omega^+(A) \cup \omega^-(A)$.

Să observăm că $\omega^+(A) = \omega^-(X - A)$.

Pentru grafele neorientate, deoarece conform convenției făcute au loc $\omega^+(i) = \omega^-(i)$ și $\omega^+(A) = \omega^-(A)$, se folosesc doar notațiile $\omega(i)$ respectiv $\omega(A)$.

Definiția 1.1.18. Numim *cocircuit* orice mulțime V de arce cu proprietatea că există o mulțime de vârfuri A astfel încât $V = \omega^+(A)$.

Mulțimile $\omega(A)$, pentru $A \subset X$, se numește *cociclu*.

O noțiune de graf generală care să cuprindă ca și clase particulare atât mutigrafele orientate cât și cele neorientate și în plus alte combinații o prezentăm în cele ce urmează.

Definiția 1.1.19. *Graf în sens larg* sau simplu *graf* este orice sistem

$$G = (X, U; I, O),$$

unde X este o mulțime de elemente oarecare numite vârfuri ale grafului, U este o altă mulțime oarecare, care nu are elemente din X , elementele lui U le numim *arce* (muchii sau legături); I și O sunt submulțimi ale lui $U \times X$, deci sunt relații binare neomogene, cu proprietățile $|I(u)| \in \{1, 2\}$ și $|O(u)| \in \{0, 1\}$, pentru $\forall u \in U$ și $O \subset I$.

Relația I se numește *relația de incidentă* și $(u, i) \in I$ se poate interpreta: arcul u este incident vârfului i sau vârful i este extremitate a arcului u . Deci, un arc poate fi incident la unul sau la două vârfuri. Relația O se numește *relația de orientare* și $(u, i) \in O$ se poate interpreta arcul u are ca extremitate terminală pe i . Deci, într-un asemenea graf se pot întâlni muchii și arce simple sau replicate (cu aceleși extremități și eventual aceeași orientare) și eventual bucle.

Pentru asemenea grafe se pot redefini noțiunile de drum (succesiuni de arce de același sens), lanț (succesiuni de muchii) și introduce noțiunea de sublanț (generalizând noțiunile de lanț și de drum, ca fiind o succesiune de vârfuri adiacente).

În continuare vom lucra cu noțiunile de graf, mutigraf orientate sau neorientate date de definițiile clasice.

1.2. Reprezentări ale grafelor

Pentru a se putea lucra mai ușor cu grafe se recurge la o reprezentare a acestora. Aceste reprezentări sunt necesare în cazul în care vrem să precizăm (dăm) graful pe hârtie (tablă) sau pentru un calculator. Așa cum se întâmplă în general la reprezentarea datelor și aici se poate vorbi despre modelul logic și cel fizic al reprezentării grafelor. Noi vom prezenta doar modelele logice. În funcție de problema care dorim să o rezolvăm și de modul de rezolvare al acesteia se va utiliza una sau alta dintre reprezentările grafelor. Bineînțeles că alegerea făcută influențează timpul de răspuns al algoritmilor sau exprimat mai general influențează complexitatea acestora. Se cunosc mai multe moduri de reprezentare ale grafelor.

1.2.1. Reprezentarea geometrică

Această reprezentare se face prin a pune în corespondență elementelor grafului elemente geometrice din plan. Corespondența este:

- vârfurile grafului se reprezintă prin puncte sau cerculete sau pătrătele, dacă este nevoie marcate (etichetate);
- fiecare arc se reprezintă printr-o linie dreaptă sau curbă (*curbă Jordan*) ce unește reprezentările extremităților arcului respectiv, orientată de la reprezentarea extremității inițiale către reprezentarea extremității terminale a arcului;
- muchiile se reprezintă ca și arcele numai că lipsește orientarea.

Uneori arcele (muchiile) sunt marcate fie cu numărul arcului respectiv fie cu alte valori care pot avea semnificații conforme problemei.

Graful dat la exemplul 1.1.1 se poate reprezenta geometric ca în figura 1.2.1.1.

Dacă reprezentările vârfurilor sunt marcate (etichetate) graful se numește graf etichetat. Aici trebuie să remarcăm că un același graf etichetat poate avea reprezentări diferențite ca în figura 1.2.1.2.

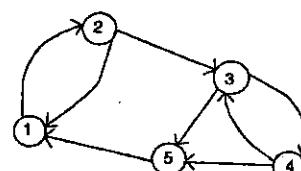


Fig. 1.2.1.1.

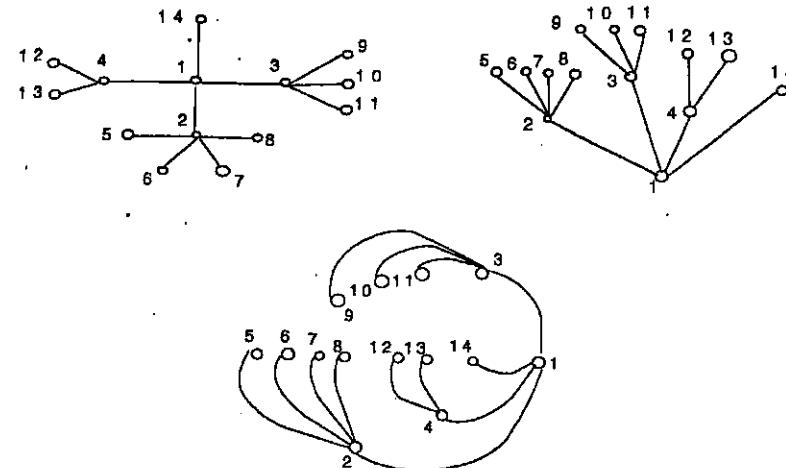


Fig. 1.2.1.2.

Dacă reprezentările vârfurilor nu sunt etichetate graful se numește *graf neeticetat*. Să remarcăm că două grafe etichetate pot fi diferențiate dar izomorfe, în timp ce grafele corespunzătoare neeticetate sunt egale. Prin urmare un graf neeticetat este o clasă de echivalență din mulțimea căt a mulțimii grafelor în raport cu relația de echivalență numită izomorfism de grafe. Grafele reprezentate în figura 1.2.1.3 sunt grafe izomorfe, deoarece există o bijectie între mulțimile de vârfuri ale celor două grafe care să verifice proprietatea din definiția grafelor izomorfe. O astfel de bijectie este $f(1) = 1, f(2) = 3, f(3) = 5, f(4) = 4, f(5) = 2, f(6) = 6$. Graful neeticetat din figura 1.2.1.3, obținut prin neglijarea etichetelor vârfurilor, este graful bipartit complet $K_{3,3}$.

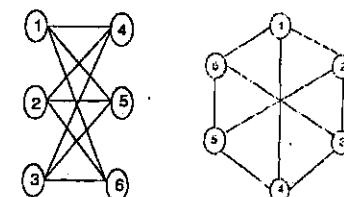


Fig. 1.2.1.3.

Definiția 1.2.1.1. *Graf planar* este un graf care admite o reprezentare geometrică în care reprezentările arcelor (muchiilor) nu se intersecțează.

Orice componentă conexă, în sens topologic, a planului din care s-au eliminat reprezentările vârfurilor și ale arcelor (muchiilor) unui graf planar se numește *față* a grafului planar respectiv. Adică față a unui graf planar este o

regiune a planului, în care s-a reprezentat un graf planar, limitată de muchii ale grafului cu proprietatea că oricare două puncte ale sale se pot uni printr-o linie poligonă care nu intersectează muchii ale grafului. Un rezultat important este acela prin care se arată că numărul fețelor unui graf planar este constant indiferent de reprezentarea planară a grafului.

Deoarece un același graf admite mai multe reprezentări geometrice nu este ușor de a stabili dacă un graf oarecare este sau nu planar. Dintre lucrările ce se ocupă de această problemă amintim aici [Kur30], [ToS95] și [Vos91].

Observația 1.2.1.1.

În spațiu orice graf se poate reprezenta astfel încât oricare două muchii (arce) să nu se intersecteze. Această reprezentare se poate face astfel:

- Se reprezintă toate vârfurile pe o dreaptă.
- Pentru reprezentarea fiecărei muchii (arc) se alege căte un plan distinct ce conține dreapta pe care sunt reprezentate vârfurile grafului, apoi se reprezintă în acel plan muchia (arcul). Cum planele în care s-au reprezentat căte o muchie (un arc) sunt disjuncte și au în comun doar dreapta pe care s-au reprezentat vârfurile grafului înseamnă că oricare două muchii (arce) nu se intersectează.

Reprezentării geometrice în calculator îi corespunde reprezentarea cu liste înlanțuite sau pointeri sau o altă structură de date convenabilă.

1.2.2. Reprezentarea matriceală

Un graf $G = (X, U)$ se poate reprezenta cu ajutorul matricei de adiacență (vârfuri-vârfuri) sau al matricei de incidentă (vârfuri-arce). Fiecare linie și coloană corespunzând unui vârf sau unui arc (muchie) al grafului.

MATRICEA DE ADIACENȚĂ

Matricea de adiacență este o matrice pătrată, linia și coloana i corespunzând vârfului i al grafului. Elementele matricei de adiacență se definesc astfel:

$$a_{ij} = \begin{cases} 1, & (i, j) \in U \\ 0, & (i, j) \notin U \end{cases}, \quad i, j \in X.$$

Astfel graful dat la exemplul 1.1.1, reprezentat geometric în figura 1.2.1.1, se poate preciza și prin matricea de adiacență următoare:

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

Observația 1.2.2.1.

- Valorile matricei de adiacență se pot considera ca valori booleene și deci elementul a_{ij} precizează dacă perechea de vârfuri (i, j) este sau nu arc (muchie) în graful reprezentat prin matricea de adiacență A .
- Dacă valorile matricei de adiacență se interpretează ca numere naturale, atunci numărul a_{ij} este numărul arcelor (muchiilor) de la i la j în graf G . Pentru multigrafe se poate conveni astfel ca valorile matricei A să fie numere naturale conforme acestei interpretări.
- Pentru grafele neorientate matricile de adiacență sunt simetrice și putem interpreta și invers, dacă matricea de adiacență a unui graf este simetrică, atunci acel graf este neorientat.

MATRICEA DE INCIDENTĂ

Matricea de incidentă se atașează grafelor simple (fără bucle) și a căror mulțime de arce s-a ordonat, adică s-au numerotat arcele, $U = \{1, 2, \dots, m\}$. Linia i corespunde vârfului i , iar coloana u corespunde arcului u . Deci matricea este de tip $n \times m$. Elementele matricei de incidentă se definesc astfel:

$$b_{iu} = \begin{cases} 1, & \exists j \in X : u = (i, j) \\ -1, & \exists j \in X : u = (j, i), \quad i \in X, u \in U \\ 0, & altfel \end{cases}$$

Graful de la exemplul 1.1.2, reprezentat geometric în figura 1.2.2.1, se poate reprezenta prin matricea de incidentă următoare:

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

Teorema 1.2.2.1. Dacă A este matricea de adiacență a unui graf de ordin n atunci matricea A^p indică prin elementul de pe linia i și coloana j numărul de drumuri de lungime p de la vârful i la vârful j .

Demonstrație:

Folosim metoda inducției complete în raport cu p . Pentru $p = 1$ proprietatea este adeverată deoarece dacă $a_{ij} = 1$, atunci de la vârful i la vârful j există arc. Cum graful este un 1-graf înseamnă că de la vârful i la vârful j există a_{ij} drumuri de lungime 1. Dacă $a_{ij} = 0$, atunci deoarece între vârfurile i și j nu există arc înseamnă că numărul de drumuri de lungime 1 este a_{ij} . Presupunând proprietatea adeverată pentru p să demonstrăm că este adeverată și pentru $p + 1$. Dacă $a_{ij} = 1$, atunci numărul drumurilor de lungime $p + 1$ de la i la j ce au ca ultim arc pe (k, j) este egal cu numărul drumurilor de lungime p de la i la k adică cu $a_{ik}^p a_{kj}$. Cum un drum de la i la j poate avea ca penultim vârf pe orice vârf k , înseamnă că numărul drumurilor de lungime $p + 1$ de la i la j este

$$\sum_{k=1}^n a_{ik}^p a_{kj} \text{ sau } a_{ij}^{p+1}.$$

Dacă A este matricea de adiacență a unui graf neorientat, atunci A^p indică prin elementul de pe linia i și coloana j numărul de lanțuri de lungime p între vârfurile i și j . \square

Definiția 1.2.2. Într-un graf G conex numim **distanță** dintre vârfurile i și j minimul lungimilor lanțurilor dintre i și j .

Notăm distanța dintre i și j cu $d(i, j)$. Convenind $d(i, i) = 0$. Noțiunea introdusă verifică axiomele distanței:

- **pozitivitatea**, adică $d(i, j) \geq 0$, $\forall i, j \in X$, egalitatea are loc dacă și numai dacă $i = j$;
- **simetria**, adică $d(i, j) = d(j, i)$, $\forall i, j \in X$, relația are loc pentru că orice lanț de la i la j induce un lanț (aceleași muchii dar în ordine inversă) de la j la i ;
- **inegalitatea triunghiului**, adică $d(i, j) \geq d(i, k) + d(k, j)$, $\forall i, j, k \in X$, care are loc din cauză că $d(i, k) + d(k, j)$ reprezintă numărul de muchii ale unui lanț de la i la j , mai precis ale lanțului obținut prin compunerea lanțului de lungime minimă de la i la k cu lanțul de

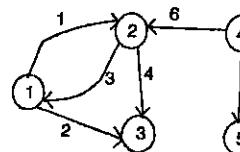


Fig. 1.2.2.1.

lungime minimă de la k la j , iar $d(i, j)$ reprezintă minimul numărului de muchii ale lanțurilor de la i la j . Ori, știm că minimul dintre elementele unei multimi este mai mic sau egal decât orice element al multimi.

Definiția 1.2.2.2. Excentricitatea vârfului i este $\max\{d(i, j) \mid j \in X\}$ și o notăm cu $e(i)$.

Raza grafului G notată prin $r(G)$ este $\min\{e(i) \mid i \in X\}$.

Centrul grafului G este mulțimea vârfurilor i pentru care $e(i) = r(G)$.

Diametrul grafului G notat cu $d(G)$ este $\max\{e(i) \mid i \in X\}$.

Matricea distanțelor D într-un graf conex se poate calcula folosind rezultatul din teorema 1.2.2.1. Astfel, calculăm succesiv matricele A^p pentru $p = 1, 2, \dots, n$ sau până când se completează întreaga matrice D . Pentru un p oarecare vom completa cu valoarea p în matricea D acele poziții de elemente care nu au fost completate până la momentul respectiv și poziția corespunzătoare din A^p are valoare diferită de zero.

Observația 1.2.2.2.

Dacă în matricea distanțelor un element de pe diagonala principală are valoare finită (diferită de zero) atunci graful respectiv conține cel puțin un ciclu.

Teorema 1.2.2.2. Într-un graf conex G are loc:

$$r(G) \leq d(G) \leq 2r(G).$$

Demonstrație:

Prima inegalitate este adeverată deoarece minimul unei multimi este mai mic sau egal decât maximul aceleiași multimi. Pentru a demonstra a doua inegalitate să alegem un vârf i din centrul grafului G . Adică,

$$e(i) = r(G) = \min\{e(j) \mid j \in X\}.$$

Notăm cu k vârful pentru care $e(k) = d(G)$. În acest caz:

$$\exists j \in X: e(k) = d(k, j) \leq d(k, i) + d(i, j) \leq e(i) + e(i) = 2r(G). \quad \square$$

Teorema 1.2.2.3. Într-un arbore G are loc:

$$2r(G)-1 \leq d(G) \leq 2r(G).$$

Observația 1.2.2.2.

- a. Dacă G este orientat, atunci din $a_{ij} = 1$ deducem că în G există arcul (i, j) și vârful i este extremitatea sa inițială, iar vârful j este extremitatea sa terminală. Pe de altă parte fiecare arc din G induce în matricea de adiacență o singură valoare egală cu 1 (nенулă) deducem:

$$g^+(i) = \sum_{j=1}^n a_{ij} \quad \text{și} \quad g^-(i) = \sum_{j=1}^n a_{ji}, i \in X.$$

- b. Dacă G este neorientat, atunci din $a_{ij} = 1$ deducem că în G există muchia (i, j) și deci și $a_{ji} = 1$. Pe de altă parte fiecare muchie din G induce în matricea de adiacență două valori egale cu 1 (nenule). Deci:

$$g^+(i) = \sum_{j=1}^n a_{ij} = g^-(i) = \sum_{j=1}^n a_{ji}, i \in X.$$

Din această observație deducem:

Teorema 1.2.2.4. Pentru un graf G de ordin n și dimensiune m , suma gradelor vârfurilor este $2m$.

Demonstrație:

În baza observației 1.2.2.3:

- a. Pentru grafele orientate are loc:

$$\sum_{i=1}^n g^+(i) = \sum_{i=1}^n \sum_{j=1}^n a_{ij} = m = \sum_{i=1}^n \sum_{j=1}^n a_{ji} = \sum_{i=1}^n g^-(i).$$

Cum

$$g(i) = g^+(i) + g^-(i)$$

înseamnă că

$$\sum_{i=1}^n g(i) = \sum_{i=1}^n g^+(i) + \sum_{i=1}^n g^-(i) = 2m.$$

- b. Pentru grafele neorientate are loc

$$\sum_{i=1}^n g^+(i) = \sum_{i=1}^n \sum_{j=1}^n a_{ij} = 2m = \sum_{i=1}^n \sum_{j=1}^n a_{ji} = \sum_{i=1}^n g^-(i).$$

Cum

$$g(i) = g^+(i) = g^-(i)$$

înseamnă că

$$\sum_{i=1}^n g(i) = \sum_{i=1}^n g^+(i) + \sum_{i=1}^n g^-(i) = 2m. \square$$

Ca și aplicație considerăm problema:

Să se arate că un poligon convex cu n vârfuri are $\frac{n(n - 3)}{2}$ diagonale.

Un poligon convex cu n vârfuri, pentru care s-au dus toate diagonalele este graful complet K_n . În graful K_n fiecare vârf, dintre cele n , are gradul $n - 1$. Deci suma gradelor vârfurilor care este $2m$ are valoarea $n(n - 1)$. Prin urmare $m = \frac{n(n - 1)}{2}$. Numărul diagonalelor este egal cu numărul muchiilor grafului minus numărul laturilor. Deci:

$$m - n = \frac{n(n - 1)}{2} - n = \frac{n(n - 3)}{2}.$$

1.2.3. Reprezentarea cu tablouri unidimensionale

Dacă un graf are numărul de vârfuri mare și un număr relativ mic de muchii atunci matricele cu care se reprezintă graful au multe valori zero. Complexitatea căutării într-o asemenea matrice și cantitatea mare de suport (memorie) fac ca să crească complexitatea algoritmilor. O reprezentare mai eficientă în aceste cazuri este cea care folosește două tablouri (liste liniare), ce necesită o cantitate de memorie de aproximativ $n + m$ celule. În continuare prezentăm doar trei astfel de moduri de reprezentare.

a. Lista arcelor

În această reprezentare se utilizează tablourile α și β de dimensiune m . Valorile $\alpha(u)$ și $\beta(u)$ fiind extremitățile inițială respectiv finală ale arcului $u \in U$. Mai jos dăm o astfel de reprezentare pentru graful reprezentat geometric în figura 1.2.3.1.

u	1	2	3	4	5	6	7	8	9	10
$\alpha(.)$	1	1	2	3	1	5	5	7	6	4
$\beta(.)$	2	3	4	4	4	4	7	6	4	7

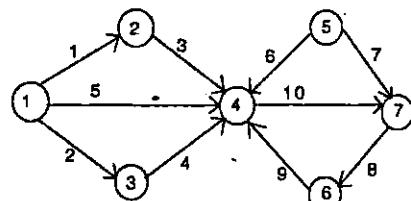


Fig. 1.2.3.1.

b. Lista succesorilor (valorile lui Γ)

În acest caz tabloul α are dimensiunea $n + 1$, iar β are dimensiunea m (pentru grafe orientate) sau $2m$ (pentru grafe neorientate). Pentru fiecare vârf $i \in X$, $\alpha(i)$ indică adresa (indicele) din tabloul β de unde sunt înregistrati succesorii săi. Succesorii vârfului i se află, deci, în tabloul β între pozițiile $\alpha(i)$ și $\alpha(i + 1) - 1$ inclusiv. Deci, dacă $\alpha(i) = \alpha(i + 1)$, atunci vârful i nu are nici un succesor.

Deci, pentru fiecare $i \in X$, au loc relațiile:

$$\alpha(i) = \sum_{j=1}^{i-1} g^+(j) + 1 \text{ și } \alpha(n+1) = m+1 \text{ sau } \alpha(n+1) = 2m+1$$

$$g^+(i) = \alpha(i+1) - \alpha(i) \quad (\text{la grafele orientate})$$

$$g(i) = \alpha(i+1) - \alpha(i) \quad (\text{la grafele neorientate})$$

$$\Gamma i = \{\beta(\alpha(i)), \beta(\alpha(i) + 1), \dots, \beta(\alpha(i + 1) - 1)\}.$$

Dacă arcele (muchiile) grafului au anumite ponderi (valori) acestea se pot păstra într-un alt tablou $p(.)$ de dimensiunea tabloului β .

Pentru exemplificare considerăm graful din figura 1.2.3.2 pentru care $n = 4$, și $m = 6$. Tablourile a și b conțin următoarele valori:

$i =$	1	2	3	4	5	
$\alpha(.)$	1	3	6	6	7	
$\beta(.)$	2	4	1	4	2	1

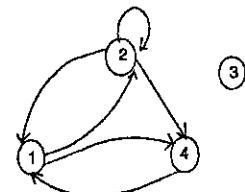


Fig. 1.2.3.2.

În mod analog se poate reprezenta un graf prin utilizarea funcției Γ^{-1} .

c. Lista predecesorilor (valorile lui Γ^{-1})

În acest caz tabloul α are dimensiunea $n + 1$, iar β are dimensiunea m (pentru grafe orientate) sau $2m$ (pentru grafe neorientate). Pentru fiecare vârf $i \in X$, $\alpha(i)$ indică adresa (indicele) din tabloul β de unde sunt înregistrati predecesorii săi. Predecesorii vârfului i se află, deci, în tabloul β între pozițiile $\alpha(i)$ și $\alpha(i + 1) - 1$ inclusiv. Dacă $\alpha(i) = \alpha(i + 1)$, atunci vârful i nu are nici un predecesor.

Deci, pentru fiecare $i \in X$, au loc relațiile:

$$\alpha(i) = \sum_{j=1}^{i-1} g^-(j) + 1 \text{ și } \alpha(n+1) = m+1 \text{ sau } \alpha(n+1) = 2m+1$$

$$g^-(i) = \alpha(i+1) - \alpha(i) \quad (\text{la grafele orientate})$$

$$g(i) = \alpha(i+1) - \alpha(i) \quad (\text{la grafele neorientate})$$

$$\Gamma^- i = \{\beta(\alpha(i)), \beta(\alpha(i) + 1), \dots, \beta(\alpha(i + 1) - 1)\}.$$

Dacă arcele (muchiile) grafului au anumite ponderi (valori) acestea se pot păstra într-un alt tablou $p(.)$ de dimensiunea tabloului β .

O problemă importantă este:

Să se elaboreze algoritmi și programe (proceduri) corespunzătoare pentru trecerea de la o reprezentare a unui graf la oricare altă.

Prezentăm în continuare algoritmi de conversie ai reprezentării unui graf dintr-o formă în altă formă.

Algoritmul 1.2.3.1.

Conversia reprezentării unui graf din matricea de adiacență A în matricea de incidentă B .

În elaborarea acestui algoritm folosim observația că fiecare valoare nenulă din A definește (referă) un arc și acestuia îi corespunde o coloană în B . Cum ordinea coloanelor în B nu are importanță, se poate parcurge matricea A fie pe linii fie pe coloane și pentru fiecare valoare nenulă găsită se va genera o coloană în matricea B . Indicii de linie și de coloană ai elementului nenul din A sunt indicii de linii în care noua coloană din B va avea elementele nenule 1 respectiv -1.

```

citeză n și A;
u:=0;
pentru i:=1,n execută
    pentru j:=1,n execută
        dacă  $a_{ij} \neq 0$  atunci
            u:=u+1
            pentru p:=1,n execută
                 $b_{pu} := 0;$ 
                sfpentru;
                 $b_{iu} := 1;$ 
                 $b_{ju} := -1;$ 
            sfdacă;
            sfpentru;
        sfpentru.
    
```

Algoritmul 1.2.3.2.

Conversia reprezentării unui graf din matricea de adiacență A în lista succesorilor definită de vectorii unidimensionali α și β .

În elaborarea acestui algoritm folosim observația că fiecare valoare nenulă din A definește (referă) un arc și acesta definește o poziție în vectorul β . Deci, numărul valorilor nenule din A este dimensiunea lui β . Dacă se parcurge matricea A pe linii valorile nenule depistate în ea, definesc predecesorii vârfului atașat acelei coloane. Acești predecesori sunt vârfurile ce corespund indicelui de linie în care se află valoarea nenulă. Dacă se introduc acești indici de linie succesiv în vectorul β , acesta va conține mulțimile $\Gamma^1, \Gamma^2, \dots, \Gamma^n$. La schimbarea coloanei se va reține în vectorul α poziția ultimului element memorat în β plus 1.

```

citeză n și A;
k:=1;
pentru i:=1,n execută
     $\alpha(i):=k;$ 

```

```

pentru j:=1,n execută
    dacă  $a_{ij} \neq 0$  atunci
         $\beta(k):=j;$ 
        k:=k+1;
    sfdacă;
    sfpentru;
    sfpentru;
     $\alpha(n+1):=k$ 

```

Algoritmul 1.2.3.3.

Conversia reprezentării unui graf din matricea de adiacență A în lista predecesorilor definită de vectorii unidimensionali α și β .

În elaborarea acestui algoritm folosim observația că fiecare valoare nenulă din A definește (referă) un arc și acesta definește o poziție în vectorul β . Deci, numărul valorilor nenule din A este dimensiunea lui β . Dacă se parcurge matricea A pe coloane valorile nenule depistate în ea, definesc predecesorii vârfului atașat acelei coloane. Acești predecesori sunt vârfurile ce corespund indicelui de linie în care se află valoarea nenulă. Dacă se introduc acești indici de linie succesiv în vectorul β , acesta va conține mulțimile $\Gamma^1, \Gamma^2, \dots, \Gamma^n$. La schimbarea coloanei se va reține în vectorul α poziția ultimului element memorat în β plus 1.

```

citeză n și A;
k:=1;
pentru j:=1,n execută
     $\alpha(j):=k;$ 
    pentru i:=1,n execută
        dacă  $a_{ij} \neq 0$  atunci
             $\beta(k):=i;$ 
            k:=k+1;
        sfdacă;
        sfpentru;
        sfpentru;
     $\alpha(n+1):=k$ 

```

Algoritmul 1.2.3.4.

Conversia reprezentării unui graf din matricea de incidentă B în matricea de adiacență A .

În elaborarea acestui algoritm folosim observația că fiecare coloană din B definește (referă) un arc și acestuia îi corespunde un element nenul (de valoare 1) în A . Poziția acestui element în matricea A este definită de indicii de

linie în care se află valorile 1 și -1 din coloana analizată a matricei B . Parcugerea matricei B o facem pe coloane.

```

citește n,m ((bij, i=1..n), j=1..m)
pentru i:=1..n execută
  pentru j:=1..n execută
    aij:=0;
    sfpentru;
    sfpentru;

  pentru u:=1..m execută
    pentru i:=1..n execută
      dacă biu=1 atunci p:=i;
      dacă biu=-1 atunci q:=i;
    sfpentru;
    apq:=1;
    sfpentru.
  
```

Algoritmul 1.2.3.5.

Conversia reprezentării unui graf din matricea de incidentă B în lista succesorilor definită de vectorii unidimensionali α și β .

În elaborarea acestui algoritm folosim observația că fiecare coloană din B definește (referă) un arc și acestuia îi corespunde un element în vectorul β . Pentru a depista și memora în β în ordine, elementele mulțimilor $\Gamma_1, \Gamma_2, \dots, \Gamma_n$ se parcurge matricea B pe linii. Vârful corespunzător unei linii are atâtia predecesori căte valori egale cu 1 sunt pe acea linie în matricea B . Astfel, dacă $b_{iu} = 1$, atunci j este succesor al lui i dacă $b_{ju} = -1$ altfel spus $u = (i, j)$. Prin urmare, j se va memora pe poziția următoare (liberă) în β .

```

citește n,m, ((bij, i=1..n), j=1..m);
k:=1;
pentru i:=1..n execută
  α(i):=k;
  pentru u:=1..m execută
    dacă biu=1 atunci
      pentru j:=1..n execută
        dacă bju=-1 atunci q:=j sfdacă;
      sfpentru;
      β(k):=q;
      k:=k+1;
      sfdacă;
    sfpentru;
    sfpentru;
  α(n+1):=m+1.
  
```

Algoritmul 1.2.3.6.

Conversia reprezentării unui graf din matricea de incidentă B în lista predecesorilor definită de vectorii unidimensionali α și β .

În elaborarea acestui algoritm folosim observația că fiecare coloană din B definește (referă) un arc și acestuia îi corespunde un element în vectorul β . Pentru a depista și memora în β corect succesiunea elementelor mulțimilor $\Gamma_1, \Gamma_2, \dots, \Gamma_n$ se parcurge matricea B pe linii. Vârful corespunzător unei linii are atâtia predecesori căte valori egale cu -1 sunt pe acea linie în matricea B . Astfel, dacă $b_{iu} = -1$, atunci j este predecesor al lui i dacă $b_{ju} = 1$. Prin urmare j se va memora pe poziția următoare (liberă) în β .

```

citește n,m, ((bij, i=1..n), j=1..m);
k:=1;
pentru i:=1..n execută
  α(i):=k;
  pentru u:=1..m execută
    dacă biu=-1 atunci
      pentru j:=1..n execută
        dacă bju=1 atunci q:=j sfdacă;
      sfpentru;
      β(k):=q;
      k:=k+1;
      sfdacă;
    sfpentru;
    sfpentru;
  α(n+1):=m+1.
  
```

Algoritmul 1.2.3.7.

Conversia reprezentării unui graf din lista succesorilor, cu ajutorul vectorilor α și β , în matricea de adiacență A .

Pentru a construi pe A , după ce inițializăm matricea cu 0, vom transforma în 1 valorile de pe pozițiile $(i, \beta(j))$ pentru $\alpha(i) \leq j < \alpha(i + 1)$ și fiecare $i = 1, n$.

- citește n, (α(i), i=1..n+1);
citește m, (β(i), i=1..m);
- pentru i:=1..n execută


```

        pentru j:=1..n execută
          aij:=0;
          sfpentru;
        sfpentru;
      
```

```

pentru i:=1,n execută
  pentru j:=α(i), α(i+1)-1 execută
    p:=β(j);
    αp:=I;
  sfpentru;
sfpentru.

```

Algoritmul 1.2.3.8.

Conversia reprezentării unui graf din lista succesorilor, cu ajutorul vectorilor α și β , în matricea de incidentă B .

Fiecare element din β va genera o coloană în matricea B . În această coloană pe linia i trebuie să fie plasată valoarea 1, iar pe linia p valoarea -1 dacă $p = \beta(j)$ cu $\alpha(i) \leq j < \alpha(i + 1)$, deoarece vârful p este succesor al vârfului $p = \beta(j)$.

```

(a) citește n,(α(i),i=1,n+1);
  citește m,(β(i),i=1,m);
(b) pentru i:=1,n execută
  pentru j:=1,m execută
    bij:=0;
  sfpentru;
sfpentru;

u:=0;
pentru i:=1,n execută
  pentru j:=α(i), α(i+1)-1 execută
    u:=u+1;
    bui:=1;
    p:=β(j);
    bpu:=−1
  sfpentru;
sfpentru.

```

Algoritmul 1.2.3.9.

Conversia reprezentării unui graf din lista succesorilor, cu ajutorul vectorilor α și β , în lista predecesorilor cu ajutorul vectorilor a și b .

Această conversie se poate realiza în două etape de exemplu prin utilizarea a doi algoritmi deja descriși sau descrierea altor etape. În prima variantă construim din lista succesorilor o matrice (de adiacență sau de incidentă) și apoi din acea matrice convertim reprezentarea grafului în lista predecesorilor. În cealaltă variantă am putea determina la o parcurgere a vectorilor α și β multimile $\Gamma_1, \Gamma_2, \dots, \Gamma_n$ (sub formă de șiruri) și cardinalele

lor, iar apoi vom da valori elementelor din tablourile a și b . Descriem cea de-a doua variantă. La prima parcurgere a vectorilor α și β vom construi matricea C de dimensiune $n \times n$ în care pentru $i = 1, n$ linia i va conține în $c(i,1)$ numărul predecesorilor lui i , iar pe următoarele $c(i,1)$ poziții vor fi memorate acești predecesori. Deci, numărul de poziții ocupate pe liniile matricei C diferă. La inițializare coloana 1 a matricei C are toate elementele 0, adică nici unui vârf al grafului nu îi s-a depistat vreun predecesor. Mai precizăm că dacă pentru valorile variabilelor i, k, p are loc $\beta(p) = k$ și $\alpha(i) \leq p < \alpha(i + 1)$, atunci se deduce (interpretează) că vârful k este succesor al vârfului i și că vârful i este predecesor al vârfului k . Din valorile coloanei 1 vom calcula valorile vectorului a , iar celelalte valori ale matricei C vor forma vectorul b (lista predecesorilor).

```

(a) citește n,(α(i),i=1,n+1);
  citește m,(β(i),i=1,m);
(b) pentru i:=1,n execută
  c(i,1):=0;
  sfpentru;

pentru i:=1,n execută
  pentru p:=α(i), α(i+1)-1 execută
    k:=β(p)
    t:=c(k,1)+1;
    c(k,1):=t;
    c(k,t+1):=i;
  sfpentru;
sfpentru;

t:=1;
pentru i:=1,n execută
  a(i):=t;
  pentru j:=1,c(i,1);
    b(t):=c(i,j+1);
    t:=t+1;
  sfpentru;
sfpentru;
a(n+1):=t.

```

Algoritmul 1.2.3.10.

Conversia reprezentării unui graf din lista predecesorilor, cu ajutorul vectorilor α și β , în matricea de adiacență A .

Pentru a construi pe A după ce inițializăm matricea cu 0, vom transforma în 1 valorile de pe pozițiile $(\beta(j), i)$ pentru $\alpha(i) \leq j < \alpha(i + 1)$ și fiecare $i = 1, n$.

```

(a) citește  $n, (\alpha(i), i=1, n+1)$ ;
    citește  $m, (\beta(i), i=1, m)$ ;
(b) pentru  $i:=1, n$  execută
    pentru  $j:=1, n$  execută
         $a_{ij}:=0$ ;
        sfpentru;
        sfpentru;

    pentru  $i:=1, n$  execută
        pentru  $j:=\alpha(i), \alpha(i+1)-1$  execută
             $p:=\beta(j)$ ;
             $a_{pi}:=1$ ;
            sfpentru;
            sfpentru.

```

Algoritmul 1.2.3.11.

Conversia reprezentării unui graf din lista predecesorilor, cu ajutorul vectorilor α și β , în matricea de incidență B .

Fiecare element din β va genera o coloană în matricea B . În această coloană pe linia i trebuie să fie plasată valoarea -1, iar pe linia p valoarea 1 dacă $p = \beta(j)$ cu $\alpha(i) \leq j \leq \alpha(i + 1) - 1$, deoarece vârful i este predecesor al vârfului $p = \beta(j)$.

```

(a) citește  $n, (\alpha(i), i=1, n+1)$ ;
    citește  $m, (\beta(i), i=1, m)$ ;
(b) pentru  $i:=1, n$  execută
    pentru  $j:=1, m$  execută
         $b_{ij}:=0$ ;
        sfpentru;
        sfpentru;

         $u:=0$ ;
        pentru  $i:=1, n$  execută
            pentru  $j:=\alpha(i), \alpha(i+1)-1$  execută
                 $u:=u+1$ ;
                 $b_{iu}:=1$ ;
                 $p:=\beta(j)$ ;
                 $b_{pu}:=1$ 
            sfpentru;
            sfpentru.

```

Algoritmul 1.2.3.12.

Conversia reprezentării unui graf din lista predecesorilor, cu ajutorul vectorilor α și β , în lista succesorilor cu ajutorul vectorilor a și b .

Această conversie se poate face în două etape de exemplu prin utilizarea a doi algoritmi deja descriși sau descrierea altor etape. În prima variantă construim din lista predecesorilor o matrice (de adiacență sau de incidență) și apoi din acea matrice convertim reprezentarea grafului în lista succesorilor. În cealaltă variantă am putea determina la o parcurgere a vectorilor α și β mulțimile $\Gamma^1, \Gamma^2, \dots, \Gamma^n$ (sub formă de siruri) și cardinalele lor, iar apoi vom atribui valori elementelor din tablourile a și b . Descriem cea de-a doua variantă. La prima parcurgere a vectorilor α și β vom construi matricea C de dimensiune $n \times n$ în care pentru $i = 1, n$, linia i va conține în $c(i, 1)$ numărul succesorilor lui i , iar pe următoarele $c(i, 1)$ poziții vor fi memorate acești succesori. La inițializare coloana 1 a matricei C are toate elementele 0, adică nici unui vârf al grafului nu își depistă vreun succesor. Mai precizăm că dacă pentru valorile variabilelor i, k, p are loc $\beta(p) = k$ și $\alpha(i) \leq p < \alpha(i + 1)$, atunci se deduce că vârful k este predecesor al vârfului i și că vârful i este succesor al vârfului k . Din valorile coloanei 1 vom calcula valorile vectorului a , iar celealte valori ale matricei C vor forma vectorul b al succesorilor.

```

citește  $n, (\alpha(i), i=1, n+1)$ ;
citește  $m, (\beta(i), i=1, m)$ ;

pentru  $i:=1, n$  execută
     $c(i, 1):=0$ ;
    sfpentru;

pentru  $i:=1, n$  execută
    pentru  $p:=\alpha(i), \alpha(i+1)-1$  execută
         $k:=\beta(p)$ 
         $t:=c(k, 1)+1$ ;
         $c(k, t):=i$ ;
         $c(k, t+1):=i$ ;
        sfpentru;
        sfpentru;

         $t:=1$ ;
        pentru  $i:=1, n$  execută
             $a(i):=t$ ;
            pentru  $j:=1, c(i, 1)$  execută
                 $b(t):=c(i, j+1)$ ;
                 $t:=t+1$ ;
            sfpentru;
            sfpentru;
             $a(n+1):=t$ .

```

1.3. Numere fundamentale în teoria grafelor

1.3.1. Număr de stabilitate internă

Definiția 1.3.1.1. Fie graful orientat $G = (X, U)$. Mulțimea $S \subset X$ se numește **mulțime stabilă interior (independentă)** dacă și numai dacă $\forall i \in S$ are loc $\Gamma_i \cap S = \emptyset$.

Proprietatea lui S se mai poate exprima:

$$\Gamma S \cap S = \emptyset$$

sau

$$\forall i, j \in S \Rightarrow (i, j) \notin U$$

sau

graful $G(S)$ este format numai din vârfuri izolate.

Din definiție se deduce următorul rezultat.

Propoziția 1.3.1.1. Dacă \mathfrak{S} este familia mulțimilor interior stabile ale grafului G atunci au loc:

- a. $\Phi \in \mathfrak{S}$;
- b. $\forall i \in X, \{i\} \in \mathfrak{S}$;
- c. dacă $S \in \mathfrak{S}$ și $A \subset S$ atunci $A \in \mathfrak{S}$.

O problemă cu aplicații și în colorarea grafelor este determinarea de mulțimi interior stabile (independente). Din propoziția de mai sus se observă că, toate mulțimile de vârfuri de cardinal cel mult 1 sunt mulțimi interior stabile și că toate submulțimile unei mulțimi interior stabile sunt și ele interior stabile. Prin urmare de interes, ar fi să se determine mulțimi interior stabile cât mai mari.

Exemplul 1.3.1.1.

Se consideră graful din figura 1.3.1.1. Mulțimea $S = \{1, 3, 5\}$ este o mulțime interior stabilă a acestui graf.

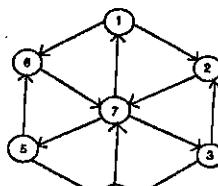


Fig. 1.3.1.1.

Definiția 1.3.1.2. Număr de stabilitate internă a grafului G este $\max \{ |S| \mid S \in \mathfrak{S} \}$ și se notează cu $\alpha(G)$.

Orice mulțimea interior stabilă S pentru care $|S| = \alpha(G)$ se numește **mulțime interior stabilă maximă**.

Să remarcăm că într-o familie de mulțimi, cum este de exemplu \mathfrak{S} , există și noțiunea de **mulțime maximală** dar acest element este în raport cu incluziunea mulțimilor, care este o relație de ordine în general parțială. Altfel spus, $S \in \mathfrak{S}$ este maximală în \mathfrak{S} dacă $\forall A \in \mathfrak{S}$ cu $S \subset A$ implică $S = A$.

Observația 1.3.1.1.

Pentru ca o mulțime S să fie interior stabilă trebuie ca $\Gamma S \cap S = \emptyset$, iar pentru ca ea să fie mulțime interior stabilă maximă, dacă graful este fără bucle, trebuie în plus $\Gamma S \cup S = X$.

Exemplul 1.3.1.2. Pentru graful din figura 1.3.1.2. au loc:

- a. $\mathfrak{S} = \{\emptyset, \{1\}, \{2\}, \{3\}, \{1, 3\}\}$;
- b. mulțimi interior stabile maxime sunt $\{2\}$ și $\{1, 3\}$;
- c. graful are o singură mulțime interior stabilă maximă, care este $\{1, 3\}$;
- d. $\alpha(G)=2$.



Fig. 1.3.1.2.

Problema 1.3.1.1.

Fiind dat graful $G = (X, U)$ să se determine toate mulțimile interior stabile ale lui G .

Determinarea mulțimilor interior stabile se poate face prin a parcurge toate submulțimile lui X și să se verifice dacă sunt sau nu mulțimi stabile. O altă metodă este de tip backtracking. O prezentare generală a metodei (a algoritmului) backtracking, pentru cazul în care se știe că problema are soluție, este descrisă în pseudocod în cele ce urmează.

CITEȘTE datele problemei;

INITIALIZEAZĂ;

AFIȘEAZĂ;

Repetă

Cât_fimp NU_ESTE_GATA execută

Dacă SE_POATE atunci AVANSEAZĂ; sfđ;

sfc;

*AFIȘEAZĂ;
REVENIRE;
până când NU_MAI_POATE;*

Deci pentru fiecare implementare în parte utilizatorul își va defini conform problemei de rezolvat procedurile: *CITEȘTE, INITIALIZEAZĂ, AVANSEAZĂ, AFIȘEAZĂ* și funcțiile booleene *NU_ESTE_GATA* și *SE_POATE*. Valoarea variabilei booleene *NU_MAI_POATE* se stabilește în procedura *REVENIRE*. Primul apel al procedurii *AFIȘEAZĂ* s-a făcut pentru că și valoarea inițială este o soluție a problemei.

Vom prezenta în continuare un program Pascal care rezolvă problema de determinare a mulțimilor interior stabilă cu metoda backtracking. La fiecare moment mulțimea de vârfuri $X = \{1, 2, \dots, n\}$, ale grafului, este descompusă în partea $A \cup B$, unde A este o mulțime interior stabilă, iar B este complementara ei. Astfel, $A = \{i \in X \mid o(i) \neq 0\}$, iar $B = \{i \in X \mid o(i) = 0\}$. Explorarea vârfurilor lui X se face în ordinea crescătoare a vârfurilor, adică $1, 2, \dots, n$. Starea vârfului i este definită de valoarea $o(i)$, $1 \leq i \leq n$. Faptul că $o(i) \neq 0$, pentru un i oarecare, se interprează că vârful i face parte din mulțimea A . Valoarea na este vârful de la care (până la n) se va căuta în continuare un vârf care să se poată adăuga mulțimii A găsită până la acel moment. Valorile $o(i) \neq 0$, pentru fiecare vârf i cu această proprietate (și care vârf face parte din A) precizează vârful începând cu care se va căuta (până la care a fost căutat) următorul vârf lui i din A . Mulțimea A este complet determinată dacă s-a făcut explorarea tuturor vârfurilor (adică s-a ajuns ca $na = n$). La fiecare revenire (cu un pas) se determină precedentul vârf i din A , al ultimului explorat, și acestuia i se caută un alt succesor dintre cele de la $o(i)$ la n . Considerăm că graful se dă prin mulțimile Γ_i , $1 \leq i \leq n$.

```
Program Mis_cu_BackT;
Uses Crt;
Const vmax=40;
Var n,i,na,j,x,incep:Word;
    Nu_Mai_Poate:Boolean;
    Gama : Array [1..vmax] of Set of 1..vmax;
    o : Array [1..vmax+1] of Word;

Procedure Citeste;
Var a: Array[1..10] of string[10];
Begin
  n:=6;
  a[1]:='010110';
  a[2]:='000100';
  a[3]:='010010';
  a[4]:='000010';
  a[5]:='010000';
  a[6]:='000110';
  For i:=1 to n Do begin
    Gama[i]:={};
    for j:=1 to n Do
```

```
      If a[i,j] ='1' then Gama[i]:=gama[i]+{j};
    end;
  x:=0;
End;

Procedure Initializeaza;
Begin
  For i:=1 to n Do o[i]:=0;
  o[n+1]:=1; na:=1; incep:=1; ClrScr;
  Nu_Mai_Poate:=False;
End;

Function Nu_Este_Gata : Boolean;
Begin
  Nu_Este_Gata:=na<=n;
End;

Function Se_Poate:Boolean;
Var Reia:Boolean;
Begin
  i:=na-1; Se_Poate:=False;
  If i<n then
  begin
    Repeat
      Reia:=True; i:=i+1;
      For j:=1 to na-1 Do
        If (o[j]<>0) then
          If (i in Gama[j]) or (j in Gama[i]) then Reia:=False;
        Until (i=n) or Reia ;
        Se_Poate:=Reia;
    end;
    na:=i+1
  End;
End;

Procedure Avanseaza;
Begin
  o[i]:=na;
End;

Procedure Afiseaza;
Begin
  x:=x+1;
  Write('Mulțimea interior stabilă a ',x,'-a este:');
  Write(' { ');
  For i:=1 to n Do
    If o[i]<>0 then Write(i,' ');
  Writeln(Chr(8),' }');
End;

Procedure Revenire;
Begin
  i:=n+1;
  Repeat i:=i-1; until (o[i]<>0) or (i=0);
  If i>0 then
  begin
    na:=i+1; o[i]:=0;
    Repeat i:=i-1; until (o[i]<>0) or (i=0);
  end;
End;
```

```

If i>0 then o[i]:=na
else
begin
    incep:=incep+1; na:=incep;
    If incep<=n+1 then For i:=1 to n Do o[i]:=0
        else Nu_Mai_Poate:=True;
end;
End;

Begin
    Citeste;
    Initializeaza;
    Afiseaza;
    Repeat
        While Nu_Este_Gata Do
            If Se_Poate Then Avanseaza;
            Afiseaza;
            Revenire;
        Until Nu_Mai_Poate;
    readln;
End..

```

Procedura "Citeste" de mai sus utilizează matricea de adiacență dată în tabloul A. Valoarea lui n și elementele tabloului A trebuie modificate de către utilizator în instrucțiunile de atribuire în corpul procedurii. O altă procedură "Citeste" care cere datele utilizatorului după lansarea în execuție a programului ar putea fi următoarea:

```

Procedure Citeste;
Begin
    ClrScr;
    Writeln('Dati numarul de virfuri ale grafului');
    Readln (n);
    For i:=1 to n Do
    begin
        Gama[i]:=[]; ClrScr;
        Writeln('Dati succesorii virfului ',i:3,' si in final 0');
        Readln(x);
        While x<>0 Do
        begin
            If (x>0) and (x<=n+1) then Gama[i]:=Gama[i]+[x];
            Readln(x);
        end;
        end;
        x:=0;
    End;

```

Exemplul 1.3.1.3. Problema celor opt dame

Pe o tablă de șah se pot așeza opt dame astfel încât nici una să nu fie atacată de o altă damă?

Această problemă a fost formulată de către Gauss. Rezolvarea ei revine la a preciza dacă există mulțimi interior stabile cu cel puțin opt elemente în graful construit după cum se prezintă în continuare. Mulțimea vârfurilor are 64 de elemente corespunzător celor 64 de pătrate ale tablei de șah. Se definește $\Gamma_i = \{j \mid i \text{ și } j \text{ sunt pe aceeași linie sau diagonală}\}$. Deci $j \in \Gamma_i$ înseamnă că damele așezate în pătratele i și j se atacă reciproc. Deoarece tabla de șah are opt linii (și mulțimi interior stabile cu mai mult de 8 elemente nu există) rezolvarea problemei revine la a determina mulțimi stabile interior maxime. Dacă acestea au opt elemente atunci răspunsul la întrebarea pusă este afirmativ.

Exemplul 1.3.1.4.

Determinarea ciclicilor dintr-un graf. Fiind dată o mulțime de persoane sau grupuri de persoane (partide, asociații etc.) care se pot simpatiza sau nu se pune problema determinării unei submulțimi de elemente (alianțe) care se simpatizează reciproc cu un număr maxim de participanți.

Considerând graful neorientat în care $i \in \Gamma_j$ dacă i și j se simpatizează reciproc, problema revine la a determina o mulțime interior stabilă maximă în graful complementar.

Vom prezenta în continuare o altă problemă rezolvată cu metoda backtracking, dar care se deosebește de prima metodă prin faptul că problema nu este sigur că are soluție și nici prin inițializare nu se obține o soluție.

Problema 1.3.1.2.

Se dă un graf $G = (X, U)$, neorientat de ordin n și de dimensiune m . Se cere să se precizeze dacă G este hamiltonian (conține cicluri hamiltoniene) și în caz afirmativ să se determine un ciclu hamiltonian în G .

Pentru graful G vom utiliza reprezentarea sa cu ajutorul listei succesorilor definită de vectorii α și β , presupunem că dimensiunea lui β este dublul numărului de muchii. Înainte de a descrie algoritmul facem câteva precizări necesare pentru înțelegerea sa. Orice ciclu hamiltonian este un ciclu elementar (deci fiecare vîrf este utilizat de ciclu o singură dată) și are n vîrfuri. Prin urmare, problema cere să determinăm valorile $x_1, x_2, \dots, x_n, x_{n+1} = x_1$ din X cu proprietățile $X = \{x_1, x_2, \dots, x_n\}$ și $(x_i, x_{i+1}) \in U$, pentru $i = 1, n$. Orice succesiune x_1, x_2, \dots, x_i cu $1 \leq i \leq n$ este un lanț elementar de lungime $i + 1$.

Deoarece $x_1, x_2, \dots, x_n, x_{n+1}$ este o permutare circulară a celor n vîrfuri ale grafului, fără a restrînge generalitatea, cîchările se vor preciza (citi) astfel

încât $x_1 = 1$. Astfel, se poate considera că $x_1 = 1$ și deci algoritmul încearcă să adauge la fiecare etapă un vârf (muchie) la lanțul elementar descoperit în graf până la acea etapă. Dacă un asemenea vârf, care să fie adăugat lanțului existent, există algoritmul avansează, altfel face o revenire, micșorând lungimea lanțului existent.

Ca variabilă de lucru folosim tabloul unidimensional o_i cu $i \in X$. Valorile acestui șir la pasul t al algoritmului (când se caută valoarea pentru x_i) au următoarele semnificații:

- Dacă $o_i \neq 0$, atunci vârful i este selectat în lanțul elementar ales până la acel moment.
- Dacă $o_i = 0$, atunci fie vârful i este ultimul vârf selectat în lanțul elementar ales până la acel moment și urmează să se determine succesorul său, fie i nu a fost selectat în lanțul elementar ales până la acel moment.
- Dacă $o_i = j$, atunci în lanțul elementar ales până la acel moment vârful i este urmat de vârful $\beta(j)$. Adică, există un $k < t$ astfel încât $x_k = i$ și $x_{k+1} = \beta(j)$.

Acum putem trece la descrierea algoritmului.

(a) etapa de inițializare

```
citește n,m, (α(i),i=1,n+1),(β(i),i=1,m)
x(1)=1;
vx:=1;
caut:=alfa[1];
t:=1;
pentru i:=1,n execută o(i):=0; x(i+1):=0 sfpentru;
```

(b) iterată de bază

```
repetă
    cît_jimp NU_EXISTĂ_SOLUȚIE și not NU_MAI_POATE_EXPLORA
    execută
        Dacă SE_POATE atunci AVANSEAZĂ
            altfel REVENIRE;
            sfj;
        sfj;
    Dacă not NU_EXISTĂ_SOLUȚIE atunci AFIȘEAZĂ;
    până_când NU_MAI_POATE_EXPLORA;
```

Algoritmul folosește:

- funcția booleană *NU_EXISTĂ_SOLUȚIE*

Are valoarea 'true' dacă încă nu s-a depistat un ciclu hamiltonian (se dispune doar de un lanț elementar de lungime mai mică decât n) și 'false' dacă s-a depistat un ciclu hamiltonian în graful dat.

- funcția booleană *NU_MAI_POATE_EXPLORA*

Are valoarea 'true' dacă au fost epuizate toate variantele de analiză (prin revenire se depășește vârful 1) și 'false' dacă mai sunt variante de analizat (se mai pot face reveniri sau avansări în analiză).

- funcția booleană *SE_POATE*

Are valoarea 'true' dacă lanțul elementar se poate dezvolta prin adăugarea unui nou vârf (această adăugare o va face procedura *AVANSEAZĂ*) și 'false' dacă lanțului elementar existent nu îl poate adăuga nici un vârf prin urmare se va face un pas înapoi pentru a face noi căutări (acest pas înapoi îl va face procedura *REVENIRE*).

- procedura *AFIȘEAZĂ*

Va afișa valorile elementelor șirului x , care constituie succesiunea de vârfuri ce formează un ciclu hamiltonian.

Programul Turbo Pascal corespunzător algoritmului descris mai sus este:

```
Program hamiltonian;
Uses Crt;
Const vmax=20;
      mmax=350;
Var n,i,vx,ib,caut,t,nr_ham:word;
     x,o,alfa:array[1..vmax] of 0..vmax;
     beta:array[1..mmax] of 1..vmax;
Procedure Citeste;
Var a:array[1..vmax] of string[vmax];
    k,j:word;
begin
  n:=6;
  a[1]:='011011';
  a[2]:='101110';
  a[3]:='110110';
  a[4]:='011011';
  a[5]:='111101';
  a[6]:='100110';
```

```

k:=1;
for i:=1 to n do
begin
  alfa[i]:=k;
  for j:=1 to n do
    if a[i,j]='1' then begin beta(k):=j;
                           k:=k+1;
                           end;
end;
alfa[n+1]:=k;
nr_ham:=0;
End;

Procedure Initializare;
Begin
  x[1]:=1;
  vx:=1;
  caut:=alfa[1];
  t:=1;
  for i:=1 to n do
    begin
      o[i]:=0;
      x[i+1]:=0;
    end;
End;

Function Nu_Este_Gata : Boolean;
Begin
  Nu_Este_Gata:=x[n+1]<>1;
End;

Function Nu_Mai_Poate : Boolean;
Begin
  Nu_Mai_Poate:=t=0;
End;

Function Se_Poate : Boolean;
var vb,capat:Word;
  locala:boolean;
Begin
  locala:=false;
  ib:=x[vx]+1;
  capat:=alfa[ib];
  ib:=caut;
  If vx < n then
  begin
    While (ib<capat) and (not locala) do
    begin
      vb:=beta(ib);
      if o[vb]<>0 then ib:=ib+1
                    else begin locala:=true; t:=vb; end;
    end;
  end else
  begin
    While (ib<capat) and (not locala) do
    begin
      vb:=beta(ib);
      if vb<>1 then ib:=ib+1
    end;
  end;
End;

```

```

else begin locala:=true; t:=1; end;
end;
Se_Poate:=locala
End;

Procedure Avanseaza;
Begin
  x[vx+1]:=t;
  caut:=alfa[t];
  t:=x[vx];
  o[t]:=ib+1;
  vx:=vx+1;
End;

Procedure Revenire;
Begin
  if vx>1 then begin
    t:=x[vx];
    o[t]:=0;
    x[vx]:=0;
    vx:=vx-1;
    t:=x[vx];
    caut:=o[t];
  end
  else t:=0;
End;

Procedure Afiseaza;
Begin
  nr_ham:=nr_ham+1;
  Write('Ciclul hamiltonian al ',nr_ham,'-lea este:');
  Write('(');
  for i:=1 to n do Write(x[i],', ');
  write(x[1],', ');
  WriteIn(chr(8),')');
  x[n+1]:=0;
End;

Begin
  Citeste;
  Initializare;
  Repeat
    While Nu_Este_Gata and (not Nu_Mai_Poate) Do
      If Se_Poate then Avanseaza
                    else Revenire;
    If not Nu_Este_Gata then Afiseaza;
    until Nu_Mai_Poate; readln;
End.

```

Programul dat rezolvă problema pentru graful din figura 1.3.1.3.

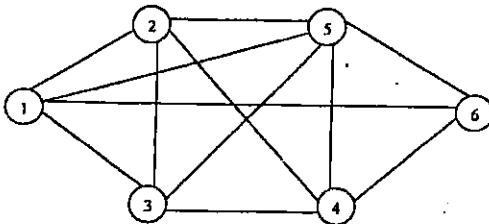


Fig.1.3.1.3.

Grafului dat nu lipsesc 3 muchii pentru a fi K_6 , care are $5!/2 = 60$ cicluri hamiltoniene. Graful din figura 1.3.1.3. are 14 cicluri hamiltoniene distincte, dar cum fiecare ciclu i se pot citi varfurile pe cele două sensuri (trigonometric sau în sensul acelor de ceasornic) programul va afisa cele 28 de succesiuni de varfuri în care se pot citi cele 14 cicluri hamiltoniene. Dacă se dorește ca programul să afiseze doar ciclurile hamiltoniene distincte atunci o modificare a programului ar fi înlocuirea ultimei instrucțiuni 'if' a programului (antepenultimul rând din program) cu

```
If not Nu_Este_Gata then
  If x[2] < x[n] then Afiseaza else x[n+1]:=0;
```

1.3.2. Număr de stabilitate externă

Definiția 1.3.2.1. Fie $G = (X, U)$ un graf orientat. O mulțime $T \subset X$ se numește **mulțime exterior stabilă (dominantă)** dacă $\Gamma_i \cap T \neq \emptyset$, $\forall i \in X - T$.

Observația 1.3.2.1.

O mulțime exterior stabilă T se mai poate defini și prin relația

$$T \cup \Gamma^1 T = X \quad \text{sau} \quad X - T \subset \Gamma^1 T \quad \text{sau} \quad \Gamma(X - T) \subset T.$$

Demonstrație:

Cum T și $\Gamma^1 T \subset X$ rezultă $T \cup \Gamma^1 T \subset X$. Să demonstrăm că $X \subset T \cup \Gamma^1 T$. Fie $i \in X$

- dacă $i \in T \Rightarrow i \in T \cup \Gamma^1 T$;
- dacă $i \notin T \Rightarrow i \in X - T$ și T dominantă $\Rightarrow \Gamma_i \cap T \neq \emptyset \Rightarrow \exists j \in \Gamma_i \cap T \Rightarrow i \in \Gamma^1 j \subset \Gamma^1 T \subset T \cup \Gamma^1 T$.

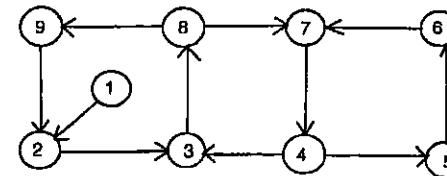


Fig. 1.3.2.1.

Exemplul 1.3.2.1.

Pentru graful din figura 1.3.2.1 mulțimea $T = \{2, 3, 5, 7\}$ este o mulțime exterior stabilă.

Propoziția 1.3.2.1. Dacă notăm cu \wp familia mulțimilor exterior stable ale grafului G atunci:

- $X \in \wp$;
- dacă $T \in \wp$ și $T \subset A \subset X$ atunci $A \in \wp$;
- dacă T_1 și $T_2 \in \wp$ atunci $T_1 \cup T_2 \in \wp$.

Definiția 1.3.2.2. Număr de stabilitate externă a grafului G este $\min\{|T| \mid T \in \wp\}$ și se notează cu $\beta(G)$.

Dacă T este o mulțime stabilă exterior cu $|T| = \beta(G)$ atunci spunem că T este **mulțime stabilă exterior minimă**.

Exemplul 1.3.2.2.

Pentru graful din figura 1.3.2.2. au loc:

- $\wp = \{\{2\}, \{1, 3\}, \{1, 2, 3\}\}$;
- mulțimile exterior stable minime sunt $\{2\}$ și $\{1, 3\}$;
- graful are o singură mulțime exterior stabilă minimă și anume $\{2\}$;
- $\beta(G) = 1$.



Fig.1.3.2.2.

Exemplul 1.3.2.3. Problema gardienilor

La o închisoare fiecare celulă trebuie supravegheată. Unele celule comunică prin culoare rectilinii. Deci gardianul care se află lângă o celulă poate supraveghea și pe celelalte celule cu care comunică aceasta printr-un culoar rectiliniu. Problema cere să se stabilească numărul minim de gardieni, pe schimb, necesari pentru supravegherea tuturor celulelor închisorii.

Cunoscând graful neorientat ce are ca vârfuri celulele închisorii și convenind $(i, j) \in U$ dacă și numai dacă celulele i și j comunică printr-un culoar rectiliniu, problema revine la a determina numărul de stabilitate externă a grafului.

O problemă asemănătoare este stabilirea numărului minim de polițiști care să supravegheze toate intersecțiile unui sistem de străzi dat.

Exemplul 1.3.2.4. Problema celor cinci dame

Pentru jocul de șah se cere plasarea unui număr minim de dame (cai, nebuni, turnuri) pe tablă astfel ca fiecare patrat al tablei să fie dominat (atacat) de cel puțin una dintre piesele plasate.

Considerând graful neorientat ce are 64 de vârfuri corespunzătoare patrelor tablei de șah și convenind $(i, j) \in U$ dacă și numai dacă patratul i este atacat de piesa de pe patratul j și invers. În acest graf $\forall i \in X$ are loc $\Gamma^{-1}i = \Gamma i$ și se demonstrează ușor că numărul minim de piese ce domină toate patratele tablei de șah este numărul de stabilitate externă al grafului.

Pentru determinarea de mulțimi exterior stabile (dominante) se pot utiliza rezultatele următoare. Fiind dat graful $G = (X, \Gamma)$ îl atașăm acestuia graful bipartit $G' = (X \cup X', \Delta)$ cu $X' = \{i' \mid i \in X\}$. Pentru $j \in X$ convenim $i' \in \Delta j \Leftrightarrow i = j$ sau $j \in \Gamma i$. Folosind graful G' mulțimile exterior stabile le caracterizăm cu teorema ce urmează.

Teorema 1.3.2.1. $T \subset X$ este mulțime exterior stabilă în graful G dacă și numai dacă $\Delta T = X' \cap G'$.

Demonstrație:

Necesitatea (\Rightarrow):

Se dă T o mulțime exterior stabilă în G . Prin urmare

$$T \neq \Phi \text{ și } T \cup \Gamma^{-1}T = X$$

Fie acum $i' \in X'$; cum $i \in X = T \cup \Gamma^{-1}T$

dacă $i \in T$ atunci $i' \in \Delta T$,

iar

dacă $i \in \Gamma^{-1}T$ atunci $\exists j \in \Gamma i \subset T$

deci

$$i' \in \Delta j \subset \Delta T$$

Așadar $X' \subset \Delta T$, dar $\Delta T \subset X'$ și deci $\Delta T = X'$.

Suficiență (\Leftarrow):

Fie $T \subset X$ cu $\Delta T = X'$ să demonstrăm că T este exterior stabilă.

Fie $i \in X$,

dacă $i \in T$ cum $\Delta T = X'$ atunci $\exists k \in T$ astfel încât $i' \in \Delta k$

deci

$$k \in \Gamma i$$

prin urmare

$$i \in \Gamma^{-1}k \subset \Gamma^{-1}T \subset T \cup \Gamma^{-1}T$$

Am demonstrat $X \subset T \cup \Gamma^{-1}T$, dar $T \cup \Gamma^{-1}T \subset X'$ și deducem că $X = T \cup \Gamma^{-1}T$. În consecință T este mulțime exterior stabilă. \square

Folosind această teoremă se poate deduce următorul algoritm pentru determinarea mulțimilor exterior stabile (dominante) ale unui graf oarecare $G = (X, \Gamma)$ dat.

Algoritmul 1.3.2.1.

Determinarea mulțimilor exterior stabile.

(a) Se dă graful $G = (X, \Gamma)$.

Construiește graful bipartit $G' = (X \cup X', \Delta)$.

Fie $T = \Phi$.

(b) Procedura de reducere

Cât timp există $j' \in X'$ cu $|\Delta^{-1}j'| = 1$ execută

$$T = T \cup \Delta^{-1}j';$$

$$G' = G' - (\Delta^{-1}j' \cup \Delta(\Delta^{-1}j'));$$

sfc

(c) Cât timp $X \neq \Phi$ execută

alege $i \in X$;

$T = T \cup \{i\}$;
 $G' = G - \{i\} \cup \Delta i$;
 elimină din X vârfurile i cu $\Delta i = \Phi$;
 execută "Procedura de reducere";
 sf.

Observația 1.3.2.1.

Dacă algoritmul execută instrucțiuni din pasul (c) atunci datorită apelului recursiv cerut de "execuță (b)" pentru fiecare alegere a lui i din X se află câte o mulțime exterior stabilă. Astfel se pot determina toate mulțimile exterior stabile ale grafului dat. Apoi se poate determina numărul de stabilitate externă a grafului și alege mulțimi exterior stabile minime.

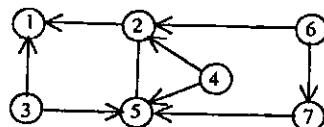


Fig. 1.3.2.2.

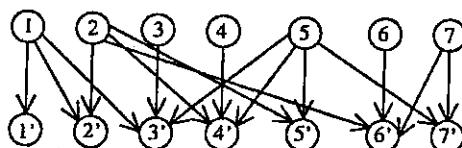


Fig. 1.3.2.3.

Pentru graful din figura 1.3.2.3, graful bipartit asociat lui este cel din figura 1.3.2.2. În urma aplicării algoritmului dat obținem ca mulțimi exterior stabile ale sale pe: $\{1, 2, 7\}$; $\{1, 2, 5\}$; $\{1, 5, 2\}$; $\{1, 5, 7\}$; $\{1, 7, 2\}$ și $\{1, 7, 5\}$.

Mulțimile independente (interior stabilă) și dominante (exterior stabilă) se pot determina și cu ajutorul unor algoritmi de tip *branch and bound* [ToC97]. Această metodă se aplică problemelor *NP*-complete care satisfac următoarele:

- mulțimea soluțiilor admisibile (posibile) S este finită $S = \{s_1, \dots, s_t\}$;
- există o funcție $f: S \rightarrow \mathbb{R}$ cu proprietatea:

dacă $f(a) = \min(\max)\{f(x) | x \in S\}$ atunci a este o soluție optimă.

- există o metodă (procedură) pentru a determina o margine superioară (respectiv inferioară) mică (mare) eventual supremumul (infimumul) pentru orice $A \subset S$, adică $\sup A$ ($\inf A$) este estimabil;
- există un criteriu de a parta (descompune) o mulțime $A \subset S$ în două sau mai multe submulțimi, adică $\exists B$ sau $\exists B_1, \dots, B_s$ astfel încât $A = B \cup C_A B$ respectiv $A = B_1 \cup \dots \cup B_s$, unde $C_A B = A - B$ este complementara lui B în raport cu A .

Având o astfel de procedură se construiește o arborescență etichetată. În eticheta fiecărui vârf se află o submulțime $A \subset S$ într-o formă explicită sau implicită și marginea superioară (inferioară) care estimează pe $\sup A$ ($\inf A$) numită *bornă*. Se va dezvolta arborescența prin a adăuga succesorii vârfului terminal (frunză) k al arborescenței, deci $\Gamma^k = \Phi$, cu bornă cea mai mică (mare). Numărul descendenților ce se vor adăuga lui k corespunde criteriului de partaționare utilizat. Dezvoltarea arborescenței se face până când se obține o soluție optimă. Apoi se continuă pentru a determina alte soluții optime prin dezvoltarea arborescenței doar pentru vârfurile cu bornă "convenabilă". Etapa de inițializare construiește doar rădăcina arborescenței etichetată conform convențiilor problemei pentru submulțimea $A = S$. Deoarece nu prezintă interes decât vârfurile terminale din arborescență se pot păstra (memora) într-o listă doar aceste vârfuri.

Algoritmul 1.3.2.2.

Metoda *branch and bound* pentru determinarea mulțimilor interior stabilă (independente) maxime constă în construirea unei arborescențe care are etichetele vârfurilor de forma (A, B, b) cu semnificația:

- $A \subset X$ este o mulțime interior stabilă și toți descendenții săi din arborescență definesc mulțimi interior stabilă care conțin pe A (deci au cardinalul mai mare sau egal cu al lui A);
- $B \subset X$ are proprietățile $\Gamma A \cup \Gamma^{-1} A \subset B$ și toate mulțimile interior stabilă din vârfurile descendente nu au vârfuri din B ;
- $b = |X - B|$ este o margine superioară a numărului de vârfuri care mai pot fi adăugate la A pentru a obține o mulțime interior stabilă maximă.

Pentru a preciza descendenții vârfului arborescenței etichetat cu (A, B, b) să observăm că $\forall i \in X - B$ mulțimea $A \cup \{i\}$ este interior stabilă deoarece $\Gamma A \cup \Gamma^{-1} A \subset B$. Atunci, ramificarea arborescenței se face astfel:

alege $i \in X - (A \cup B)$ și construiește succesorii etichetați unul cu $(A \cup \{i\}; B \cup \Gamma i \cup \Gamma^{-1} i; |X - A \cup B \cup \Gamma i \cup \Gamma^{-1} i|)$, iar celălalt cu $(A; B \cup \{i\}; b - 1)$.

Adică, un vîrf are două subarborescențe una care definește toate mulțimile interior stabile care conțin pe $A \cup \{i\}$ și ceeaலătă care definește toate submulțimile interioare stabile din $X - \{i\}$ care conțin pe A .

Vâfurile arborescenței cu borna 0 nu se pot dezvolta și după ce s-a obținut un asemenea vîrf se reține în variabila α , de exemplu, cardinalul lui A din eticheta acelui vîrf și se va dezvolta dintre vâfurile terminale etichetate (A, B, b) cu $|A| + b > \alpha$ cel care are $|A| + b$ maxim. De fiecare dată când se obține un vîrf frunză se va actualiza valoarea lui α la cardinalul maxim al mulțimilor interior stabile deja depistate.

Algoritm 1.3.2.3.

Algoritmul dat mai sus pentru determinarea de mulțimi interior stabile poate fi prezentat sub formă unei metode de tip *branch and bound* pentru a determina mulțimi exterior stabile (toate sau numai pe cele maxime). Acest algoritm construiește o arborescență ale cărui vârfuri terminale definesc mulțimi exterior stabile. Dacă se dezvoltă doar anumite vâfururi și atunci se obțin toate mulțimile exterior stabile maxime. Etichetele arborescenței sunt de forma $(G'; T; b)$, unde:

- G' este graful redus (obținut din graful curent căruia i s-a aplicat procedura de reducere din algoritmul de determinare a mulțimilor exterior stabile dat mai sus);
- T este o mulțime de vârfuri ale grafului care este conținută de toate mulțimile exterior stabile definite de descendenții vârfului curent al arborescenței;
- b este $|X'|, X'$ din G' .

Rădăcina arborescenței se etichetează cu $(G'; T; |X'|)$, valori obținute de procedura de reducere aplicată grafului bipartit G' asociat grafului dat G . Deoarece pentru fiecare nod are loc $\forall i \in X$, din graful G^i definit de etichetă, are loc $|\Delta i| \geq 2$, atunci succesorii nodului îi definim astfel:

- alege $i \in X$;
- un sucesor se etichetează cu valorile date de procedura de reducere aplicată după ce modificăm pe T cu $T \cup \{i\}$, pe X cu $X - \{i\}$ și pe X' cu $X' - \Delta i$;
- celălalt succesor se etichetează cu valorile date de procedura de reducere aplicată după ce modificăm pe X cu $X - \{i\}$.

Ramificările se fac până când se obține un vîrf care are în graful G' din etichetă $X' = \Phi$ caz în care și $X = \Phi$ și deci vârful este un vîrf frunză al arborescenței. Mulțimea T din eticheta oricărui vîrf frunză este o mulțime exterior stabilă a grafului dat G . Dacă se dorește să se obțină doar mulțimile exterior stabile minime, atunci se vor dezvolta doar vâfurile pentru care $b + |X'|$ este cea mai mică și b nu depășește cardinalul mulțimilor exterior stabile deja depistate.

1.3.3. Nucleul unui graf

Definiția 1.3.3.1. Fiind dat un graf $G = (X, \Gamma)$, o mulțime $N \subset X$ se numește *nucleu* al grafului G dacă și numai dacă N este interior stabilă și exterior stabilă.

Observația 1.3.3.1.

Dacă N este nucleu al grafului G atunci

- $N \cap \Gamma N = \Phi$ și $N \cup \Gamma^1 N = X$;
- dacă $i \in X$ și $i \in \Gamma^1 i$ atunci $i \notin N$ și dacă $i \in X$ cu $\Gamma^1 i = \Phi$ atunci $i \in N$.

Această observație se poate interpreta astfel:

- orice succesor al unui vîrf aparținând nucleului nu face parte din nucleu și orice succesor al unui vîrf care nu face parte din nucleu aparține nucleului;
- vâfurile care au bucle nu fac parte din nucleu și toate vâfurile care nu au predecesori (deci și cele izolate) fac parte din nucleu.

Exemplul 1.3.3.1.

Pentru graful din figura 1.3.3.1 mulțimea $N = \{3, 5, 7\}$ este un nucleu.

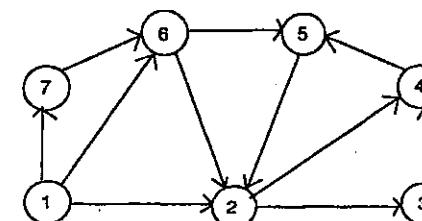


Fig. 1.3.3.1.

Teorema 1.3.3.1. Dacă N este un nucleu al grafului $G = (X, G)$ atunci mulțimea N este o mulțime maximală a familiei \mathcal{S} a mulțimilor interior stabile, adică

$$A \in \mathcal{S} \text{ și } N \subset A \Rightarrow A = N.$$

Demonstrație:

Vom demonstra prin metoda reducerii la absurd. Să presupunem că există $A \in \mathcal{S}, N \subset A$ și $N \neq A$, deci $\exists i \in A - N$.

Cum N este mulțime exterior stabilă înseamnă că $\Gamma_i \cap N = \emptyset$, dar $N \subset A$ deci $\Gamma_i \cap A \neq \emptyset$ ceea ce contrazice relația $A \in \mathcal{S}$. \square

Observația 1.3.3.2.

Reciproca teoremei 1.3.3.1 în general nu are loc. Adică pentru un graf oarecare nu orice mulțime interior stabilă maximală este nucleu. Există grafe care nu au nuclee dar au mulțimi stable interior.

Teorema 1.3.3.2. Într-un graf simetric fără bucle, orice mulțime maximală a familiei \mathcal{S} este nucleu.

Demonstrație:

Fie $N \in \mathcal{S}$ maximală. Vom demonstra că N este mulțime exterior stabilă prin reducere la absurd. Să presupunem că $\exists i \in X - N$ astfel încât $\Gamma_i \cap N = \emptyset$, deci $\forall j \in N, j \notin \Gamma_i$ și cum G este simetric deducem că $i \notin \Gamma_j$. Deoarece $i \notin \Gamma_j$ rezultă că $N \cup \{i\}$ este mulțime interior stabilă ceea ce contrazice maximalitatea lui N . Prin urmare N este nucleu al grafului. \square

Având în vedere că $\mathcal{S} \neq \emptyset$, din propoziția 1.3.3.2. deducem:

Corolar 1.3.3.1. Un graf simetric fără bucle admite cel puțin un nucleu.

Să remarcăm că pentru teoremă și corolar este esențială proprietatea de simetrie a grafului.

Pornind de la relațiile a ale observației 1.3.3.1 se poate construi următorul algoritm pentru determinarea unui nucleu al unui graf fără circuite.

Algoritm 1.3.3.1.

Determinarea unui nucleu într-un graf fără circuite.

(a) fie $N = \emptyset$;

(b) Cât timp există $i \in X - N$ cu $\Gamma^I i = \emptyset$ execută

fie $N = N \cup \{i\}$;
fie $G = G - (i) - \Gamma_i$;
sfc.

Exemplu 1.3.3.2.

Aplicând algoritmul 1.3.3.1. pentru graful din figura 1.3.3.2 se obține nucleul $N = \{1, 5, 8, 6\}$.

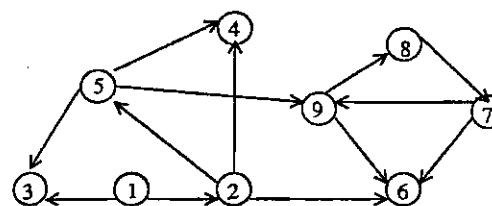


Fig.1.3.3.2.

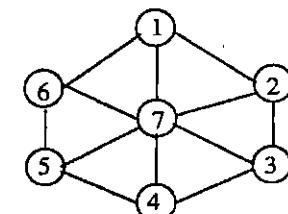


Fig.1.3.3.3.

Observația 1.3.3.3.

Să remarcăm că există grafe care au mai multe nuclee. De exemplu pentru graful din figura 1.3.3.3 mulțimile de vârfuri $\{1, 3, 5\}$, $\{2, 4, 6\}$ și $\{7\}$ sunt nuclee, în timp ce graful din figura 1.3.1.1, care nu este simetric, nu are nici un nucleu.

1.3.4. Număr cromatic

Problema colorării grafelor este o problemă foarte veche. Unele dintre aceste probleme privesc colorarea muchiilor, iar altele colorarea vârfurilor. Aceste probleme cer să se determine numărul de culori cu care se pot colora unele elemente ale grafului astfel încât colorarea să îndeplinească niște condiții. Algoritmii care rezolvă aceste probleme au o complexitate mare.

Definiția 1.3.4.1. Numărul minim de culori cu care se pot colora muchiile grafului G astfel încât oricare două muchii adiacente să fie colorate cu culori diferite se numește *indice cromatic*.

Indicele cromatic se notează de obicei cu $q(G)$.

Teorema 1.2.4.1. Teorema lui Vizing

Pentru orice graf G are loc:

$$\chi(G) \leq \max\{d(i) \mid i \in V\} + 1.$$

Definiția 1.3.4.2. Un graf G este k -colorabil dacă vârfurile sale se pot colora cu cel mult k culori astfel încât oricare două vârfuri adiacente să fie colorate diferit.

Definiția 1.3.4.3. Număr cromatic al grafului G este cel mai mic număr k pentru care graful este k -colorabil.

Numărul cromatic se notează cu $\chi(G)$.

Propoziția 1.3.4.1. Număr cromatic al grafului $G = (X, U)$ este cel mai mic număr natural p cu proprietatea că există o funcție surjectivă $f: X \rightarrow \{1, 2, \dots, p\}$ cu proprietatea $\forall i, j \in X$ cu $(i, j) \in U$ are loc $f(i) \neq f(j)$.

Mulțimea $\{1, 2, \dots, p\}$, de mai sus, o numim mulțimea culorilor, iar funcția f se numește *colorare* a vârfurilor lui G .

Problema 1.3.4.1. Problema colorării grafului

Fiind dat un graf $G = (X, U)$ se cere să se determine:

- numărul cromatic al lui G ;
- o colorare a grafului G .

Teorema 1.3.4.2. Dacă $f: X \rightarrow C$ este o colorare a grafului G atunci:

- $\forall a, b \in C$ cu $a \neq b$ are loc $f^{-1}(a) \cap f^{-1}(b) = \emptyset$;
- $\forall a \in C$ are loc $f^{-1}(a)$ este o mulțime interior stabilă a lui G ;
- $X = \bigcup_{a \in \text{Im } f} f^{-1}(a)$.

Pentru grafele complete K_n colorările sunt funcții bijective și deci $\chi(K_n) = n$. Prin urmare vârfurile oricărei cliici trebuie colorate cu culori diferite sau $\chi(H) = |A|$ pentru orice clică $H = (A, V)$.

Teorema 1.3.4.3. Teorema lui König

Un graf este bicromatic dacă și numai dacă graful nu conține cicluri de lungime impară.

Demonstrație:

Necesitatea (\Rightarrow)

Dacă graful este bicromatic înseamnă că orice ciclu are vârfurile adiacente colorate diferit. Deci numărul vârfurilor care compun ciclurile este par. Lungimea ciclurilor, care fiind numărul de muchii ce compun ciclul, este egală cu numărul vârfurilor. Deci orice ciclu are lungime pară.

Suficiența (\Leftarrow)

Vom colora vârfurile grafului cu două culori folosind următorul procedeu. Se alege un vârf și se colorează cu o culoare. Se colorează cu ceea cealaltă culoare toate vârfurile necolorate și adiacente unui vârf colorat. Se repetă cea de a doua acțiune cât se poate. Se reia procedeul de la prima acțiune până când vor fi colorate toate vârfurile grafului. Dacă două vârfuri adiacente ar fi colorate cu aceeași culoare înseamnă că graful admite cicluri de lungime impară ceea ce ar contrazice ipoteza. Deci graful este bicromatic. \square

Observația 1.3.4.1.

- Dacă graful din teorema 1.3.4.3 nu este graful vid (X, Φ) atunci numărul său cromatic este 2.
- Un graf este bicromatic dacă și numai dacă este bipartit.

Graful care îndeplinește una dintre condițiile afirmației b are proprietatea că mulțimea vârfurilor sale se împarte în două părți disjuncte cu proprietatea că în fiecare parte orice două vârfuri nu sunt adiacente. Adică graful este bipartit și deci colorând vârfurile fiecărei părți cu câte o culoare graful este în același timp și bicromatic.

Teorema 1.3.4.4. Fie $G = (X, U)$ un graf neorientat de ordin n și dimensiune m , iar \bar{G} graful complementar al lui G , atunci au loc relațiile:

- $\chi(G) + \chi(\bar{G}) \leq n + 1$;
- $\chi(G)\chi(\bar{G}) \leq (n + 1)^2$;
- $\chi(G) \geq n^2/(n^2 - 2m)$.

Demonstrație:

a. Utilizăm metoda inducției matematice. Pentru $n = 1$ și pentru $n = 2$ cunoștem K_n se verifică imediat că $\chi(G) + \chi(\overline{G}) = n + 1$.

Fie $p \in \mathbb{N}$, să presupunem că relația este adeverată pentru orice graf cu $p - 1$ vârfuri și să considerăm un graf $G = (X, U)$ de ordin p colorat cu $\chi(G)$ culori. Fie $i \in X$ pentru că i poate fi colorat cu o culoare pe care o au sau nu alte vârfuri din G deducem

$$\chi(G) \leq \chi(G - \{i\}) + 1 \quad \text{și} \quad \chi(\overline{G}) \leq \chi(\overline{G} - \{i\}) + 1.$$

Dacă cel puțin una dintre inegalități este strictă, pentru că inegalitățile sunt cu numere naturale și conform presupunerii făcute $\chi(G - \{i\}) + \chi(\overline{G} - \{i\}) \leq p$, însumându-le obținem

$$\chi(G) + \chi(\overline{G}) < p + 2 \Rightarrow \chi(G) + \chi(\overline{G}) \leq p + 1.$$

Dacă ambele egalități sunt verificate cu egalitate, rezultă că vârful i are în G și \overline{G} vârfuri adiacente colorate cu fiecare culoare din cele $\chi(G - \{i\})$ respectiv $\chi(\overline{G} - \{i\})$ culori. De aici rezultă $\chi(G - \{i\}) + \chi(\overline{G} - \{i\}) \leq g_G(i) + g_{\overline{G}}(i)$, pe de altă parte $g_G(i) + g_{\overline{G}}(i) \leq p - 1$ și deci

$$\begin{aligned} \chi(G) + \chi(\overline{G}) &= \chi(G - \{i\}) + \chi(\overline{G} - \{i\}) + 2 \leq \\ &\leq g_G(i) + g_{\overline{G}}(i) + 2 \leq p - 1 + 2 = p + 1. \end{aligned}$$

Conform inducției matematice deducem că pentru orice graf G de ordin n și complementar său \overline{G} are loc

$$\chi(G) + \chi(\overline{G}) \leq n + 1$$

b. Folosind relația de la a și relația $4\chi(G)\chi(\overline{G}) \leq (\chi(G) + \chi(\overline{G}))^2$ se obține relația de demonstrat.

c. Pentru a simplifica notăm $k = \chi(G)$, n_i = numărul vârfurilor lui G colorate cu culoarea i , unde $i \in \{1, 2, \dots, k\}$, V_0 și V_1 numărul valorilor egale cu 0 respectiv 1 din matricea de adiacență a grafului G . Deoarece $V_1 = 2m$ între oricare două vârfuri colorate cu aceeași culoare nu există muchie, înseamnă că în matricea de adiacență corespunzător culorii i există n_i^2 valori egale cu 0. Deci

$$V_0 \geq n_1^2 + n_2^2 + \dots + n_k^2 \geq \frac{(n_1 + n_2 + \dots + n_k)^2}{k} = \frac{n^2}{k};$$

și deci

$$n^2 = V_1 + V_0 \geq 2m + \frac{n^2}{k} \Rightarrow k = \chi(G) \geq \frac{n^2}{n^2 - 2m}. \quad \square$$

Problema 1.3.4.2.

Într-un teritoriu există un sistem de relee de retransmitere a mesajelor (radio, tv, etc.). Din cauză că se bruiază reciproc releele care sunt la o distanță mai mică decât ce trebuie să emite pe canale diferite. Se cere să se determine numărul minim de canale necesar.

Pentru rezolvare considerăm graful ce are ca vârfuri releele și între două relee există o muchie dacă și numai dacă distanța dintre relee este mai mică decât d . Determinarea numărului minim de canale revine la a determina numărul cromatic al grafului, deoarece asociem unei culori un canal.

Propoziția 1.3.4.2. *Graful G are cel puțin o clică cu $\chi(G)$ vârfuri.*

Demonstrația se poate face prin reducere la absurd. Deoarece numărul maxim de vârfuri dintr-o clică a grafului G este $\max\{g(i) \mid i \in X\} + 1$ se deduce.

Corolarul 1.3.4.1. *Are loc $\chi(G) \leq \max\{g(i) \mid i \in X\} + 1$.*

Acest rezultat se poate obține și pe baza teoremei lui Vizing și a faptului că $\chi(G) = q(L(G))$. Folosind acest rezultat pentru G și \overline{G} prin însumare se obține delimitarea din teorema 1.3.4.4. a.

Pentru un graf G și un număr natural x se notează cu $C(G, x)$ numărul de colorări distincte ale grafului G cu x culori. Polinomul a cărui valoare în x este $C(G, x)$ se numește *polinom cromatic*, prin abuz de limbaj chiar pe $C(G, x)$ îl numim polinom cromatic.

Propoziția 1.3.4.3.

- Dacă $x < \chi(G)$, atunci $C(G, x) = 0$.
- $C(K_n, x) = x(x - 1) \dots (x - n + 1)$ sau numărul colorărilor grafului complet este egal cu numărul funcțiilor definite pe X cu valori în C care sunt injective.
- Dacă $G = (X, \emptyset)$, atunci $C(G, x) = x^n$ sau numărul colorărilor grafului vid este egal cu numărul funcțiilor definite pe X cu valori în C .

d. Dacă $G = T_n$ este un arbore cu n vârfuri, atunci $C(T_n, x) = x(x - 1)^n$.

Demonstrația se face fie prin inducție fie cu algoritm:

Se coloarează un vârf oarecare cu oricare dintre cele x culori apoi cât timp există vârf colorat cu vârfuri adiacente necolorate se coloarează un vârf adiacent celui colorat cu oricare dintre cele $x - 1$ culori diferite de culoarea vârfului deja colorat. Graful neavând cicluri nu se vor colora două vârfuri adiacente cu aceeași culoare.

Teorema 1.3.4.5. Dacă $G = (X, U)$ este un graf, $p, q \in X$ două vârfuri neadiacente și εG este morfismul prin care vâfurile p și q se identifică, atunci

$$C(G, x) = C(G + pq, x) + C(\varepsilon G, x).$$

Demonstrație:

Pentru că vâfurile p și q nu sunt adiacente rezută că numărul colorărilor grafului G este egal cu numărul colorărilor în care p și q sunt colorate diferit adunat cu numărul colorărilor lui G în care p și q sunt colorate cu aceeași culoare. Cele două numere sunt chiar $C(G + pq, x)$ respectiv $C(\varepsilon G, x)$. \square

Relația din teorema 1.3.4.5. este o relație de recurență care exprimă numărul colorărilor unui graf ca fiind suma numerelor colorărilor unui graf cu o muchie în plus respectiv a unui graf cu un vârf în minus față de graful dat. Aplicând recursiv aceeași relație pentru termenii obținuți se poate ca expresia după un număr finit de pași să conțină numai termeni de forma $C(K_p, x)$ pentru diferite valori ale lui $p \leq n$. Apoi acestea se înlocuiesc cu expresiile date de formula b) din propoziția 1.3.4.3.

Corolar 1.3.4.2.

- a. Coeficientul lui x^n este 1, iar al lui x^{n-1} este egal cu m , numărul muchiilor lui G .
- b. Are loc $C(G + pq, x) = C(\varepsilon G, x) - C(G, x)$.
- c. $C(C_n, x) = (x-1)^n + (-1)^n(x-1)$, unde C_n este ciclul elementar cu n vârfuri și n muchii (graf conex și cu toate vâfurile de grad 2).

1.4. Conexitate

1.4.1. Componente conexe

Definițiile grafelor tare conexe și a celor conexe au fost date în primul paragraf al acestui capitol. Considerând un graf $G = (X, U)$ neorientat definim o relație $R \subset X \times X$ astfel

$$i R j \Leftrightarrow i = j \text{ sau există în } G \text{ un lanț de la } i \text{ la } j.$$

Deoarece un lanț de la i la j induce un lanț de la j la i și pentru că două lanțuri cu o extremitate comună se pot compune, relația definită este o relație de echivalență (reflexivă, simetrică și tranzitivă). Această relație de echivalență determină o împărțire a mulțimii vâfurilor grafului G în clase de echivalență. Fie X_1, X_2, \dots, X_p clasele de echivalență.

Definiția 1.4.1.1. Orice clasă de echivalență X_k pentru $k \in \{1, 2, \dots, p\}$ sau graful $G(X_k)$ se numește **componentă conexă** a grafului dat G .

Cu p notăm numărul componentelor conexe.

Observația 1.4.1.1.

- a. Orice graf are cel puțin o componentă conexă, adică $p \geq 1$.
- b. Un graf care are o singură componentă conexă este graf conex.

În continuare vom prezenta un algoritm pentru determinarea componentelor conexe ale unui graf, care se bazează pe *algoritmul lui Tarjan* [Tar72]. Ideea acestui algoritm este de a marca vâfurile grafului cu numere (rol îndeplinit în algoritm de valorile variabilei k). Vâfurile marcate cu același număr fac parte din aceeași componentă conexă. Algoritmul folosește următoarele notății:

- $d(i) = |\Gamma_i|$, pentru $i \in X$;
- $\Gamma_i(j)$ este al j -lea element al mulțimii Γ_i , cu $i \in X$, deci elementele mulțimilor Γ_i se consideră ordonate;
- E este numărul vâfurilor clasificate, la un moment dat;
- $n(i)$ conține indicele ultimului vârf explorat din mulțimea Γ_i considerată ordonată, la un moment dat;

iar în finalul algoritmului

- $c(i) = k$ înseamnă că vârful i face parte din componenta X_k ;

- $p(j) = i$ înseamnă că vârful i a fost introdus în aceeași componentă conexă cu vârful j datorită faptului $j \in \Gamma_i$.

Algoritmul 1.4.1.1.

Determinarea componentelor conexe.

(a) etapa de initializare

```

pentru  $i=1, n$  execută
   $p(i)=0; d(i)=|\Gamma_i|;$ 
   $c(i)=0; n(i)=0;$ 
sfp;
 $k=0; et=0;$ 
 $E=0; s=1;$ 
```

(b) iterarea de bază

Cât timp $E \neq n$ execută

Dacă $c(s)=0$ atunci

```

 $k=k+1;$ 
 $c(s)=k; p(s)=s;$ 
```

$i=s; et=et+1;$

$o(i)=et; E=E+j;$

Cât timp $(n(i) \neq d(i)$ sau $i=s$) execută

Dacă $n(i)=d(i)$

atunci $i=p(i)$

altfel $n(i)=n(i)+1; j=\Gamma_i(n(i));$

Dacă $p(j)=0$ atunci $p(j)=i; l=j;$

$c(i)=k; et=et+1;$

$o(i)=et; E=E+1;$

sfd;

sfc;

sfd;

$s=s+1;$

sfc.

Observația 1.4.1.2.

Numărul componentelor conexe este valoarea lui k cu care se termină algoritmul.

Exemplul 1.4.1.1.

Considerăm graful din figura 1.4.1.1, în care:

$$\begin{aligned}\Gamma_1 &= \{4\}, \Gamma_2 = \{3; 4\}, \\ \Gamma_3 &= \{2, 4\}, \Gamma_4 = \{1, 2, 3, 5\}.\end{aligned}$$

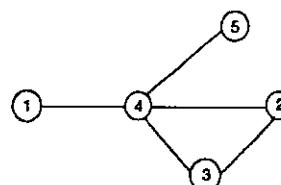


Fig. 1.4.1.1.

$$\Gamma_5 = \{4\}.$$

Execuția algoritmului 1.4.1.1 pas cu pas face ca variabilele folosite să primească următoarele valori:

- (a) $p(1)=p(2)=p(3)=p(4)=p(5)=0;$
 $d(1)=1; d(2)=2; d(3)=2; d(4)=4; d(5)=1;$
 $c(1)=c(2)=c(3)=c(4)=c(5)=0;$
 $n(1)=n(2)=n(3)=n(4)=n(5)=0;$
 $k=0; E=0; l=1; et=0;$
- (b) cum $E \neq n$ $c(1)=0$ atunci
 $k=1; c(1)=1; p(1)=1; i=1; E=1; et=1; o(1)=1;$
cum $n(1) \neq d(1)$ atunci $n(1)=1; j=\Gamma_1(1)=4; \text{cum } p(4)=0$
 $p(4)=1; i=4; c(4)=1; et=2; o(4)=2; E=2;$
cum $n(4) \neq d(4)$ atunci $n(4)=1; j=\Gamma_4(1)=1; \text{cum } p(1) \neq 0$
 $\text{cum } n(4) \neq d(4) \text{ atunci } n(4)=2; j=\Gamma_4(2)=2; \text{cum } p(2)=0$
 $p(2)=4; i=2; c(2)=1; et=3; o(2)=3; E=3;$
cum $n(2) \neq d(2)$ atunci $n(2)=1; j=\Gamma_2(1)=3; \text{cum } p(3)=0$
 $p(3)=2; i=3; c(3)=1; et=4; o(3)=4; E=4;$
cum $n(3) \neq d(3)$ atunci $n(3)=1; j=\Gamma_3(1)=2; \text{cum } p(2) \neq 0$
cum $n(3) \neq d(3)$ atunci $n(3)=2; j=\Gamma_3(2)=4; \text{cum } p(4) \neq 0$
cum $i \neq 1$ și $n(3)=d(3)$ atunci $i=p(3)=2;$
cum $n(2) \neq d(2)$ atunci $n(2)=2; j=\Gamma_3(2)=4; \text{cum } p(4) \neq 0$
cum $i \neq 1$ și $n(2)=d(2)$ atunci $i=p(2)=4;$
cum $n(4) \neq d(4)$ atunci $n(4)=3; j=\Gamma_4(3)=3; \text{cum } p(3) \neq 0$
cum $n(4) \neq d(4)$ atunci $n(4)=4; j=\Gamma_4(4)=5; \text{cum } p(5)=0$
 $p(5)=4; i=5; c(5)=1; et=5; o(5)=5; E=5;$
cum $n(5) \neq d(5)$ atunci $n(5)=1; j=\Gamma_5(1)=4; \text{cum } p(4) \neq 0$
cum $i \neq 1$ și $n(5)=d(5)$ atunci $i=p(5)=4;$
cum $i \neq 1$ și $n(4)=d(4)$ atunci $i=p(4)=1;$
cum $i=1$ și $n(1)=d(1);$
 $s=2;$

- (b) cum $E=n$ atunci STOP.

Deoarece $n=5$ și $c(1)=c(2)=c(3)=c(4)=c(5)=1$ înseamnă că toate vîrfurile grafului fac parte din aceeași componentă conexă și deci graful este conex.

Observația 1.4.1.3.

$\sigma(i) = t$ înseamnă că vârful i a fost inspectat la pasul t . Folosind valorile tablourilor σ și p se poate obține, prin orientarea unor muchii, o arborescență a unui graf conex.

Prin urmare $\sigma(1) = 1$, $\sigma(2) = 3$, $\sigma(3) = 4$, $\sigma(4) = 2$ și $\sigma(5) = 5$ înseamnă că vârfurile grafului au fost inspectate în ordinea 1, 4, 2, 3, 5. Deoarece $p(4) = 1$, $p(2) = 4$, $p(3) = 2$ și $p(5) = 4$ înseamnă că în graful dat există muchiile (4,1), (2,4) (3,2), (5,4) pe care dacă le orientăm ținând cont de ordinea de inspecțare se obține o arborescență (figura 1.4.1.2).

Un alt algoritm pentru determinarea componentelor conexe ale unui graf se poate concepe prin utilizarea variabilelor de tip mulțime. Componentele conexe sunt mulțimi (clase) ce au ca tip de bază subdomeniul ce conține vârfurile grafului. La citirea fiecărei muchii se stabilește dacă:

0. nici un vârf al muchiei nu este în vreo componentă conexă existentă deja;
1. un singur vârf sau ambele vârfuri se află într-o componentă conexă existentă deja;
2. cele două vârfuri ale muchiei sunt în componente conexe diferite (formate până la acel moment). Apoi depinzând de caz se va:
 0. construi o nouă componentă conexă ce va conține doar extremitățile muchiei citite (numărul componentelor conexe crește cu 1);
 1. reuni componentă conexă cu mulțimea formată din cele două vârfuri ale muchiei (numărul componentelor conexe rămâne același);
 2. forma o componentă conexă care să fie reuniunea celor două componente conexe (numărul componentelor conexe se va micșora cu 1).

Prezentăm în continuare o descriere în Turbo Pascal a algoritmului schițat mai sus.

Program Componente_Conexe;

```
Var n,nrc,i,j,k,x,y : Word;
    cl : Array[1..40] of Set of 1..80;
    c : Char;
    a : Array[1..2] of 1..40;

Begin
    nrc:=0; Write ('Dati nr. de virfuri '); Readln(n);
```

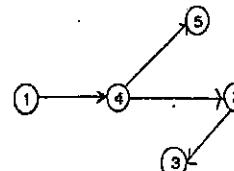


Fig. 1.4.1.2.

```
Writeln('Dati muchiile (virf,spatiu,virf)' +
        ' in final (0,spatiu,orice)');
Read(x);
While x>0 Do
begin
    Readln(c,y);
    k:=0;
    For i:=1 to nrc Do If (x IN cl[i]) or (y IN cl[i]) then
        begin k:=k+1; a[k]:=i; end;
    Case k of
        0: begin nrc:=nrc+1 cl[nrc]:=[x,y]; end;
        1: cl[a[1]]:=cl[a[1]]+[x,y];
        2: begin cl[a[1]]:=cl[a[1]]+cl[a[2]];
            If a[2]<>nrc then cl[a[2]]:=cl[nrc];
            nrc:=nrc-1;
        end;
    end;
    Read(x);
end;
For i:=1 to nrc Do
begin
    Writeln('Componenta conexă cu numărul ',i:2,
            ' conține virfurile:');
    For j:=1 to n Do If j IN cl[i] then write(j:3);
    Writeln;
end;
End.
```

Teorema 1.4.1.1. Fie $G = (X, U)$ un graf neorientat.

G este graf conex \Leftrightarrow

$\forall A \subset X$ cu $\emptyset \neq A \neq X$, are loc $\omega(A) \neq \emptyset$.

Demonstrație:

" \Rightarrow " Dispunem de un graf $G = (X, U)$ conex. Fie $A \subset X$ cu $\emptyset \neq A \neq X$ alegem $p \in A$ și $q \in X - A$. Graful G fiind conex între p și q există cel puțin un lanț. Parcurgând acest lanț de la $p \in A$ spre $q \in X - A$ se va depista o muchie care are o extremitate în A și cealaltă extremitate în $X - A$. Această muchie face parte din $\omega(A)$ și deci $\omega(A) \neq \emptyset$.

" \Leftarrow ". Să demonstrăm că dacă pentru graful $G = (X, U)$ are loc $\forall A \subset X$ cu $\emptyset \neq A \neq X$ și $\omega(A) \neq \emptyset$, atunci G este conex. Demonstrăm prin reducere la absurd. Presupunem că G nu este conex deci are cel puțin două componente conexe. Alegem o componentă conexă ca fiind A și deci $\omega(A) = \emptyset$ ceea ce este în contradicție cu ipoteza. Deci, G este graf conex. \square

Teorema 1.4.1.2. Să se demonstreze că dacă în graful $G = (X, U)$, $\forall i \in X$ $g(i) \geq (n - 1)/2$ atunci G este conex.

Demonstrația 1:

Demonstrăm că G este conex prin reducere la absurd. Să presupunem că G este un graf cu n vârfuri și nu este conex. Estimăm gradele vâfurilor acestor grafe. Fiindcă G are cel puțin două componente conexe, atunci dintre acestea există cel puțin una care are cel mult $n/2$ vârfuri. Într-o asemenea componentă gradul maxim al vâfurilor este $n/2 - 1 < (n - 1)/2$, ceea ce setează o contradicție cu ipoteza. Deci, orice asemenea graf este conex. \square

Dăm o altă demonstrație tot cu metoda reducerii la absurd.

Demonstrația 2:

Dacă G nu ar fi conex rezultă că există două vârfuri, să le notăm p și q între care nu există nici un lanț. Dacă $g(p) \geq (n - 1)/2$ (altfel ar fi în contradicție cu ipoteza) și pentru că vârful q nu este adjacent cu nici unul dintre cele $g(p)$ vârfuri adiacente cu p , cu vârful p și nici cu el însuși deducem că

$$g(q) \leq n - g(p) - 2 \leq n - (n - 1)/2 - 2 = (n - 3)/2 < (n - 1)/2$$

ceea ce este în contradicție cu ipoteza. \square

Teorema 1.4.1.3. *Orice graf de ordin n cu mai mult de $(n-1)(n-2)/2$ muchii este conex.*

Demonstrație:

Demonstrăm că numărul maxim de muchii pentru grafele care nu sunt conexe este $(n - 1)(n - 2)/2$ ceea ce demonstrează teorema. Demonstrația are două etape mai întâi se arată că numărul maxim de muchii pentru grafele cu un număr dat (constant) de vârfuri îl au grafele cu o singură componentă conexă (complete). Aceasta rezultă din faptul că dacă numărul de vârfuri este a iar graful ar avea două componente conexe cu $1 \leq b < a$ și $a - b$ vârfuri atunci

$$a(a - 1)/2 > a(a - 1)/2 - b(a - b) = b(b - 1)/2 + (a - b)(a - b - 1)/2.$$

De aici se deduce că, dintre grafele care nu sunt conexe număr maxim de muchii îl au grafele cu două componente conexe.

Cealaltă etapă constă în a demonstra că un graf cu număr dat a de vârfuri și cu două componente conexe are număr maxim de muchii dacă o componentă are un singur vârf. Rezultat ce se obține din inegalitatea

$$\begin{aligned} b(b - 1)/2 + (a - b)(a - b - 1)/2 &= \\ &= (a - 1)(a - 2)/2 - b(a - b) - (a - 1) \leq (a - 1)(a - 2)/2, \end{aligned}$$

pentru $b \geq 1$ și $b < a$. \square

Din demonstrația făcută se deduce că dintre grafele care au n vârfuri și nu sunt conexe numărul maxim de muchii, care este $(n - 1)(n - 2)/2$, îl are graful cu două componente conexe și una dintre acestea este un vârf izolat.

Teorema 1.4.1.4. *Pentru orice graf neorientat $G = (X, U)$, cu $|X| = n > 1$ are loc:*

$$G \text{ este eulerian și fără vârfuri izolate} \Leftrightarrow$$

$$G \text{ este conex și } \forall i \in X, g(i) \text{ este par.}$$

Demonstrație:

“ \Rightarrow ” Dacă graful are un ciclu eulerian (simplu și utilizează toate muchiile), atunci parcurgând complet acest ciclu începând cu o muchie (vârf) oarecare. La fiecare intrare într-un vârf se și ieșe din acel vârf și de fiecare se parcurge o muchie nefolosită. Mai întâi să deducem de aici că graful, neavând vârfuri izolate, este conex. Apoi, deoarece se parcurg toate muchiile, înseamnă că gradul fiecărui vârf este par, căci el este suma numărului de intrări în vârf cu numărul de ieșiri din acel vârf.

“ \Leftarrow ”. Graful fiind conex nu are vârfuri izolate (fiecare vârf izolat ar fi o componentă conexă).

Vom da un algoritm de determinare a unui ciclu eulerian pentru un asemenea graf. Pentru corectitudinea algoritmului vom utiliza teorema 1.4.1.1.

1. Alegem (ne plasăm în) un vârf oarecare a .
2. Fiindcă vârful în care s-a ajuns (ales) are muchii incidente neparcuse (neselectate), având gradul par, se alege (parcurge) o asemenea muchie pentru ieșirea din vârf deja.
3. Repetă pasul (2) până când se ajunge în vârful de start și toate muchiile incidente lui au fost utilizare (parcurse). Deci s-a depistat (parcurs) un ciclu simplu.
4. Dacă au fost parcurse toate muchiile atunci ciclul depistat este un ciclu eulerian.
5. Dacă nu au fost parcurse toate muchiile, atunci fie A mulțimea vâfurilor care au muchii incidente lor și neselectate (neparcuse). Dintre vâfurile lui A cel puțin un vârf a fost utilizat (vizitat, parcurs). Altfel, $\omega(A) = \emptyset$, G este conex și ar contrazice teorema 1.4.1.1. Alegem b un asemenea vârf (din alegerea făcută deducem $a \neq b$). Se repetă pașii (2) și (3). Fiindcă s-au depistat două cicluri simple din

acestea vom construi un singur ciclu simplu cu care va continua algoritmul. Presupunând că succesiunile de vârfuri ale celor două cicluri sunt $a, \dots, x, b, y, \dots, a$ și b, p, \dots, q, b atunci ciclu depistat până la acest moment este $a, \dots, x, b, p, \dots, q, b, y, \dots, a$.

6. Reia cu (4) sau goto (4).

Pașii (4) - (6) pot fi înlocuiți de o frază cât timp (while). \square

Teorema 1.4.1.5. Pentru orice graf orientat $G = (X, U)$, cu $|X| = n > 1$ are loc:

$$G \text{ are un circuit eulerian și nu are izolate} \Leftrightarrow$$

$$G \text{ este tare conex și } \forall i \in X, g^+(i) = g^-(i).$$

Demonstrație:

Demonstrația este similară cu cea a teoremei precedente numai că alegerea elementelor din U se face ținând cont de orientarea arcelor. \square

Problema 1.4.1.1.

Fie X o mulțime finită de numere naturale. Definim muchiile astfel $(p, q) \in U \Leftrightarrow p$ și q sunt numere prime între ele. Să se verifice dacă următoarele grafe sunt sau nu conexe:

- a. $X = \{1, 2, \dots, n\}$, pentru $n > 1$;
- b. $X = \{k, k+1, \dots, k+n\}$, pentru k și $n > 1$.

Definiția 1.4.1.2. Fie $G = (X, U)$ un graf conex. Un vârf $i \in X$ se numește punct (vârf) de articulație dacă graful $G - \{i\}$ nu este conex.

O muchie $u \in U$ se numește isticmă dacă $G - \{u\}$ nu este conex. Adică $\{u\}$ este un cociclu al grafului G .

Teorema 1.4.1.6. Pentru un graf $G = (X, U)$ conex are loc:

un vârf $i \in X$ este punct de articulație al lui $G \Leftrightarrow$

$\exists p, q \in X$ astfel încât orice lanț între p și q îl confine (trece prin) pe i .

Demonstrație:

“ \Rightarrow ” Fie i un punct de articulație a lui G . Rezultă că graful $G - i = (X - \{i\}, U - \omega(i))$ nu mai este conex. Fie A o componentă conexă a acestuia. Pentru că $\emptyset \neq A \neq$

X , alegem vârfurile p din A și q din $X - A$. Atunci cum între p și q în $G - i$ nu există lanțuri deducem că orice lanț între p și q din G are muchii din $\omega(i)$ și deci trece prin vârful i .

“ \Leftarrow ”. Fie i un vârf oarecare al grafului cu proprietatea că există vârfurile p și q astfel încât orice lanț între p și q îl conține pe i . Deci, fiecare din aceste lanțuri conțin cel puțin o muchie din $\omega(i)$. Prin urmare, în graful $G - i$ nu există lanț între p și q , adică p și q sunt în componente conexe diferite. Așadar, $G - i$ nu este conex sau i este un punct de articulație. \square

Mulțime de articulație este o mulțime $A \subset X$, de vârfuri, cu proprietatea că $G - A$ nu este conex.

G este graf k -conex, cu $1 \leq k < |X|$, dacă k este mai mic decât cardinalul minim al mulțimilor de articulație ale grafului G .

Altfel spus G este k -conex dacă $G - A$ este conex pentru orice $A \subset X$ cu $|A| < k$.

Algoritmul 1.4.1.1 poate fi transformat pentru a determina punctele de articulație ale unui graf. În continuare prezentăm aceste transformări [Tar72]. Pentru fiecare vârf i , notăm cu $D(i)$ mulțimea descendenților săi din arborescența definită de algoritmul 1.4.1.1. Fiecărui vârf $j \in D(i)$ îi asociem numărul

$$l(j) = \min\{o(k) \mid k \in \Gamma_j\}$$

și indicele

$$m(i) = \min\{l(j) \mid j \in D(i)\}.$$

Vârful i pentru care $m(i) = o(i)$ este un punct de articulație.

Algoritm 1.4.1.2.

Determinarea punctelor de articulație pornind de la vârful $s = 1$.

(a) etapa de inițializări

```

pentru i=1,n execută
    p(i)=0; d(i)=|Γi|;
    n(i)=0; m(i)=+∞
stop;
et=1; s=1; i=s;
o(s)=1; p(s)=s;
```

(b) iterația de bază

```

Cât timp (n(i)≠d(i) sau i≠s) execută
    Dacă n(i)=d(i)
        atunci
```

$Q = m(i);$
 $i = p(i); \quad m(i) = Q;$
 Dacă $m(i) = o(i)$ atunci
 Tipărește i , "este punct de articulație";
 sfd;
 altfel
 $n(i) = n(i) + 1; \quad j = \Gamma(n(i));$
 Dacă $p(j) = 0$ atunci
 $m(i) = \min(m(i), o(i));$
 $p(j) = i; \quad i = j; \quad et = et + 1; \quad o(i) = et;$
 altfel
 $r = p(j); \quad m(i) = \min(m(i), o(r));$
 sfd;
 sf.

Într-un mod analog se definiște componentele tare conexe. Adică definișim relația R astfel

$$i R j \Leftrightarrow i = j \text{ sau în } G \text{ există drum de la } i \text{ la } j \text{ și de la } j \text{ la } i.$$

Cu algoritmul 1.4.1.2 aplicat grafelor orientate se pot determina componentele tare conexe. Vârful i pentru care $m(i) = o(i)$ deconectează graful în cel puțin două componente tare conexe, una dintre acestea corespunde mulțimii de vârfuri care au fost inspectate după ultima trecere prin vârful i .

Observația 1.4.1.4.

Componenta tare conexă C_i care conține vârful i se poate determina și folosind relația:

$$C_i = \hat{\Gamma}^i \cap \hat{\Gamma}^{-1}_i.$$

1.4.2. Număr ciclomatic și număr cocyclomatic

Fie $G = (X, U)$ un graf orientat de ordin n , de dimensiune m și cu p componente conexe.

Definiția 1.4.2.1. Numărul $m - n + p$ se numește **număr ciclomatic** și se notează cu $v(G)$.

Definiția 1.4.2.2. Numărul $n - p$ se numește **număr cocyclomatic** și se notează cu $\rho(G)$.

Aceste numere, așa cum se va vedea mai departe, sunt dimensiunile unor spații vectoriale definite cu ajutorul ciclurilor respectiv cociclurilor unui graf oarecare G .

Ordonând mulțimea arcelor grafului orientat $G = (X, U)$, adică $U = \{1, 2, \dots, m\}$ fiecarui ciclu μ îi atașăm un vector $\vec{\mu} = (\mu_1, \mu_2, \dots, \mu_m)$ din spațiu \mathbb{R}^m . Valorile componentelor vectorilor se stabilesc astfel:

Dacă d_k reprezintă numărul de treceri ale ciclului m prin arcul k în sensul arcului, iar r_k este numărul de treceri în sens contrar, atunci $\mu_k = d_k - r_k$.

Dacă ciclul este simplu sau elementar atunci componentele vectorului atașat lui sunt 0, 1 sau -1. Într-un mod analog se atașează vectori și cociclurilor grafului.

Deoarece în precizarea unui ciclu nu are importanță vârful cu care se începe și se termină succesiunea de vârfuri (sau arce) ale ciclului operația de compunere a ciclurilor se definește altfel decât cea de compunere a drumurilor și o vom numi sumă a ciclurilor.

Mai întâi vom defini în mulțimea ciclurilor unui graf o operație externă numită **produs** cu numere întregi. Fieind dat un ciclu μ și un număr întreg a prin $a\mu$ vom înțelege ciclul obținut prin parcursarea lui μ de $|a|$ ori în sensul lui μ dacă $a > 0$ sau în sens invers dacă $a < 0$. Adăugăm la mulțimea ciclurilor și ciclul vid căruia îi corespunde vectorul 0 și care convenim că este rezultatul lui 0μ pentru orice ciclu μ . Vectorul 0 corespunde și ciclurilor care conțin arcele unui lanț elementar de la un vârf i la un vârf j parcuse de la i la j și apoi de la j la i .

Definim acum **suma a două cicluri elementare** μ^1 și μ^2 notată cu $\mu^1 + \mu^2$ ca fiind mulțimea arcelor ce au componentele corespunzătoare din vectorul sumă a celor doi vectori asociați ciclurilor μ^1 și μ^2 nenulă.

Exemplul 1.4.2.1.

S-ar putea ca în vectorii $\vec{\mu}^1$ și $\vec{\mu}^2$ componente corespunzătoare unui arc

să fie diferite de zero dar în $\vec{\mu}^1 + \vec{\mu}^2$ respectiva componentă să fie zero.

Pentru exemplificare considerăm graful din figura 1.4.2.1 în care s-au ordonat arcele conform numerotărilor din figură. Alegem ciclurile

$$\mu^1 = \{6, 2, 5, 9, 6\} \text{ și } \mu^2 = \{2, 1, 3, 5, 2\}$$

care au ca vectori asociați pe

$$\vec{\mu}^1 = (0, 0, 0, -1, 0, 0, 0, 0, 1, 1, 0, 0, 1)$$

și respectiv

$$\vec{\mu}^2 = (-1, 1, -1, 0, 0, 0, 0, 0, 0, 0, 0, -1)$$

atunci suma celor două cicluri este definită de vectorul

$$\vec{\mu}^1 + \vec{\mu}^2 = (-1, 1, -1, 0, 0, 0, 1, 1, 0, 0, 0, 0)$$

care este vectorul asociat ciclului $\mu^3 = \{2, 1, 3, 5, 9, 6, 2\}$. Ciclul $\{3, 1, 2, 1, 3\}$ are atașat vectorul nul din \mathbb{R}^{13} .

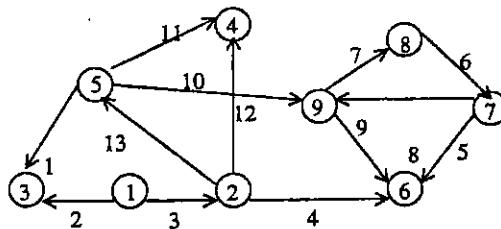


Fig 1.4.2.1.

Lema 1.4.2.1. Dacă $\mu \neq 0$ este suma a două cicluri elementare μ^1 și μ^2 atunci $\{u \in U \mid \mu_u \neq 0\}$ este fie un ciclu elementar fie o reuniune de cicluri elementare.

Demonstratie:

Dacă $\{u \in U \mid \mu_u^1 \neq 0\} \cup \{u \in U \mid \mu_u^2 \neq 0\} \subset \{u \in U \mid \mu_u \neq 0\}$ atunci lema are loc în mod evident.

Dacă însă componenta asociată unui singur arc u devine zero după însumare înseamnă că ea a fost parcursă de cele două cicluri elementare în sensuri diferite. Notăm extremitățile arcului u cu i și j . Schematic situația se prezintă ca în figura 1.4.2.2. Dacă se compune lanțul de la i la j din ciclul μ^1 cu lanțul de la j la i din ciclul μ^2 se obține un ciclu elementar ce are ca

vector asociat fie pe $\vec{\mu}$ fie pe $-\vec{\mu}$. Deci lema este demonstrată și în acest caz. \square

Dacă există mai multe arce ale căror componente din vectorul sumă sunt zero atunci într-un mod analog cazului precedent se descompun ciclurile μ^1 și μ^2 în mai multe lanțuri care apoi compuse formează un ciclu ce are ca vector asociat pe $\vec{\mu}$ sau pe $-\vec{\mu}$.

Suma a a cicluri egale cu ciclul μ este $a\mu$. Convenim ca pentru a și b două numere întregi, iar μ un ciclu elementar $a\mu + b\mu$ să fie ciclul $(a+b)\mu$. Extindem noțiunea de ciclu la succesiuni de vârfuri sau muchii care se pot descompune în cicluri elementare. Folosind noțiunea de ciclu cunoscută se poate defini recursiv mulțimea extinsă de cicluri după cum urmează. Prin *ciclu în sens larg* al unui graf G înțelegem:

- a. un ciclu elementar sau
- b. un ciclu elementar parcurs de un număr întreg de ori sau
- c. o reuniune de cicluri în sens larg.

Suma a două cicluri oarecare o definim ca fiind suma sumelor algebrice a ciclurilor elementare care compun cele două cicluri.

Lema 1.4.2.1 este valabilă și în cazul în care μ este o combinație liniară de cicluri cu scalari numere întregi. Aceasta pentru că dacă pentru un arc $u = (i, j)$ valoarea corespunzătoare din vectorul sumă este zero, adică numărul de treceri prin u pe un sens este egal cu numărul de treceri pe celălalt sens, atunci ciclurile elementare care trec prin u se pot descompune în lanțuri cu proprietatea că numărul lanțurilor ce intră în i respectiv j este egal cu al celor ce ies din i respectiv j . Componem apoi aceste lanțuri unul ce intră în i cu unul ceiese din i astfel ca ciclurile obținute să fie disjuncte și reunionea lor (ciclul în sens larg corespunzător) are ca vector asociat pe μ .

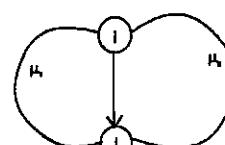


Fig 1.4.2.2.

Prin analogie cu noțiunile din algebra liniară spunem că mulțimea ciclurilor unui graf este mulțimea generată de către ciclurile elementare disjuncte ale grafului având ca scalari numere întregi. Această structură algebrică verifică axiomele unui spațiu liniar. Să reținem că mulțimea scalarilor are doar o structură de inel fără divizori ai lui zero. Astfel se poate defini un izomorfism între spațiul liniar al mulțimii ciclurilor unui graf și spațiul liniar al vectorilor asociati ciclurilor. Vom demonstra, corolarul 1.4.3.2, că numărul ciclomatic este dimensiunea acestui spațiu liniar. Pentru aceasta a rămas de arătat faptul că există $v(G)$ cicluri elementare liniar independente. Să remarcăm acum că deși ciclurile definite de vectorii μ^1, μ^2, μ^3 din exemplul 1.4.2.1 sunt toate cicluri elementare ele nu sunt liniar independente.

Un alt exemplu de spațiu vectorial de dimensiune $v(G)$ este dat în capitolul 2.

Teorema 1.4.2.1. Fie $G = (X, U)$ un multigraf iar $G' = (X, V)$ graful obținut din G prin adăugarea muchiei (i, j) . Atunci:

1. fie $\rho(G') = \rho(G)$ și $v(G') = v(G) + 1$ (dacă i și j sunt în aceeași componentă conexă a lui G);
2. fie $\rho(G') = \rho(G) + 1$ și $v(G') = v(G)$ (dacă i și j sunt în componente conexe diferite ale lui G).

Demonstrație:

Deoarece $m' = m + 1$ și $n' = n$, iar p' după cum i și j sunt sau nu în aceeași componentă conexă este p respectiv $p - 1$. Acum prin simplul calcul al numerelor $\rho(G)$ și $v(G)$ se obține concluzia teoremei. \square

Corolar 1.4.2.1. Pentru orice graf G avem $\rho(G) \geq 0$ și $v(G) \geq 0$.

Demonstrație:

Rezultatele se obțin dacă pornind de la graful vid $G' = (X, \Phi)$ în care $\rho(G') = 0$ și $v(G') = 0$ adăugăm pe rând câte o muchie până se obține graful G . Conform teoremei valorile lui $\rho(G)$ și $v(G)$ sunt numere naturale mai mari sau egale decât valorile de pornire. \square

Corolar 1.4.2.2. Dacă pentru un graf G , $v(G) = 0$ atunci graful nu conține cicluri (se subînțelege diferența de ciclul vid).

Demonstrație:

Procedând ca mai sus dacă valoarea lui $v(G)$ a rămas 0 înseamnă că la fiecare adăugare de muchie ne-am situat în cazul 2 al teoremei 1.4.2.1. Deci sau adăugat mereu muchii între vârfuri din componente conexe diferite, care muchii nu pot închide cicluri. \square

Corolar 1.4.2.3. Dacă pentru un graf $v(G) = 1$ atunci graful conține un singur ciclu elementar.

Demonstrație:

Considerăm că am obținut graful G pornind de la graful vid ca în cazul corolarului precedent. Cum $v(G) = 1$ înseamnă că s-a trecut prin cazul 1 al teoremei 1.4.2.1 doar o singură dată. În acea situație muchia (i, j) adăugată a închis un singur ciclu. Altfel dacă s-ar fi închis un al doilea ciclu înseamnă că între i și j ar fi existat cel puțin două lanțuri diferite deci în graful de până la acel moment există un ciclu. Cum muchiile ciclului s-au adăugat pe rând în momentul când s-a adăugat ultima muchie a ciclului respectiv am fi fost tot în situația 1 din

teorema 1.4.2.1 deci trebuie ca $v(G) \geq 2$. Prin urmare dacă $v(G) = 1$ graful are un singur ciclu. \square

Din corolarele 1.4.2.1 și 1.4.2.3 se deduce:

Corolar 1.4.2.4. Dacă graful G este fără cicluri, atunci $v(G) = 0$.

1.4.3. Arbori și păduri

Așa cum s-a definit în primul paragraf al acestui capitol prin *arbore* înțelegem un graf conex și fără cicluri.

Teorema 1.4.3.1. Fie $G = (X, U)$ un graf de ordin $n \geq 2$. Afirmațiile următoare sunt echivalente (și caracterizează noțiunea de arbore):

1. G este un arbore;
2. G este fără cicluri și are $n - 1$ muchii;
3. G este conex și are $n - 1$ muchii;
4. G este fără cicluri și prin adăugarea unei muchii între două vârfuri neadiacente se formează un singur ciclu;
5. G este conex și suprimând o muchie nu mai este conex;
6. între oricare două vârfuri ale grafului există un singur lanț.

Observație:

Să observăm că propozițiile 3 și 5 se pot formula

" G este conex și are număr minim de muchii între grafele conexe cu n vârfuri",

iar 2 și 4 exprimă

" G este fără cicluri și are număr maxim de muchii între grafele aciclice cu n vârfuri".

Demonstrație:

Vom da o demonstrație circulară echivalențelor din teoremă.

$1 \Rightarrow 2$

G fiind conex înseamnă $p = 1$. Deoarece G este fără cicluri din corolarul 1.4.2.4 deducem $v(G) = 0$ adică $m - n + 1 = 0$ de unde $m = n - 1$.

$2 \Rightarrow 3$

G fără cicluri $\Rightarrow v(G) = m - n + p = 0$ dar $m = n - 1$ deci $p = 1$ și prin urmare G este conex.

$3 \Rightarrow 4$

G fiind conex ($p = 1$) și are $n - 1$ muchii ($m = n - 1$) rezultă deci $v(G) = 0$.

Aplicând teorema 1.4.2.1 fiind în cazul 1 deducem $v(G') = v(G) + 1 = 1$. Conform corolarului 1.4.2.3 înseamnă că G' obținut din G prin adăugarea unei muchii conține un singur ciclu.

$4 \Rightarrow 5$

Vom demonstra prin metoda reducerii la absurd că G este conex. Dacă G nu este conex atunci adăugând o muchie între două componente conexe ale sale nu se formează nici un ciclu, ceea ce este în contradicție cu ipoteza. Deci G este fără cicluri și conex atunci $v(G) = 0$ de unde $m = n - 1$. Dacă suprimăm o muchie graful obținut are $v(G') = 0$, $m' = n - 2$ și $p' = v(G') - n - m = 2$ deci nu mai este conex.

$5 \Rightarrow 6$

G fiind conex între oricare două vârfuri există un lanț. Demonstrăm prin reducere la absurd că între oricare două vârfuri există un singur lanț. Dacă ar exista două vârfuri între care să existe două lanțuri atunci între cele două vârfuri există un ciclu și suprimând o muchie a ciclului graful rămâne conex.

$6 \Rightarrow 1$

Deoarece între oricare două vârfuri ale grafului există lanț graful este conex. Cum între oricare două vârfuri există un singur lanț graful nu are cicluri. Deci graful este conex și fără cicluri adică este arbore. \square

Teorema 1.4.3.2. Fie $n \in \mathbb{N}^*$ și $0 < d_1 \leq d_2 \leq \dots \leq d_n$, n numere naturale. Să se demonstreze că $d_1 + d_2 + \dots + d_n = 2n - 2$ dacă și numai dacă există un arbore $G = (X, U)$, cu $X = \{1, 2, \dots, n\}$ și d_i gradul vârfului i , $\forall i \in X$.

Demonstrație:

Dacă $G = (X, U)$ este un arbore cu $X = \{1, 2, \dots, n\}$ și d_i gradul vârfului i , $\forall i \in X$, pentru că $m = n - 1$ și $\sum_{i=1}^n g(i) = 2m$ deducem $d_1 + d_2 + \dots + d_n = 2n - 2$.

Demonstrăm cealaltă implicație prin inducție matematică.

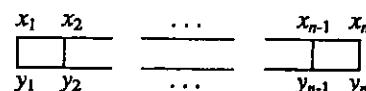
Pentru $n = 2$ din relația dată, considerată ca ecuație cu necunoscutele d_1 și d_2 din \mathbb{N}^* , rezultă că singura soluție este $d_1 = d_2 = 1$, iar arborele $G = (\{1, 2\}; \{(1, 2)\})$ este un argument că teorema este verificată și pentru $n = 2$. Fie $p \in \mathbb{N}^*$ și să presupunem că pentru fiecare set d_1, d_2, \dots, d_p de p numere naturale care verifică $d_1 + d_2 + \dots + d_p = 2p - 2$ există căte un arbore cu p vârfuri care au gradele d_1, d_2, \dots, d_p . Fie, acum $p + 1$ numere naturale $0 < d_1 \leq d_2 \leq \dots \leq d_p \leq d_{p+1}$ cu $d_1 + d_2 + \dots + d_{p+1} = 2p$. Deducem $d_1 = 1$, căci altfel $2 \leq d_1$ și $2p + 2 \leq d_1 + d_2 + \dots + d_{p+1} = 2p$. Pe de altă parte $d_n \geq 2$, căci dacă $d_i = 1$, $\forall i \in \{1, 2, \dots, p+1\}$ se obține $p + 1 = d_1 + d_2 + \dots + d_{p+1} = 2p$. Deci $\exists i \in \{2, \dots, p+1\}$ astfel încât $d_{i-1} < d_i$ îl alegem pe cel mai mic i cu această proprietate. Deoarece dispunem de p numere naturale care verifică $0 < d_2 \leq \dots \leq d_{i-1} \leq d_i - 1 \leq \dots \leq d_{p+1}$ și $d_2 + \dots + d_{i-1} + (d_i - 1) + \dots + d_{p+1} = 2p - 2$, conform ipotezei săcute există un arbore $G = (X, U)$ cu $X = \{2, \dots, p+1\}$ și $d_j = g(j)$, $\forall j \in X - \{i\}$ și $d_{i-1} = g(i)$. Construim graful $G' = (X'; U')$, unde $X' = X \cup \{1\}$ și $U' = U \cup \{(1, i)\}$. Acest graf este arbore pentru că $v(G') = v(G) = 0$ deoarece G' s-a obținut din G prin adăugarea unui vârf și apoi a unei muchii între vârfuri din componente conexe diferite (vezi teorema 1.4.2.1). Din $v(G') = 0$ deducem că G' nu are cicluri, în plus G' are $p + 1$ vârfuri și p muchii (G are p vârfuri și fiind arbore are $p - 1$ muchii). Deci G' este arbore și gradele vârfurilor sale verifică $d_j = g(j)$, $\forall j \in X'$. Adică am demonstrat că proprietatea este adevărată și pentru $p + 1$.

Conform metodei utilizate implicația are loc pentru $\forall n \in \mathbb{N}^*$. \square

Corolar 1.4.3.1. Orice arbore are cel puțin două vârfuri de grad 1 (vârfuri frunză).

Problema 1.4.3.1.

Se dă graful scără cu $2n$ vârfuri:



- a. Să se arate că acest graf are $\frac{(2+\sqrt{3})^n - (2-\sqrt{3})^n}{2\sqrt{3}}$ arbori parțiali (grafe parțiale, cu $2n$ vârfuri, care sunt arbori).
- b. Câte mulțimi K cu n muchii neadiacente (cuplaje maxime și perfecte) există.

Rezolvare:

a. Pentru a rezolva cerința problemei vom determina numărul de elemente al mulțimii formată din toți arborii parțiali. Deoarece se cere un număr a cărui expresie depinde de n , se poate interpreta că se cere expresia termenului general al unui șir de numere. Determinarea acestei expresii se poate face prin a determina o relație de recurență. Apoi, fie se scrie un program care să utilizeze o procedură recursivă și care să afișeze un număr fixat de valori ale șirului în cauză, fie se determină expresia termenului general de exemplu pe baza ecuației caracteristice dacă relația de recurență este liniară. Pentru a aplica metoda sugerată, notăm cu A_n mulțimea arborilor parțiali ai grafului scară cu $2n$ vârfuri și descompunem această mulțime într-o parte cu mulțimi a căror cardinal, $a_n = |A_n|$, este exprimabil prin cardinalele mulțimilor A_p cu $p < n$. Vom considera submulțimi ale lui A_n după cum conțin sau nu muchii dintre cele 3 care apar în plus față de graful scară cu $2n-2$ muchii în graful A_n . Utilizăm descompunerea

$$A_n = B \cup C \cup D \cup E$$

unde

$$\begin{aligned} B &= \{P \in A_n \mid (x_{n-1}, x_n) \notin P\} = \{P \in A_n \mid (y_{n-1}, y_n) \text{ și } (x_n, y_n) \in P\}, \\ C &= \{P \in A_n \mid (x_n, y_n) \notin P\} = \{P \in A_n \mid (x_{n-1}, x_n) \text{ și } (y_{n-1}, y_n) \in P\}, \\ D &= \{P \in A_n \mid (y_{n-1}, y_n) \notin P\} = \{P \in A_n \mid (x_{n-1}, x_n) \text{ și } (x_n, y_n) \in P\}, \\ E &= A_n - (B \cup C \cup D) = B = \{P \in A_n \mid (x_{n-1}, x_n); (y_{n-1}, y_n) \text{ și } (x_n, y_n) \in P\}. \end{aligned}$$

Ultimele egalități de mai sus au loc pentru că orice arbore este conex și deci există lanț între x_n respectiv y_n și fiecare dintre celelalte vârfuri. Deoarece între mulțimile B , C respectiv D și A_{n-1} există câte o bijecție definită de eliminarea vârfurilor x_n și y_n și a muchiilor incidente acestora din fiecare arbore $P \in B$, C respectiv D . Graful astfel obținut din P notat $f_B(P)$, $f_C(P)$ respectiv $f_D(P)$ este arbore (în baza condiției 2 a teoremei 1.4.3.1.) și este din A_{n-1} . Funcțiile f_B , f_C și f_D precizate sunt (se demonstrează ușor) injective și surjective deci bijective. De aici se deduce $|B| = |C| = |D| = |A_{n-1}| = a_{n-1}$. Pentru a exprima cardinalul lui E cu termeni ai șirului (a_n) , facem următoarea descompunere a lui A_{n-1} : $A_{n-1} = Y \cup Z$, unde

$$Y = \{P \in A_{n-1} \mid (x_{n-1}, y_{n-1}) \notin P\} \quad \text{și} \quad Z = \{P \in A_{n-1} \mid (x_{n-1}, y_{n-1}) \in P\}$$

Analog calculului cardinalelor lui B , C și D se obține $|Y| = a_{n-2}$. Pe de altă parte între mulțimile E și Z există bijecția $f_E: E \rightarrow Z$ definită astfel pentru fiecare $P \in E$ construim pe $f_E(P)$ ca fiind graful obținut din E prin eliminarea vârfurilor x_n și y_n și a celor trei muchii din P incidente lor și apoi se adaugă muchia (x_{n-1}, y_{n-1}) . Funcția astfel definită este corectă pentru că $f_E(P)$ este arbore (în baza condiției 2 din teorema 1.4.3.1.) și este din Z . Această funcție este injectivă și surjectivă și pentru că $Y \cap Z = \emptyset$ deducem:

$$|E| = |Z| = |A_{n-1}| - |Y| = a_{n-1} - a_{n-2}.$$

Din cele arătate mai sus și pentru că $B \cap C \cap D \cap E = \emptyset$ obținem:

$$a_n = 3a_{n-1} + a_{n-1} - a_{n-2} \Leftrightarrow a_{n-1} - 4a_{n-1} + a_{n-2} = 0$$

Ecuația caracteristică asociată acestei relații de recurență are rădăcinile $2 \pm \sqrt{3}$ și deci

$$a_n = c_1 (2 + \sqrt{3})^n + c_2 (2 - \sqrt{3})^n.$$

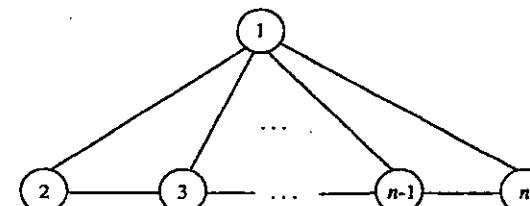
Constantele c_1 și c_2 se determină din valorile $a_1 = 1$ și $a_2 = 4$ obținute prin numărarea arborilor parțiali ai grafului scară cu 2 respectiv 4 vârfuri. Prin rezolvarea sistemului astfel format obținem $c_1 = \frac{1}{2\sqrt{3}}$ și $c_2 = -\frac{1}{2\sqrt{3}}$, ceea ce conduce la rezolvarea problemei.

b. Într-un mod asemănător se determină că șirul numerelor al cărui termen general se cere este șirul lui Fibonacci ($f_n = f_{n-1} + f_{n-2}$ cu $f_0 = f_1 = 1$ și $f_2 = 2$) deci

$$f_n = \frac{1}{\sqrt{5}} \left[\left(\frac{1+\sqrt{5}}{2} \right)^{n+1} - \left(\frac{1-\sqrt{5}}{2} \right)^{n+1} \right]$$

Problema 1.4.3.2.

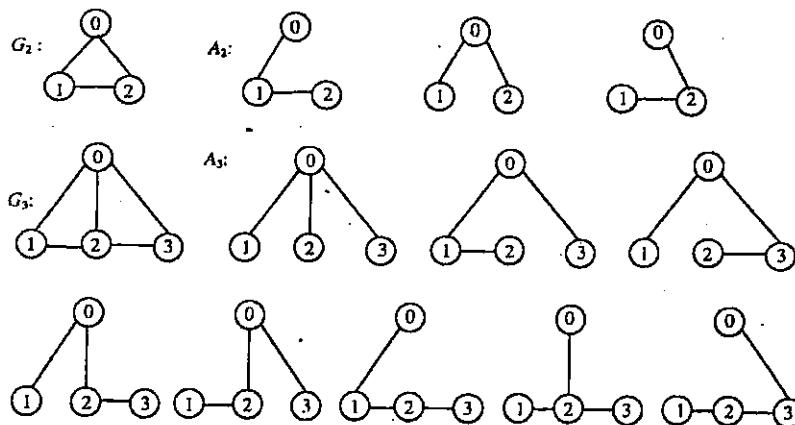
Pentru fiecare $n \in \mathbb{N}^*$ se consideră graful $G_n = (X, U)$, cu $X = \{0, 1, \dots, n\}$ și $U = \{(i, j) \mid i = 0 \text{ sau } j - i = 1\}$. Acest graf reprezentat geometric este



Să se determine numărul arborilor de acoperire ai lui G_n .

Rezolvare:

Similar rezolvării problemei 1.4.3.1 vom considera mulțimea A_n formată din toți arborii de acoperire ai lui G_n , adică $A_n = \{P \mid P = (X, V), V \subset U, P \text{ este arbore}\}$. Se poate defini șirul $(a_n)_{n \geq 1}$. Ceea ce se cere, se poate interpreta că, este să se determină termenul general al acestui șir. Pentru $n = 1, 2, 3$ și 4 , prin enumerarea grafelor parțiale ale lui G_n se obțin elementele lui A_n și $a_1 = 1, a_2 = 3, a_3 = 8$ și $a_4 = 21$. Reprezentarea geometrică este:



Pentru a urma metoda de la problema precedentă facem următoarea partitiorare $A_n = B \cup C \cup D$, unde:

- $B = \{P \in A_n \mid (0, n) \in P \text{ și } (n-1, n) \notin P\}$, adică mulțimea arborilor parțiali ai lui G_n care conțin muchia $(0, n)$ și nu conțin muchia $(n, n-1)$;
- $C = \{P \in A_n \mid (0, n) \notin P \text{ și } (n-1, n) \in P\}$, adică mulțimea arborilor parțiali ai lui G_n care conțin muchia $(n-1, n)$ și nu conțin muchia $(0, n)$;
- $D = \{P \in A_n \mid (0, n) \in P \text{ și } (n-1, n) \in P\}$, adică mulțimea arborilor parțiali ai lui G_n care conțin atât muchia $(0, n)$ cât și muchia $(n, n-1)$.

Se deduce că $B \cap C = B \cap D = C \cap D = \Phi$. Vom demonstra că există bijectii între mulțimile:

B și A_{n-1} ; C și A_{n-1} și D și $\bigcup_{k=0}^{n-2} A_k$, am presupus $n \geq 2$.

Astfel

- Definim $f: B \rightarrow A_{n-1}$, prin care pentru fiecare $P \in B$ îi corespunde $f(P)$ care este graful obținut din P prin eliminarea vârfului n și muchiei $(0, n)$. Din definiție deoarece P are $n+1$ vârfuri, este conex și are n muchii, deducem că $f(P)$ are n vârfuri, este conex și are $n-1$ muchii. Deci $f(P) \in A_{n-1}$. Dacă $P_1, P_2 \in B$ și $P_1 \neq P_2$ rezultă că fiecare, dintre arborii P_1 și P_2 , are cel puțin o muchie (neincidentă vârfului n) care nu este în celălalt arbore. Deci, $f(P_1) \neq f(P_2)$, adică f este injectivă. Cum, prin adăugarea vârfului n și a muchiei $(0, n)$ oricărui arbore $S \in A_{n-1}$ se obține un arbore $P \in B$ și deci $f(P) = S$, pentru $P \in B$, deducem că f este și surjectivă. Așadar, $|B| = |A_{n-1}| = a_{n-1}$.
- Analog, funcția $g: C \rightarrow A_{n-1}$, definită prin $g(P) =$ graful obținut, din arboarele $P \in C$, prin eliminarea vârfului n și a muchiei $(n, n-1)$, este bijectivă. Deci, $|C| = |A_{n-1}| = a_{n-1}$.
- Convenim ca A_0 să conțină doar graful (arborele) care are doar vârful 0 și nici o muchie. Definim funcția $h: \bigcup_{k=0}^{n-2} A_k \rightarrow D$, astfel

pentru fiecare $P \in \bigcup_{k=0}^{n-2} A_k$ rezultă că există un $k \in \{0, 1, \dots, n-2\}$

astfel încât $P \in A_k$ și alegem pe $h(P)$ ca fiind graful obținut din P prin adăugarea vârfurilor $k+1, \dots, n$ și a lanțului $\{0, n, n-1, \dots, k+1\}$. Este ușor de demonstrat că graful $h(P) \in D$ și h este bijectie.

Din cele de mai sus, deducem că

$$a_n = |A_n| = |B| + |C| + |D| = a_{n-1} + a_{n-1} + \sum_{k=0}^{n-2} a_k =$$

$$= 2a_{n-1} - a_{n-2} + (2a_{n-2} + \sum_{k=0}^{n-3} a_k) = 2a_{n-1} - a_{n-2} + a_{n-1}.$$

Deci termenii şirului $(a_n)_{n \in \mathbb{N}}$ verifică relaţia de recurenţă:

$$a_n - 3a_{n-1} + a_{n-2} = 0.$$

Cum, rădăcinile ecuaţiei $x^2 - 3x + 2 = 0$ sunt $x_{1,2} = \frac{3 \pm \sqrt{5}}{2}$, expresia termenului general este

$$a_n = c_1 \left(\frac{3+\sqrt{5}}{2} \right)^n + c_2 \left(\frac{3-\sqrt{5}}{2} \right)^n.$$

Constantele c_1 şi c_2 obţinute din sistemul $a_1 = 1$ şi $a_2 = 8$ sunt $c_1 = -c_2 = \frac{1}{\sqrt{5}}$. Prin urmare:

$$\begin{aligned} a_n &= \frac{1}{\sqrt{5}} \left[\left(\frac{3+\sqrt{5}}{2} \right)^n - \left(\frac{3-\sqrt{5}}{2} \right)^n \right] = \\ &= \frac{1}{\sqrt{5}} \left[\left(\frac{1+\sqrt{5}}{2} \right)^{(2n-1)+1} - \left(\frac{1-\sqrt{5}}{2} \right)^{(2n-1)+1} \right]. \end{aligned}$$

Pe de altă parte ştim că termenii şirului lui Fibonacci au expresia

$$f_n = \frac{1}{\sqrt{5}} \left[\left(\frac{1+\sqrt{5}}{2} \right)^{n+1} - \left(\frac{1-\sqrt{5}}{2} \right)^{n+1} \right].$$

Deci un alt mod de exprimarea a valorilor termenilor şirului (a_n) , cu ajutorul termenilor şirului lui Fibonacci, este $a_n = f_{2n-1}, \forall n \in \mathbb{N}^*$.

Problema 1.4.3.3.

Fie $G = (X, U)$ un arbore de ordin n și dimensiune m . Notăm cu v_i numărul vârfurilor de grad i , pentru fiecare $i \in \mathbb{N}$ și cu v numărul vârfurilor de grad mai mare ca 3. Să se demonstreze că $v_i \geq v + 2$.

Deoarece G este arbore rezultă că

$$m = n-1 \text{ sau } 2m = 2n - 2.$$

Pe de altă parte

$$2(v_1 + v_2 + v_3 + \dots) = 2n$$

și pentru că kv_k este numărul de muchii incidente tuturor vârfurilor de grad k și fiecare muchie este incidentă la două vârfuri se deduce:

$$v_1 + 2v_2 + 3v_3 + \dots = 2m.$$

Din ultimele trei relații se deduce relația de demonstrație.

Așadar, orice arbore are cel puțin 2 vârfuri frunză.

Definiția 1.4.3.1. Un graf care nu are cicluri se numește *pădure*.

Deci fiecare componentă conexă a unei păduri este un arbore. O pădure formată dintr-o singură componentă conexă este un arbore.

Fiind dat un graf oarecare acesta are ca subgrafe mai multe păduri. Astfel se poate pune problema de a determina o pădure cu anumite proprietăți.

Definiția 1.4.3.2. Pentru un graf $G = (X, U)$ subgrauul $P = (X, T)$ este *pădure maximală* dacă T este maximală în raport cu incluziunea în mulțimea părților mulțimii U .

Vom da în continuare un algoritm pentru construcția unei păduri maximale a unui graf $G = (X, U)$ dat. Algoritmul analizează muchiile grafului G într-o ordine oarecare și colorează muchia analizată cu una dintre culorile roșu sau verde. La fiecare moment G_R reprezintă subgrauul care conține doar muchiile colorate cu roșu, iar G_C reprezintă subgrauul care conține toate muchiile colorate ale grafului. Să descriem acum algoritmul. Fie u muchia din G ce se analizează la un moment dat. Sunt posibile două situații:

- prin u trece un ciclu cu toate celelalte muchii colorate în roșu, atunci vom colora muchia u în verde;
- prin u nu trece nici un ciclu cu toate celelalte muchii colorate în roșu, atunci vom colora muchia u în roșu.

Observația 1.4.3.1.

Muchia u ce se analizează la un moment dat are extremitățile în aceeași componentă conexă a lui G_R și a lui G_C sau în componente conexe

diferite după cum ne aflăm în cazul *a* respectiv *b* al algoritmului.

Teorema 1.4.3.3. Subgraful G_R este o pădure maximală a grafului G .

Demonstrație:

Din modul de construcție graful G_R este fără cicluri la fiecare moment, deci și în finalul algoritmului. Tot din modul de construcție toate muchiile lui G au fost colorate, atunci orice altă muchie în afara celor ale lui G_R sunt colorate în verde. Deci dacă s-ar mai adăuga o muchie a lui G la G_R acesta ar conține cicluri și nu ar mai fi pădure. \square

Corolar 1.4.3.2.

- Dacă $P = (X, T)$ este o pădure maximală a grafului $G = (X, U)$ atunci G și P au același număr de componente conexe.
- Dacă G este de ordin n și are p componente conexe atunci G_R are $n - p$ muchii.
- Dacă $P = (X, T)$ este o pădure maximală a grafului $G = (X, U)$ atunci prin toate arcele $u \in U - T$ trece un singur ciclu cu toate celelalte muchii din T .
- Dacă $P = (X, T)$ este o pădure maximală a grafului $G = (X, U)$ atunci prin toate arcele $u \in T$ trece căte un singur cociclu θ^u cu toate celelalte muchii aparținând lui $U - T$.

Demonstrație:

a. P fiind subgraf al lui G are doar o parte dintre muchiile lui G . Deci P nu poate avea mai puține componente conexe decât G . Dacă P ar avea mai multe componente conexe decât G atunci înseamnă că în G există cel puțin o muchie care conectează două componente conexe ale lui P . Ori adăugând o asemenea muchie la P se obține o pădure care ar contrazice maximalitatea lui P . În concluzie orice pădure maximală a unui graf are același număr de componente conexe ca și graful.

b. Din modul de colorare al muchiilor la fiecare adăugare de muchie în G_R numărul de componente conexe din G_R scade cu 1. Dar G_R pornește de la graful vid care are n componente conexe și se ajunge la un graf cu p componente conexe. Prin urmare au fost adăugate $n - p$ muchii.

c. P fiind o pădure maximală are același număr de componente conexe ca și G . Deci orice muchie $u \in U - T$ are extremitățile în același arbore al lui P .

Conform teoremei 1.4.3.1 prin adăugarea unei muchii la un arbore,adică la P , se va forma un singur ciclu.

d. Fie $u = (i, j) \in T$ atunci u face parte dintr-un arbore. Dacă se elimină muchia u din P , deci din arborele respectiv acesta nu mai este conex. Notând cu A mulțimea vârfurilor unei componente conexe a arborelui atunci $\omega(A)$ este un cociclu cu proprietatea precizată. Să demonstrăm că acest cociclu este unic în mulțimea cocicclurilor grafului G cu proprietatea că din T conține doar muchia u . Mai întâi să remarcăm că în orice alt cociclu care conține doar muchia u din T intră doar muchii ale componentei conexe a lui G care conține muchia u , altfel din acel cociclu ar trebui să mai facă parte și alte muchii din T . Deci putem presupune fără a restrâng generalitatea că graful G este conex. Dacă ar exista un alt cociclu $\omega(B)$ cu proprietatea precizată atunci înseamnă că unul dintre cociccluri are cel puțin o muchie (k, l) din $U - T$ în plus. Să presupunem că vârfurile k și l sunt în A dar numai unul este în B (dacă vârfurile k și l ar fi din B s-ar raționa analog acestui caz). În acest caz cum $P - \{u\}$ are două componente conexe și vârfurile k și l sunt în aceeași componentă conexă între ele există un lanț μ cu muchii din P . Dar vârfurile k și l nefăcând parte ambele din B sau din $X - B$ înseamnă că o muchie din lanțul $\mu \subset T$ va face parte din $\omega(B)$. Acest fapt contrazice modul de alegere al lui $\omega(B)$ și deci cociccul ce trece prin u este unic cu proprietatea precizată. \square

Teorema 1.4.3.4. Fie $G = (X, U)$ un graf de ordin, n de dimensiune m și cu p componente conexe. Fie $P = (X, T)$ o pădure maximală a lui G , pentru $u \in U - T$ notăm cu μ^u unicul ciclu conținut de $T \cup \{u\}$. Ciclurile μ^u , considerate pentru fiecare $u \in U - T$, formează o bază de cicluri pentru spațiul liniar al mulțimii ciclurilor grafului G .

Demonstrație:

Fie μ un ciclu oarecare cu vectorul asociat notat tot cu $\mu = (\mu_1, \mu_2, \dots, \mu_m)$. Atunci vectorul

$$\delta = \mu - \sum_{u \in U - T} \mu_u \mu^u$$

fiind o combinație liniară de vectori asociați unor cicluri este vectorul asociat unui ciclu (în sens larg) cu toate componente corespunzătoare arcelor u din $U - T$ egale cu 0.

Deci

$$\{u \in U \mid \delta_u \neq 0\} \subset T.$$

Dar conform lemei 1.4.2.1 $\{u \in U \mid \delta_u \neq 0\}$ trebuie să conțină cel puțin un ciclu elementar. Cum $P = (X, T)$ este o pădure înseamnă că T nu conține cicluri și deci

$$\{u \in U \mid \delta_u \neq 0\} = \emptyset,$$

aceasta înseamnă că $\delta = 0$ de unde

$$\mu = \sum_{u \in U - T} \mu_u \mu^u.$$

Prin urmare $\{\mu^u \mid u \in U - T, \mu^u$ unicul ciclu din $T \cup \{u\}\}$ este un sistem de generatori pentru mulțimea ciclurilor grafului G .

Pe de altă parte deoarece fiecare arc $u \in U - T$ este conținut într-un singur ciclu μ^u deducem că ciclurile μ^u cu $u \in U - T$ sunt liniar independente. \square

Corolar 1.4.3.2. Dimensiunea spațiului liniar format din ciclurile unui graf G este $v(G)$, adică numărul ciclomatic.

Demonstrație:

Pentru graful G considerăm pădurea maximală $G_R = (X, T)$ obținută cu algoritmul de colorare a arcelor grafului G prezentat mai sus. Dacă graful G este de ordin n , dimensiuni m și cu p componente conexe atunci conform corolarului 1.4.3.1. $b \mid T \mid = n - p$. Prin urmare dimensiunea bazei mulțimii ciclurilor lui G este

$$|\{\mu^u \mid u \in U - T, \mu^u$$
 unicul ciclu din $T \cup \{u\}\}| = |U - T| = m - n + p.$

Într-un mod analog se demonstrează că spațiul liniar format din mulțimea cociclurilor grafului G are dimensiunea $p(G)$, adică numărul cociclomatic. \square

1.4.4. Arbore de pondere minimă

Vom considera în continuare că graful G este conex deci orice pădure a grafului G este un arbore. Considerăm de asemenea că fiecărui arc u al grafului G i s-a atașat o valoare $l(u)$ care este număr real.

Prin *valoarea sau ponderea unui arbore* $P = (X, T)$ subgraf al grafului G notată $l(P)$ înțelegem

$$\sum_{u \in T} l(u).$$

Problemă:

Fiind dat un graf G cu valori pentru fiecare arc, se cere să se determine un arbore al său care să aibă valoarea minimă.

Teorema 1.4.4.1. O condiție necesară și suficientă pentru ca $P = (X, T)$ să fie arbore de pondere minimă a grafului G este

$$\forall u \in T \text{ și cocicul unic } \theta^u \text{ cu } \theta^u \cap T = \{u\} :$$

$$l(u) \leq l(v), \forall v \in \theta^u.$$

Demonstrație:

Necesitatea

Demonstrăm prin metoda reducerii la absurd. Fie $P = (X, T)$ un arbore de valoare minimă a grafului $G = (X, U)$ și să presupunem că

$$\exists u \in T \text{ și } \exists v \in \theta^u \text{ cu } l(u) > l(v).$$

Considerăm subgraful $P' = (X, T \cup \{v\} - \{u\})$ al lui G . P' fiind un arbore, din teorema 1.4.3.1 deducem că $G(T - \{u\})$ este format din două componente, care sunt conectate de orice muchie din θ^u . Înseamnă că P' este conex. Deci P' fiind conex și cu $n - 1 = |T'|$ muchii este arbore. Dar $l(P') < l(P)$ ceea ce contrazice faptul că P este de valoare minimă. Deci s-a demonstrat necesitatea.

Suficiența

Fie $P = (X, T)$ un arbore care îndeplinește condițiile din teoremă. Să considerăm acum un arbore $P' = (X, T')$ care este de valoare minimă. P și P' fiind arbori ai lui G înseamnă că $|T| = |T'| = n - 1$. Vom demonstra că pornind de la arborele P' muchiile din $T' - T$ se pot înlocui pe rând cu muchii din $T - T'$ pentru a ajunge la P și valoarea arborilor obținuți să fie $l(P')$.

Dacă $P \neq P'$ atunci există u în $T - T'$. Fie θ^u cocicul din G cu

$$\theta^u \cap T = \{u\}.$$

Cum $(X, T - \{u\})$ nu este conex, iar (X, T') este conex deducem

$$\exists v \in T' \cap \theta^u \text{ și deci } l(u) \leq l(v).$$

Dacă $l(u) < l(v)$ atunci $P'' = (X, T' \cup \{u\} - \{v\})$ este un arbore de valoare mai mică decât valoarea lui P' , care este de valoare minimă. Deci $l(u) = l(v)$ și $l(P'') = l(P')$. Procedând în mod similar și cu celelalte muchii din $T' - T$, care sunt în număr finit, se ajunge la concluzia că $l(P) = l(P')$ și deci P este un arbore de valoare minimă. \square

Observația 1.4.4.1.

Condiția din enunțul teoremei 1.4.4.1 se poate interpreta astfel:

Nici o muchie u a arborelui P nu poate fi înlocuită de o altă muchie v a grafului G astfel încât să se obțină un arbore de valoare mai mică decât valoarea lui P .

Teorema 1.4.4.2. O condiție necesară și suficientă pentru ca $P = (X, T)$ să fie un arbore de pondere minimă a grafului $G = (X, U)$ este $\forall v \in U - T$ pentru ciclul unic $\mu^v \subset T \cup \{v\}$ să aibă loc proprietatea:

$$l(u) \leq l(v), \forall u \in \mu^v.$$

Demonstrație:

Necesitatea

Vom demonstra folosind metoda reducerii la absurd. Fie $P = (X, T)$ un arbore de valoare minimă a grafului $G = (X, U)$ și să presupunem că

$$\exists v \in U - T \text{ și } \exists u \in \mu^v : l(u) > l(v).$$

Atunci subgraful $(X, T \cup \{v\})$ conține ciclul μ^v din care eliminând muchia u se obține subgraful $(X, T \cup \{v\} - \{u\})$ care este conex și are $n - 1$ muchii deci este arbore. Valoarea acestui arbore este mai mică decât valoarea arborelui P ceea ce contrazice ipoteza că P este de valoare minimă.

Suficiența

Fie $P = (X, T)$ un arbore cu proprietatea din teoremă. Atunci

$$\forall u \in T \text{ considerăm cociclul } \theta^u \text{ cu } \theta^u \cap T = \{u\}.$$

Pentru $\forall v \in \theta^u$ atunci prin considerarea ciclului $\mu^v \subset T \cup \{v\}$ din ipoteza teoremei deducem că $l(u) \leq l(v)$. Adică am demonstrat că are loc condiția din teorema 1.4.4.1 și deci arborele P este un arbore de valoare minimă. \square

Observația 1.4.4.2.

Condiția din enunțul teoremei 1.4.4.2 se poate interpreta astfel:

Nici o muchie v care nu face parte din arborele P nu poate să înlocuiască vreo muchie u a arborelui P astfel încât subgraful obținut să fie arbore și să aibă o valoare mai mică decât valoarea lui P .

Corespunzător teoremei 1.4.4.1 și 1.4.4.2 se pot construi doi algoritmi pentru determinarea unui arbore de valoare minimă într-un graf dat. În funcție de faptul că muchiile grafului se dau sau nu ordonate, în raport cu valorile acestora, algoritmii au variante diferite. Deci, vom prezenta în continuare patru algoritmi pentru determinarea unui arbore de valoare minimă într-un graf G dat. Fiecare dintre acești algoritmi este cunoscut cu numele de *algoritmul lui Kruskal*. Kruskal a elaborat în 1956 [Kru56] primul algoritm pentru problema determinării unui arbore de pondere minimă.

Algoritmul 1.4.4.1.

Se dă U prin lista muchiilor în ordine crescătoare a valorilor lor.

(a) fie $T = \{u_1\}$;
(b) pentru $k=2, m$ execută
 dacă $T \cup \{u_k\}$ nu conține cicluri
 atunci $T = T \cup \{u_k\}$;
 sfz;
sfp.

Algoritmul 1.4.4.2.

Se dă U prin lista muchiilor în ordine descrescătoare a valorilor lor.

(a) fie $T = U$
(b) pentru $k=1, m$ execută
 dacă $T - \{u_k\}$ este conex
 atunci $T = T - \{u_k\}$;
 sfz;
sfp.

Observația 1.4.4.3.

În algoritmii de mai sus se pot parurge muchiile grafului doar până când T conține $n - 1$ muchii. Deci ultimii $m - n + 1$ execuții ale blocurilor "pentru" nu sunt necesare.

Algoritmul 1.4.4.3.

Se dă U prin lista muchiilor într-o ordine oarecare.

(a) fie $T = \{u_1\}$;
(b) pentru $k=2, m$ execută

fie $T = T \cup \{uv\}$;
 Dacă T conține un ciclu μ
 atunci determină muchia v din μ cu valoarea maximă;
 fie $T = T - \{v\}$;
 sfps;

Algoritmul 1.4.4.4.

Se dă U prin lista muchiilor sale într-o ordine oarecare.

(a) fie $T = U$;
 (b) pentru $k=1, m$ execută
 dacă $(X, T - \{u_k\})$ nu este conexă atunci
 alege A o componentă conexă a lui $(X, T - \{u_k\})$;
 determină muchia v de valoare minimă din $\omega(A)$;
 fie $T = T \cup \{v\} - \{u_k\}$;
 sfps.

Observația 1.4.4.4.

- Pentru ultimii doi algoritmi parcurgerea întregii multimi este absolut necesară.
- Algoritmii lui Kruskal pornesc de la graful (X, T) care are toate vârfurile lui G și muchiile $T = \emptyset$ sau $T = U$ și se modifică doar mulțimea de muchii T pentru a obține un arbore parțial de pondere minimă.

Un alt algoritm de determinare a unui arbore de pondere minimă pentru graful $G = (X, U)$ dat este algoritmul lui Prim [Pri57]. Acest algoritm pornește de la graful (S, T) cu S o mulțime cu un vârf oarecare, iar $T = \emptyset$ și în $n - 1$ pași se vor adăuga $n - 1$ vârfuri și tot atâtea muchii (cât un vârf și o muchie la fiecare pas). Corectitudinea algoritmului lui Prim este dată teorema 1.4.4.1. și constă în a alege la pasul k muchia u din $\omega(S_k)$ de valoare minimă dintre acestea, unde S_k mulțimea de vârfuri selectate până la pasul k .

Algoritmul 1.4.4.5. (algoritmul lui Prim)

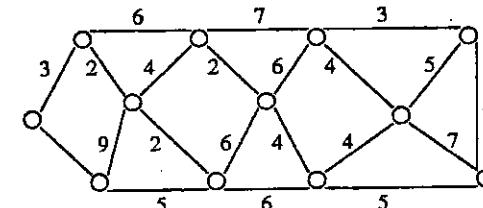
Se dă graful $G = (X, U)$ și valorile muchiilor sale.

(a) $k := 1$;
 $T = \emptyset$;
 Alege x_k din X ;
 $S_k = \{x_k\}$;

(b) pentru $k=2, n$ execută
 pentru $i := 1, k-1$ execută
 alege ca y_i din $X - S_k$ vârful adjacente lui x_i astfel încât muchia (x_i, y_i) este
 muchia de valoare minimă din $\omega(X - S_k) \cap \omega(x_i)$
 sfpentru
 alege dintre muchiile selectate muchia (x_k, y_k) de valoare minimă
 adică $l(x_k, y_k) = \min \{l(x_i, y_i) \mid i = 1, k-1\}$;
 $x_k := y_k$;
 $S_k := S_k \cup \{x_k\}$;
 $T := T \cup \{(x_i, y_i)\}$;
 sfpentru.

Problema 1.4.4.1.

Să se determine căte un arbore de acoperire de pondere minimă pentru graful reprezentat geometric mai jos.



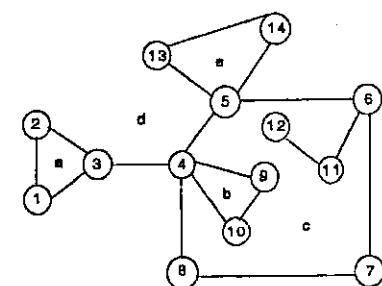
1.5. Grafe particulare

1.5.1. Grafe planare

Noțiunile de graf planar și cea de față a unui graf planar au fost date în definiția 1.2.1.1.

Definiția 1.5.1.1. *Frontiera* unei fețe este mulțimea muchiilor care ating acea față.

De exemplu în figura 1.5.1.1 fața c are frontieră formată din muchiile $(4,5)$; $(5,6)$; $(6,11)$; $(11,12)$; $(6,7)$; $(7,8)$; $(8,4)$; $(4,10)$; $(10,9)$ și $(9,4)$.



Două fețe se numesc **adiacente** dacă frontierele lor au o muchie comună (însă două fețe care nu se ating decât într-un vârf nu sunt adiacente). De exemplu în figura 1.5.1.1 fețele d și c sunt adiacente dar fețele e și c nu sunt adiacente.

Orice graf planar are o singură față nemărginită pe care o numim **față infinită**. Celelalte fețe le numim **fețe finite**.

Numim **contur** al unei fețe ciclul elementar din frontieră acelei fețe care conține în interiorul reprezentării sale toate celelalte muchii ale frontierei. De exemplu fața c din figura 1.5.1.1 are conturul $\{(4,5), (5,6), (6,7), (7,8), (8,4)\}$. Să remarcăm că față infinită nu are contur, dar are frontieră.

Observația 1.5.1.1.

Într-un graf planar frontieră unei fețe este formată din: unul sau mai multe cicluri elementare disjuncte, muchii suspendate sau muchii care unesc cicluri disjuncte (și care sunt istmuri).

Numărul fețelor unui graf planar îl notăm cu f .

Teorema 1.5.1.1. Într-un graf planar G , contururile fețelor finite constituie o bază pentru spațiul ciclurilor grafului respectiv (deci are loc $f = v(G) + 1$).

Demonstrație:

Vom demonstra prin inducție după f . Dacă graful nu are nici o față finită (adică $f = 1$ și $v(G) = 0$) și dacă graful are o singură față finită (adică $f = 2$ și $v(G) = 1$) teorema este adevărată.

Să presupunem că teorema este adevărată pentru un graf care are $f - 1$ fețe finite și să demonstrează că este adevărată și pentru un graf care are f fețe finite. Alegem o muchie u care atinge față infinită și este în conturul unei fețe finite. Suprimând această muchie se obține un graf G' cu $f - 1$ fețe finite. Conform ipotezei de inducție contururile fețelor finite sunt liniar independente și $f - 1 = v(G') + 1$. Adăugând muchia u grafului G' se obține graful G , care are o față finită în plus față de G' . Conturul acestei fețe este un ciclu care are muchia u și această muchie nu se află în nici un alt ciclu (contur) al celor $f - 2$ fețe finite ale grafului G' care sunt și contururi pentru aceleași $f - 2$ fețe din G . Deci cele $f - 1$ cicluri (contururile fețelor finite) din G sunt liniar independente. Pe de altă parte conform teoremei 1.4.2.1 cazul 1) (deoarece extremitățile muchiei u sunt în aceeași componentă conexă) deducem $v(G) = v(G') + 1$ și deci $f = v(G) + 1$. Așadar cele $f - 1$ contururi sunt cicluri liniar independente și sunt ca număr egale cu dimensiunea spațiului liniar al ciclurilor grafului G . Deci ele formează o bază a acestui spațiu liniar. \square

Corolar 1.5.1.1. Într-un graf planar conex cu n vârfuri, m muchii și f fețe are loc relația

$$n - m + f = 2 \quad (\text{formula lui Euler}).$$

Demonstrație:

Cum $p = 1$ și $f = v(G) + 1$, deducem

$$f = m - n + 1 + 1 \text{ de unde } n - m + f = 2. \square$$

Corolar 1.5.1.2. În orice graf planar conex $G = (X, U)$, există $i \in X$ astfel încât $g(i) \leq 5$.

Demonstrație:

G nefiind multigraf fiecare față are în contur cel puțin 3 muchii distincte și o muchie este în frontieră a cel mult două fețe. Deci $3f \leq 2m$ sau $f \leq 2m/3$. Presupunând $g(i) \geq 6$ pentru orice vârf i deducem

$$6n \leq 2m \text{ și deci } n \leq 2m/6 = m/3.$$

Prin urmare

$$2 = n - m + f \leq m/3 - m + 2m/3 = 0 \text{ contradicție. } \square$$

Exemplul 1.5.1.1. (Euler)

Să considerăm, în spațiu cu 3 dimensiuni, un poliedru convex cu n vârfuri, m muchii și f fețe. Deoarece acesta se poate reprezenta pe o sferă astfel încât două muchii să nu se intersecteze decât în extremități și efectuând o proiecție stereografică al cărei centru este mijlocul unei fețe se obține un graf planar.

Deci $n - m + f = 2$ pentru orice poliedru convex.

Corolar 1.5.1.3. Graful K_5 (complet cu 5 vârfuri) nu este planar.

Demonstrație:

Folosim metoda reducerii la absurd. Cum $n = 5$, $m = C_5^2 = 10$. Dacă K_5 ar fi planar atunci $f = 2 - n + m = 2 - 5 + 10 = 7$. Deoarece fiecare față are cel puțin 3 muchii în conturul său și fiecare muchie este în conturul a cel mult două fețe deducem $3f \leq 2m$. Adică $21 \leq 20$ ceea ce este o contradicție. \square

Corolar 1.5.1.4. Graful $K_{3,3}$ nu este planar.

Demonstrație:

Utilizăm de asemenei metoda reducerii la absurd. Presupunând că $K_{3,3}$ este planar, cum $n = 6$ și $m = 9$ rezultă $f = 2 - n + m = 2 - 6 + 9 = 5$. Graful $K_{3,3}$, fiind bipartit conform teoremei lui König și observației 1.3.4.1. b, are cicluri de lungime pară. Prin urmare fiecare contur are cel puțin patru muchii, adică $4f \leq 2m$. Deci $20 \leq 18$ contradicție. \square

Corolar 1.5.1.5. În orice graf planar conex cu n vârfuri $f \leq 2n - 4$ și $m \leq 3n - 6$.

Demonstrație:

Valorile maxime pentru f și m când n este constant se obțin pentru același graf deoarece valoarea $n - m + f$ este constantă. Acest graf este cel care are în frontieră fiecare fețe exact trei muchii, cum fiecare muchie este în exact două frontiere deducem $3f = 2m$. Folosind și relația lui Euler se obține:

$$f = 2n - 4 \text{ și } m = 3n - 6. \quad \square$$

Un rezultat important este *teorema lui Kuratowski*, de caracterizare a grafelor planare. Pentru această teoremă se definesc două tipuri speciale de grafe. Astfel un graf este de tipul 1 dacă se pot pune în evidență 6 vârfuri între care există lanțuri astfel încât considerând fiecare lanț ca și o muchie graful este $K_{3,3}$. Un graf este de tipul 2 dacă se pot pune în evidență 5 vârfuri cu proprietatea că între oricare două vârfuri există un lanț ce nu trece prin nici unul dintre celelalte trei. Grafele de tip 1 și de tip 2 sunt prezentate în figura 1.5.1.2.

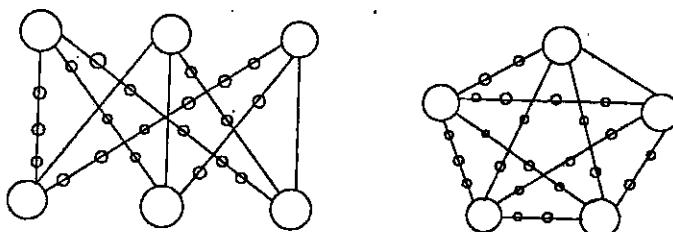


Fig. 1.5.1.2.

Teorema 1.5.1.2. Teorema lui Kuratowski

Condiția necesară și suficientă pentru ca un graf să fie planar este ca el să nu admită subgrafuri parțiale de tipul 1 sau de tipul 2.

Demonstrație:

Demonstrația acestei teoreme se află în [Ber69] p.224. O altă demonstrație ce folosește conceptul de pod într-un graf și care se bazează pe unele rezultate ale lui W.T. Tutte se găsește în [Vos91]. Rezultate privind planaritatea unor grafe demonstate cu ajutorul podurilor în grafe se află și în [ToS95]. \square

Un algoritm polinomial pentru testarea planarității unui graf a fost dat de Hopcroft și Tarjan în 1974 [HoT74]. Ca aplicații ale acestei probleme amintim reprezentarea diferitelor molecule sub forma unui graf planar, proiectarea circuitelor VLSI. Tot legată de grafele planare este și célébra conjectură a celor 4 culori, care are următorul enunț:

Regiunile oricărei hărți geografice se pot colora cu 4 culori astfel încât două regiuni adiacente să fie colorate diferit.

Această conjectură a fost demonstrată în 1976 de Appel și Haken [App76] prin construcția grafului dual topologic (geometric) G' al unui graf planar G . Pentru G' mulțimea vârfurilor corespunde fețelor grafului G și două vârfuri din G' se consideră adiacente dacă și numai dacă fețele corespunzătoare lor, în G , sunt adiacente.

1.5.2. Grafe perfecte

Mai întâi să precizăm câteva notări. Pentru un graf $G = (X, U)$ notăm cu:

- $\alpha(G)$ - numărul de stabilitate internă, adică numărul maxim de vârfuri două căte două neadiacente;
- $\chi(G)$ - numărul cromatic, adică numărul minim de culori cu care se pot colora vârfurile grafului astfel încât să nu fie colorate cu aceeași culoare două vârfuri adiacente;
- $\theta(G)$ - numărul clici maximale (subgrafe complete maximale) ale grafului G ;
- $\omega(G)$ - numărul maxim de vârfuri dintr-o clică a lui G .

Este evidentă inegalitatea $\chi(G) \geq \omega(G)$, deoarece oricare două vârfuri dintr-o clică sunt adiacente. Deci numărul cromatic este cel puțin $\omega(G)$.

Definiția 1.5.2.1. Un graf G se numește *χ -perfect* dacă și numai dacă are loc egalitatea $\chi(G(A)) = \omega(G(A))$ pentru orice submulțime de vârfuri A .

De asemenei $\alpha(G) \leq \theta(G)$, deoarece oricare două vârfuri dintr-o mulțime interior stabilă trebuie să facă parte din cíci diferite. Deci numărul maxim de vârfuri dintr-o mulțime interior stabilă este cel mult numărul maxim de cíci ale aceluia graf.

Definiția 1.5.2.2. Un graf G se numește *α -perfect* dacă și numai dacă $\alpha(G(A)) = \theta(G(A))$ pentru orice submulțime A de vârfuri.

În 1972 Lovász L. a demonstrat că cele două tipuri de grafe perfecte de fapt coincid.

Teorema 1.5.2.1. Un graf este *α -perfect* dacă și numai dacă este χ -perfect. [Lov72]

Astfel de grafe le numim simplu *grafe perfecte*. În 1971 Fulkerson, în [Ful71], a făcut o legătură între această problemă și proprietăile analitice ale optimizării liniare.

Vom prezenta în continuare câteva grafe perfecte și anume:

- grafele de comparabilitate;
- grafele triunghiulare;
- grafele asociate unei mulțimi de intervale din \mathbb{R} .

Definiția 1.5.2.3. Un graf $G = (X, U)$ neorientat se numește *graf de comparabilitate* dacă se pot orienta muchiile sale astfel încât graful $G' = (X, G)$ obținut să verifice proprietatea

$$\forall i \in X, \forall j \in \Gamma i \text{ și } \forall k \in \Gamma j \text{ atunci } k \in \Gamma i.$$

Considerând relația " $i \leq j$ " pentru i, j din X cu $j \in \Gamma i$ proprietatea de mai sus exprimă tranzitivitatea acestei relații. O familie de grafe de comparabilitate este cea a grafelor bipartite. În figura 1.5.2.1 sunt prezentate două grafe, unul este de comparabilitate și celălalt nu. Pentru cel de comparabilitate s-a indicat și orientarea muchiilor.

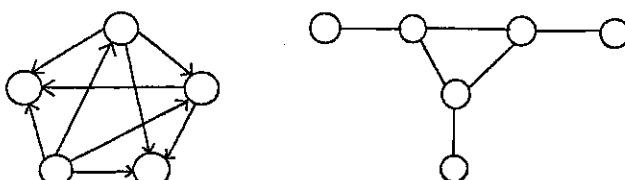


Fig.1.5.2.1.

Teorema 1.5.2.2. Un graf simplu $G = (X, U)$ este un graf de comparabilitate dacă și numai dacă pentru orice ciclu de lungime impară $\mu = \{i_1, i_2, \dots, i_{2k} = i_1\}$ are loc

$$(i_j, i_{j+2}) \in U \text{ pentru } 1 \leq j \leq 2k - 2. \text{ [Ber70]}$$

Definiția 1.5.2.4. Numim *coardă* pentru un ciclu elementar orice muchie a grafului care nu face parte din ciclu dar ciclul trece prin extremitățile muchiei.

Definiția 1.5.2.5. Un graf se numește *graf triangulat* dacă și numai dacă toate ciclurile de lungime cel puțin patru admit o coardă.

Definiția 1.5.2.6. Fiind dată o familie de intervale din \mathbb{R} , $\mathcal{I} = \{I_1, I_2, \dots, I_n\}$, numim *graf asociat familiei* \mathcal{I} sau *graful intervalelor* graful ce are n vârfuri asociate către unul pentru fiecare interval și între două vârfuri există muchie dacă intervalele corespunzătoare lor nu sunt disjuncte.

În figura 1.5.2.2 este dat un exemplu de graf de intervale.

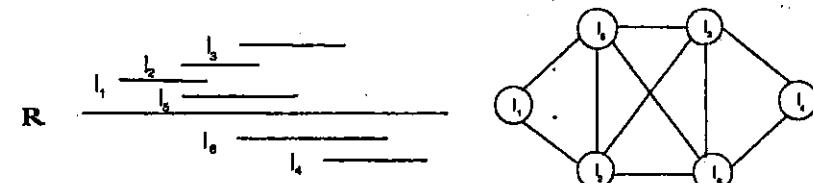


Fig. 1.5.2.2.

Teorema 1.5.2.3. Un graf G este un graf de intervale dacă și numai dacă

- G este un graf triangulat;
- Complementarul său este un graf de comparabilitate. [GiH64]

2. SPAȚII LINIARE ASOCIAȚE GRAFELOR

2.1. Spațiul liniar al părților unei mulțimi

Așa după cum se știe mulțimea valorilor booleene $B_2 = \{0, 1\}$ (sau cifrele sistemului de numerație în baza 2) are structura algebrică de corp în raport cu operațiile “+” și “.” definite de tablele următoare:

+	0	1
0	0	1
1	1	0

.	0	1
0	0	0
1	0	1

Observăm că operația “+” este “sau exclusiv”, iar “.” este “conjuncția” (respectiv adunarea fără transfer și înmulțirea în baza 2).

Folosind ca mulțime suport (tip de bază) pe B_2 se poate defini spațiul vectorial V format din mulțimea tablourilor unidimensionale (array-uri) cu n componente, $n \in \mathbb{N}$ (tipul de index este $1 \dots n$). Mulțimea V are 2^n elemente și ca spațiu liniar are dimensiunea n . În acest spațiu “+” se extinde ca operația de adunare a vectorilor (însumarea vectorilor se face pe componente), iar “.” este folosită la definirea operației de înmulțire cu scalarii lui B_2 (se înmulțește fiecare componentă cu respectivul scalar). Ca problemă de rezolvat formulăm:

Să se determine numărul bazelor spațiului vectorial V .

Pornind de la acest spațiu vectorial se poate defini și studia spațiul vectorial al submulțimilor unei mulțimi X cu n elemente date, notat și 2^X sau $P(X)$. Astfel, elementele lui X se consideră odonate și pentru că nu ne interesează natura lor se poate presupune că $X = \{1, 2, \dots, n\}$. Fiecarei submulțimi $A \subset X$ i se poate asocia vectorul sau funcția caracteristică (a_1, a_2, \dots, a_n) , unde $a_k = 1$ sau 0 după cum $k \in A$ respectiv $k \notin A$. Această corespondență (bijecție) permite să demonstreze că 2^X este izomorf cu spațiul vectorial V amintit mai sus. Prin urmare 2^X este și el un spațiu vectorial de dimensiune n , vectorii acestui spațiu vectorial sunt submulțimile lui X .

Un alt mod de organizare a lui 2^X ca spațiu vectorial peste corpul scalarilor B_2 îl descriem în continuare. Acest mod este numai formal diferit de spațiul vectorial induș de corespondență (izomorfismul) de mai sus.

Definim adunarea vectorilor (submulțimilor) ca fiind diferența simetrică, adică:

$$\forall A, B \in 2^X \text{ definim } A + B = (A \cup B) - (A \cap B) \Leftrightarrow \\ \Leftrightarrow A + B = (A - B) \cup (B - A).$$

Din $A, B \subset X$ și definițiile reuniunii, intersecției și diferenței mulțimilor rezultă că $A + B \subset X$, adică “+” este o lege de compoziție internă în mulțimea 2^X (sau 2^X este închisă în raport cu adunarea vectorilor). Înmulțirea cu scalari se definește cu ajutorul relațiilor:

$$0A = \emptyset \text{ și } 1A = A, \quad \forall A \in 2^X.$$

În continuare vom verifica axiomele necesare adunării vectorilor și înmulțirii cu scalari pentru ca 2^X să fie spațiu vectorial.

Din proprietățile reuniunii, intersecției și diferenței rezultă că “+” este asociativă și comutativă (aceste proprietăți se demonstrează în algebra de clasa a IX-a).

Pentru orice $A \subset X$

$$A + \emptyset = (A - \emptyset) \cup (\emptyset - A) = A;$$

$$A + A = (A - A) \cup (A - A) = \emptyset.$$

Deci pentru “+”, care este comutativă, deducem că elementul neutru este \emptyset și fiecare element este simetricul său.

Pentru a demonstra proprietățile înmulțirii cu scalari să considerăm λ și μ din B_2 și $A, B \in 2^X$. Cum

$$0(A + B) = \emptyset = \emptyset + \emptyset = 0A + 0B$$

și

$$1(A + B) = A + B = 1A + 1B$$

rezultă

$$\lambda(A + B) = \lambda A + \lambda B.$$

De asemenea

$$(0 + 0)A = 0A = \emptyset = \emptyset + \emptyset = 0A + 0A,$$

$$(0 + 1)A = 1A = A = \emptyset + A = 0A + 1A,$$

$$(1 + 0)A = 1A = A = A + \emptyset = 1A + 0A,$$

$$(1 + 1)A = 0A = \emptyset = A + A = 1A + 1A,$$

adică

$$(\lambda + \mu)A = \lambda A + \mu A.$$

În final

$$(0\mu)A = 0A = \emptyset = 0(\mu A)$$

$$(1\mu)A = \mu A = 1(\mu A),$$

adică

$$(\lambda\mu)A = \lambda(\mu A).$$

Din cele de mai sus rezultă că mulțimea tuturor submulțimilor unei mulțimi finite X , notată și cu 2^X , este un spațiu vectorial peste corpul B_2 .

Observăm că pentru orice $A \subset X$ are loc

$$A = \sum_{x \in A} \{x\}$$

Deci, singleoanele lui X (cele n submulțimi ale lui X de cardinal 1, $\{i\}$, $i \in X$) sunt generatori ai spațiului vectorial 2^X .

În plus

$$\sum_{i=1}^n \lambda_i \{i\} = \Phi \Leftrightarrow \lambda_1 = \lambda_2 = \dots = \lambda_n = 0$$

Adică, singleoanele sunt și vectori liniari independenți în spațiu vectorial 2^X . Deci spațiu vectorial 2^X este de dimensiune $n = |X|$ și o bază a acestuia este mulțimea singleoanelor lui X , adică $\{\{i\} \mid i \in X\}$.

Fiind dat graful $G = (X, U)$ pentru că X și U sunt mulțimi deducem, din cele de mai sus, că 2^X și 2^U au structura algebrică de spațiu vectorial pe care le putem considera spații vectoriale atașate grafului G . Câte o bază a acestor spații vectoriale constă în mulțimea tuturor submulțimilor formate dintr-un singur vârf X respectiv dintr-o singură muchie a lui U . Spațiu vectorial 2^U are mai multe subspații dintre care cele mai interesante, pe care le prezentăm în continuare, sunt subspațiul generat de ciclurile simple ale lui G sau cel generat de cociclurile lui U . Un alt subspațiu al lui 2^U pe care doar îl amintim este cel al cuplajelor grafului G .

2.2. Subspațiul ciclurilor

Fiind dat graful $G = (X, U)$, vectorii spațiului vectorial 2^U sunt mulțimi de muchii. Deci în fiecare vector, o muchie oarecare a lui G poate să facă sau nu parte. Vom considera mai întâi submulțimile lui U care sunt cicluri în G .

ACESTE cicluri sunt deci cicluri simple dar pot fi neelementare (ciclurile conțin o mulție cel mult o dată, dar pot trece de mai multe ori printr-un vârf). Subspațiu generat de acești vectori îl numim **subspațiuul ciclurilor grafului G** . Vectorii generați de aceste mulțimi de muchii (vectori) sunt mulțimi de muchii care conțin muchiile unui ciclu simplu sau reuniune de cicluri simple muchie disjuncte (ciclurile nu au muchii comune). Să observăm că un ciclu simplu și neelementar se poate interpreta ca fiind reuniunea ciclurilor elementare și simple care compun acel ciclu. Pentru o imagine geometrică să ne imaginăm că graful este reprezentat geometric astfel încât fiecare ciclu elementar minimal este reprezentat de o curbă închisă (pe curba sunt vârfurile și muchiile ciclului respectiv). Două asemenea cicluri pot să: nu aibă în comun nimic sau să aibă în comun numai vârfuri sau să aibă în comun și vârfuri și muchii (lanțuri). Două cicluri care au vârfuri în comun le reprezentăm ca să fie tangente în punctele asociat vârfurilor comune. Două cicluri elementare care au muchii și vârfuri comune vor avea în reprezentarea geometrică porțiuni de curbă comune. Asemenea situații sunt prezentate în figura 2.2.1.

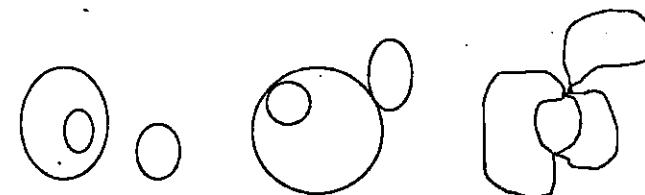


Fig.2.2.1.

Suma a doi asemenea vectori este tot o reuniune de cicluri elementare muchie disjuncte, care constituie fie un ciclu simplu fie o reuniune de cicluri simple.

Teorema 2.2.1. *Mulțimea formată din ciclurile simple și reuniunile de cicluri simple muchie disjuncte, pe care o notăm cu C , este subspațiu vectorial al spațiului vectorial 2^U .*

Demonstrație:

Să constatăm mai întâi că \emptyset este element al acestui subspațiu fiindcă

$$\Phi = \bigcup_{i \in \Phi} A_i$$

Pentru a arăta că C este mulțime închisă în raport cu “+” reamintim că un graf oarecare (o mulțime de muchii împreună cu extremitățile lor ca mulțime de vârfuri sau X) în care toate vâfurile au grad par este o reuniune de cicluri simple (vezi demonstrația teoremei 1.4.1.4). Fie deci, A, B două reuniuni de cicluri simple ale lui G și p un vârf al grafului. Fiindcă “+” este comutativă sunt posibile și analizăm doar trei situații:

- p nu este în nici un ciclu simplu dintre cele ale lui A sau B (nici o muchie din A sau B nu este incidentă lui p);
- p face parte numai din cicluri ale lui A (un număr par nenul de muchii ale lui A sunt incidente lui p și nici o muchie din B nu este incidentă lui p);
- p face parte și din cicluri ale lui A și din cicluri ale lui B (câte un număr par nenul de muchii din A și din B sunt incidente lui p , notăm aceste numere cu n_A respectiv n_B).

În cazul a vârful p în $A + B$ are gradul 0, deci par. În cazul b vârful p în $A + B$ are gradul pe care îl are și în A , deci par. În cazul c fiindcă $A + B = (A \cup B) - (A \cap B)$ și ciclurile din A și B sunt simple deducem că numărul muchiilor incidente lui p din $A + B$ este $n_A + n_B - 2|A \cap B|$ și deci p are gradul par în $A + B$.

Prin urmare în $G(A + B)$ toate vâfurile au gradele pare și deci $A + B$ este o reuniune de cicluri simple muchie disjuncte ale lui G . \square

Teorema 2.2.2. Fie $G = (X, U)$ un graf de ordin, n de dimensiune m și cu p componente conexe. Fie $P = (X, T)$ o pădure maximală a lui G , pentru $u \in U - T$ notăm cu μ^u unicul ciclu conținut de $T \cup \{u\}$. Ciclurile μ^u , considerate pentru fiecare $u \in U - T$, formează o bază de cicluri pentru spațiul vectorial al mulțimii formată din ciclurile simple grafului G sau reuniuni de cicluri simple muchie disjuncte.

Demonstrație:

Fie u și $v \in U - T$ cu $u \neq v$ atunci $v \notin \mu^u$ și $u \notin \mu^v$ de aici deducem că:

$$\sum_{u \in U - T} \lambda_u \mu^u = 0 \Leftrightarrow \lambda_u = 0, \quad \forall u \in U - T$$

Ceea ce înseamnă că $C_i = \{\mu^u \mid u \in U - T\}$ este o mulțime de cicluri simple și independente (privite ca submulțimi ale lui U).

Să demonstrăm că C_i este un sistem de generatori. Fie A o mulțime de muchii ale lui G care este reuniunea unor cicluri simple muchie disjuncte.

Alegem scalarii $\lambda_u = 1$ pentru fiecare $u \in A - T$ și $\lambda_u = 0$ pentru fiecare $u \in U - (A - T)$ (în rest). Cu această alegere are loc:

$$B = A + \sum_{u \in U - T} \lambda_u \mu^u \subset T$$

Pe de altă parte în $G(B)$ toate vâfurile au gradele pare (B este o reuniune de cicluri simple), cum singura submulțime din T cu această proprietate este \emptyset rezultă $B = \emptyset$. Pentru că simetricul unui element din 2^U este el însuși se deduce:

$$A = \sum_{u \in U - T} \lambda_u \mu^u$$

Ceea ce demonstrează că A este o combinație liniară (generată) de elementele lui C_i . \square

Din demonstrația făcută deducem că acest spațiu are dimensiunea $m - n + p$. Celălalt spațiu liniar al ciclurilor (prezentat în paragraful 1.4.2) este diferit de acesta nu numai prin faptul că el a fost definit pentru un graf orientat dar ciclurile puteau să nu fie simple și scalarii erau din \mathbb{Z} , dar dimensiunea sa este tot $m - n + p$.

2.3. Subspațiul cociclurilor

Fiind dat un graf $G = (X, U)$ și spațiul vectorial al părților lui U , 2^U vom considera mulțimea tuturor cociclurilor lui G împreună cu reuniunile de cocicluri muchie disjuncte, notată Co , despre care vom demonstra că este un subspațiu vectorial al lui 2^U .

Lema 2.3.1. Dacă T și V sunt două cocicluri, atunci $T + V \in Co$.

Demonstrație:

T și V fiind cocicluri înseamnă că există A_1 și $A_2 \subset X$ astfel încât $T = \omega(A_1)$ și $V = \omega(A_2)$. Să notăm cu B_1 și B_2 complementarele în X ale mulțimilor A_1 respectiv A_2 . Conform notărilor are loc $T = \omega(B_1)$ și $V = \omega(B_2)$. Pentru a introduce în notație atât A_1 și A_2 cât și complementarele lor vom nota pe T și V cu $[A_1, B_1]$ respectiv cu $[A_2, B_2]$. Are loc

$$A_1 \cup B_1 = X = A_2 \cup B_2$$

Prin extensie notația $[M, N]$ desemnează mulțimea muchiilor lui G care au o extremitate în M și cealaltă extremitate în N , unde $M, N \subset X$ fără a se cere că ele să fie complementare în X . De aici $[M, N] = [N, M]$.

Vom exprima, mai întâi, în spațiul vectorial 2^X mulțimile $A_1 + A_2$ și $B_1 + B_2$ numai cu ajutorul operațiilor de reuniune și intersecție a mulțimilor și operanții fiind dintre cele patru submulțimi ale lui X .

$$\begin{aligned} A_1 + A_2 &= (A_1 - A_2) \cup (A_2 - A_1) = (A_1 \cap B_2) \cup (A_2 \cap B_1) = \\ &= (A_1 \cap B_2) \cup (B_1 \cap A_2) \end{aligned}$$

Prin analogie sau direct obținem:

$$B_1 + B_2 = A_2 + B_1 = (A_2 \cap A_1) \cup (B_1 \cap B_2) = (A_1 \cap A_2) \cup (B_1 \cap B_2)$$

Observăm astfel că $(A_1 + A_2) \cup (B_1 + B_2) = X$, adică $A_1 + A_2$ și $B_1 + B_2$ sunt complementare în X (formează o partitie a lui X).

Conform definiției cociclurilor pentru orice M, N, S cu $M \cup N \cup S = X$ are loc:

$$\omega(M \cup N) = \omega(M) \cup \omega(N).$$

Dar și în cazul general pentru $M, N, S \subset X$ au loc

$$[M \cup N, S] = [M, S] \cup [N, S] \quad \text{și} \quad [M, N \cup S] = [M, N] \cup [M, S]$$

Folosind formulele de mai sus obținem:

$$\begin{aligned} T = [A_1, B_1] &= [(A_1 \cap A_2) \cup (A_1 \cap B_2), (B_1 \cap A_2) \cup (B_1 \cap B_2)] = \\ &= [A_1 \cap A_2, (B_1 \cap A_2) \cup (B_1 \cap B_2)] \cup \\ &\quad \cup [A_1 \cap B_2, (B_1 \cap A_2) \cup (B_1 \cap B_2)] = \\ &= [A_1 \cap A_2, B_1 \cap A_2] \cup [A_1 \cap A_2, B_1 \cap B_2] \cup \\ &\quad \cup [A_1 \cap B_2, B_1 \cap A_2] \cup [A_1 \cap B_2, B_1 \cap B_2] \end{aligned}$$

$$\begin{aligned} V = [A_2, B_2] &= [(A_1 \cap A_2) \cup (B_1 \cap A_2), (A_1 \cap B_2) \cup (B_1 \cap B_2)] = \\ &= [A_1 \cap A_2, A_1 \cap B_2] \cup [A_1 \cap A_2, B_1 \cap B_2] \cup \\ &\quad \cup [B_1 \cap A_2, A_1 \cap B_2] \cup [B_1 \cap A_2, B_1 \cap B_2] \end{aligned}$$

Expresii care permit calculul lui $T + V$ ca fiind:

$$\begin{aligned} T + V &= [A_1 \cap A_2, B_1 \cap A_2] \cup [A_1 \cap B_2, B_1 \cap B_2] \cup \\ &\quad \cup [A_1 \cap A_2, A_1 \cap B_2] \cup [B_1 \cap A_2, B_1 \cap B_2] = \\ &= [A_1 \cap A_2, B_1 \cap A_2] \cup [A_1 \cap A_2, A_1 \cap B_2] \cup \end{aligned}$$

$$\begin{aligned} &\cup [A_1 \cap B_2, B_1 \cap B_2] \cup [B_1 \cap A_2, B_1 \cap B_2] = \\ &= [A_1 \cap A_2, (B_1 \cap A_2) \cup (A_1 \cap B_2)] \cup \\ &\quad \cup [(A_1 \cap B_2) \cup (B_1 \cap A_2), B_1 \cap B_2] = \\ &= [A_1 \cap A_2, A_1 + A_2] \cup [A_1 + A_2, B_1 \cap B_2] = \\ &= [A_1 \cap A_2, A_1 + A_2] \cup [B_1 \cap B_2, A_1 + A_2] = \\ &= [(A_1 \cap A_2) \cup (B_1 \cap B_2), A_1 + A_2] = [B_1 + B_2, A_1 + A_2] = \\ &= [A_1 + A_2, B_1 + A_2]. \end{aligned}$$

Pentru că $A_1 + A_2$ și $B_1 + B_2$ este o partitie a lui X (s-a demonstrat mai sus), deducem că $T + V$ este un cociclu al lui G , adică $T + V \in Co$. \square

Teorema 2.3.1. *Mulțimea Co a tuturor cociclurilor împreună cu reuniunile de cocicluri muchie disjuncte este un subspațiu vectorial al lui 2^U .*

Demonstratie:

Demonstrația rezultă imediat în baza lemei 2.3.1 și a faptului că $\emptyset \in Co$ care se deduce din $T + T = \emptyset$ oricare ar fi $T \in Co$. Pentru orice graf $G = (X, U)$ mulțimea $Co \neq \emptyset$ pentru că fiecare submulțime a lui induce un cociclu. \square

Teorema 2.3.2. *Fie $G = (X, U)$ un graf de ordin, n de dimensiune m și cu p componente conexe. Fie $P = (X, T)$ o pădure maximală a lui G , pentru $u \in T$ notăm cu θ^u unicul cociclu obținut în componenta conexă a lui G în care este u după eliminarea lui u . Ciclurile θ^u , considerate pentru fiecare $u \in T$, formează o bază de cocicluri pentru subspațiul vectorial al mulțimii formată din cociclurile grafului G sau reuniuni de cocicluri muchie disjuncte.*

Demonstratie:

Fie u și $v \in T$ cu $u \neq v$ atunci $v \notin \theta^u$ și $u \notin \theta^v$ de aici deducem că:

$$\sum_{u \in T} \lambda_u \theta^u = 0 \Leftrightarrow \lambda_u = 0, \forall u \in T$$

Ceea ce înseamnă că $C = \{\theta^u \mid u \in T\}$ este o mulțime de cocicluri independente.

Să demonstrăm că C este un sistem de generatori. Fie A o mulțime de muchii ale lui G care este reuniunea unor cocicluri muchie disjuncte. Alegem

scalarii $\lambda_u = 1$ pentru fiecare $u \in A \cap T$ și $\lambda_u = 0$ pentru fiecare $u \in U - A$ (în rest). Cu această alegere are loc:

$$B = A + \sum_{u \in T} \lambda_u \theta^u \cap T = \Phi.$$

În baza lemei 2.3.1. rezultă că $B \in Co$, cum singurul cociclu care nu are nici o muchie din T este $\emptyset = [X, \emptyset]$ rezultă $B = \emptyset$. Pentru că simetricul unui element din 2^U este el însuși se deduce:

$$A = \sum_{u \in T} \lambda_u \theta^u$$

Ceea ce demonstrează că A este o combinație liniară (generată) de elementele lui C . \square

Din demonstrația făcută deducem că acest spațiu are dimensiunea $|T| = n - p$.

3. DRUMURI ÎN GRAFE

Problemele drumurilor în grafe sunt cele mai vechi în teoria grafelor. Pe lângă problema podurilor din Königsberg mai amintim problemele de traversare a râului și anume: problema trecerii lupului, caprei și a verzei de către un barcaj cu o barcă și cea a traversării a trei perechi de soț-soție a unui râu cu o barcă fără barcaj astfel încât să nu rămână pe un mal o anumită configurație. Alte probleme care se rezolvă ca probleme de drumuri în grafe sunt:

- problema programării unor turnee, sportive sau nu;
- probleme de proiectare sau realizare a unor investiții;
- probleme de gestiunea stocurilor;
- probleme de optimizării în rețele;
- probleme de inteligență artificială și recunoașterea formelor;
- probleme de tratare numerică a semnalelor, a codificării și decodificării informației;
- probleme militare, etc.

Problemele care se pun în legătură cu drumurile în grafe sunt:

- verificarea existenței unui drum de lungime dată;
- determinarea drumurilor de lungime dată;
- optimizări pe mulțimea drumurilor.

Verificarea existenței drumurilor de lungime dată o putem considera rezolvată dacă avem în vedere teorema 1.2.2.1.

Probleme de optimizări pe mulțimea drumurilor se pun pentru grafe $G = (X, U)$ în care se asociază fiecărui arc $u \in U$ un număr $l(u) \in \mathbb{R}$. Semnificația lui $l(u)$ poate fi:

- distanța dintre extremitățile arcului u ;
- durata trecerii de la extremitatea inițială la extremitatea terminală a arcului;
- costul acestei treceri;
- probabilitatea de trecere pe acel arc;
- capacitatea arcului respectiv, etc.

Numărul $l(u)$ îl numim *valoarea arcului u* . Valoarea unui drum μ , notată $l(\mu)$, se poate defini în mai multe moduri. De exemplu valoarea unui drum poate fi:

$$l(\mu) = \sum_{u \in \mu} l(u) \text{ sau } l(\mu) = \prod_{u \in \mu} l(u) \text{ sau } l(\mu) = \min(\max)\{l(u) \mid u \in \mu\}.$$

Problemele de optim se referă la determinarea valorilor minime sau maxime a drumurilor dintr-o mulțime de drumuri, eventual cere și precizarea unui drum din mulțime care să aibă acea valoare. Această mulțime de drumuri poate fi cea a drumurilor care leagă două vârfuri date. Majoritatea algoritmilor determină către o valoare optimă pentru mulțimile de drumuri care pornesc dintr-un vârf dat către fiecare dintre celelalte vârfuri. Există și algoritmi care determină valori optime pentru drumurile care conectează fiecare pereche de vârfuri. Deci pentru un graf de ordin n aceste probleme cer să se determine o valoare, n valori respectiv n^2 valori de drumuri și eventual către un drum pentru fiecare valoare aflată. Primele două probleme de optim se rezolvă cu aceiași algoritmi. Pentru cea de a treia problemă algoritmii se mai numesc și *algoritmi matriceali*, deoarece cu ajutorul lor se determină o matrice, de ordin n , de valori.

Exemplul 3.1.

Considerăm o hartă turistică. Se pună problema de a determina un traseu de lungime minimă între două puncte date de pe hartă.

Această problemă revine la a determina drumul de valoare minimă în graful atașat hărții astfel: fiecărei localități din zona reprezentată în hartă sau intersecțiilor de căi de comunicație îi atașăm un vârf. Între două vârfuri există muchie dacă pe hartă există o cale de comunicație directă între corespondentele celor două vârfuri. Fiecarei muchii îi atașăm ca valoare lungimea căii de acces dintre cele două puncte de pe hartă. A determina un traseu de valoare (lungime) minimă între două puncte de pe hartă este echivalent cu a determina un drum în graful atașat hărții de valoare minimă.

Într-un mod analog se poate determina traseul cel mai rapid sau de cost minim doar prin schimbarea valorilor muchiilor grafului.

Un tip similar de probleme cu acesta sunt unele dintre problemele de rutare în rețele de calculatoare.

Exemplul 3.2.

Trebuie realizată o conectare între două puncte dintr-un anume teritoriu.

Această conectare se face în scopul transferării unor mesaje (telefonice, video) sau a unor materiale (apă, curent electric, mașini, etc.). Pentru realizarea acestei conectări există mai multe variante (trasee posibile). Fiecare variantă se compune din mai multe tronsoane cărora li se asociază valori (limite de capacitate, costuri de realizare etc.). Problema cere să se stabilească drumul de valoare optimă, adică cost minim sau capacitate (beneficiu) maximă. Acestei probleme i se

poate atașa un graf (arcile corespunzând tronsoanelor). În graf există mai multe drumuri între cele două vârfuri între care trebuie construită calea de comunicație. Problema care se pune este de a alege între aceste variante pe cea optimă. Deci problema revine la a determina un drum optim în acest graf.

O situație similară este în teoria jocurilor sau strategii militare când despre intențiile partenerului (adversarului) avem informații parțiale și valorile arcelor sunt probabilități cu care partenerul nostru va trece dintr-o stare în cealaltă (stările fiind vârfuri ale grafului ce modelează problema sau jocul).

Dorim să accentuăm că prin *valoarea unui drum* se înțelege un număr care poate avea diferite semnificații concrete (lungime, cost, durată, beneficiu, probabilitate, etc.), dar nu distanță euclidiană. De asemenea precizăm că valoarea și lungimea unui drum sunt în general diferite. Deci, utilizarea termenilor trebuie făcută cu grijă.

3.1. Optimizări de drumuri în ipoteza $I(\mu) = \sum_{u \in \mu} l(u)$

Algoritmii de rezolvare a acestor probleme se pot baza pe unele proprietăți ale valorilor arcelor și/sau ale grafelor. Dintre aceste proprietăți amintim: pozitivitatea valorilor arcelor, $l(u) = 1, \forall u \in U$, graful este fără circuite. Dacă $l(u) = 1$ algoritmii vor determina lungimile minime (maxime) ale drumurilor. Pentru a da condițiile de existență a valorii minime a drumurilor de la un vârf i la un vârf j să considerăm că între aceste drumuri există cel puțin un drum μ care conține un circuit ω . Situația este ilustrată în figura 3.1.1, în care s-au reprezentat doar segmente ale drumului.

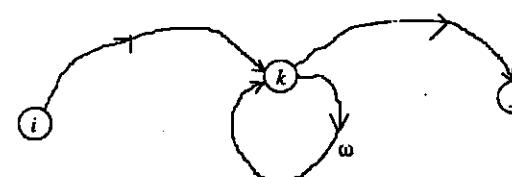


Fig. 3.1.1.

Notăm cu μ' drumul de la i la j care nu conține circuitul ω . Deci are loc $I(\mu) = I(\mu') + I(\omega)$.

Dacă $I(\omega) < 0$ atunci nu există valoare minimă a drumurilor de la i la j deoarece circuitul ω se poate parcurge de oricăte ori și valoarea acestor drumuri tindă la $-\infty$.

Dacă $l(\omega) \geq 0$ atunci $l(\mu') \leq l(\mu)$ și deci în căutarea valorii minime ne putem restrângă la multimea drumurilor elementare de la i la j . Deci se poate considera că grafele sunt simple și nu conțin circuite de valoare negativă, când se caută valoarea minimă a drumurilor, respectiv circuite de valoare pozitivă când se caută valoarea maximă a drumurilor. Deoarece multimea drumurilor elementare, într-un graf finit, este finită rezultă că problema determinării valorii minime a drumurilor în condițiile precizate are soluție.

3.1.1. Valori minime de drumuri de la un vîrf dat.

Fie $G = (X, U)$ cu $X = \{1, 2, \dots, n\}$. Pentru arcul $u = (i, j) \in U$ notăm cu $l_{ij} = l(u)$ valoarea sa. Pentru vîrful $i \in X$ notăm cu λ_i^* valoarea minimă a drumurilor de la vîrful 1 la vîrful i .

Algoritm 3.1.1.1. (Moore-Dijkstra)

Determină valorile minime ale drumurilor de la vîrful 1 la orice alt vîrf într-un graf cu valori pozitive ale arcelor.

(a) inițializări

$$\begin{aligned}\lambda_1 &:= 0, \\ \lambda_i &:= \begin{cases} l_{1i}, & i \in \Gamma_1, \\ \infty, & \text{altele} \end{cases}, \\ S &:= \{2, 3, \dots, n\}.\end{aligned}$$

(b) testul de stop

Fie $j \in S$ astfel încât $\lambda_j = \min \{\lambda_i \mid i \in S\}$;
 $S := S - \{j\}$;
Dacă $S = \emptyset$ atunci stop sfârșit;

(c) iterarea de bază

Pentru $i \in \Gamma_j \cap S$ execută $\lambda_i = \min(\lambda_i, \lambda_j + l_{ji})$ sfârșit;
goto (b).

După cum se poate constata algoritmul are nevoie de $n - 1$ pași. Convergența sa este asigurată de următoarea propoziție.

Lema 3.1.1.1. Fie $j \in S$ astfel încât $\lambda_j = \min \{\lambda_i \mid i \in S\}$ atunci $\lambda_j^* = \lambda_j$.

Demonstrație:

În mod evident există un drum de la 1 la j de valoare λ_j . Fie acum un drum μ , oarecare, de la 1 la j . Împărțim acest drum în două părți astfel:

- μ_1 este drumul de la vîrful 1 la primul vîrf din S întâlnit în μ , notat h ;
- μ_2 partea din μ de la h la j .

Atunci:

$$l(\mu_1) \geq \lambda_h \geq \lambda_j \quad \text{și} \quad l(\mu) = l(\mu_1) + l(\mu_2) \geq l(\mu_1) \geq \lambda_j$$

Deci

$$\lambda_j^* = \lambda_j \quad \square$$

Să remarcăm că în algoritm valorile λ cresc sau rămân nemodificate pentru că valorile arcelor sunt pozitive. Deci dacă ar exista drumuri la vîrful j care trec prin h acestea au valori mai mari decât valoarea.

Exemplul 3.1.1.2.

Algoritmul lui Moore-Dijkstra aplicat pentru graful din figura 3.1.1.1 conduce la următoarele valori ale variabilelor din algoritm:

(a) $S = \{2, 3, 4, 5, 6\}$, $\lambda_1 = 0$, $\lambda_2 = 3$, $\lambda_3 = 10$, $\lambda_4 = 12$, $\lambda_5 = \lambda_6 = \infty$;

(b) $j = 2$, $S = \{3, 4, 5, 6\}$;

(c) $\Gamma_2 \cap S = \{3, 5\}$, $\lambda_3 = 9$, $\lambda_5 = 5$;

(b) $j = 5$, $S = \{3, 4, 6\}$;

(c) $\Gamma_5 \cap S = \{3, 4, 6\}$, $\lambda_3 = 8$, $\lambda_4 = 8$, $\lambda_6 = 11$;

(b) $j = 4$, $S = \{3, 6\}$;

(c) $\Gamma_4 \cap S = \{6\}$, $\lambda_6 = 10$;

(b) $j = 3$, $S = \{6\}$;

(c) $\Gamma_3 \cap S = \emptyset$;

(b) $j = 6$, $S = \emptyset$ stop.

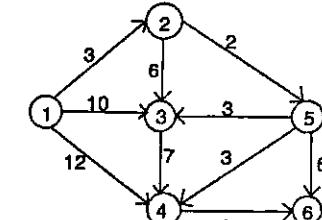


Fig. 3.1.1.1.

Deci valorile minime ale drumurilor de la vîrful 1 sunt $\lambda_1 = 0$, $\lambda_2 = 3$, $\lambda_3 = 8$, $\lambda_4 = 8$, $\lambda_5 = 5$, $\lambda_6 = 10$.

Algoritmul 3.1.1.1 se poate aplica și pentru grafele neorientate dacă se consideră fiecare muchie ca fiind înlăturată de două arce între extremitățile muchiei cu sensuri contrare. În cazul că vrem să aflăm lungimile minime ale drumurilor de la vîrful 1 la celelalte, adică $l(u) = 1$ pentru orice arc u , algoritmul 3.1.1.1 poate fi îmbunătățit și arată astfel:

Algoritmul 3.1.1.2.

Determină lungimile minime ale drumurilor de la vârful 1 la celelalte vârfuri ale grafului.

(a) inițializări

$$\lambda_1 := \begin{cases} 0, & i = 1 \\ \infty, & i \geq 2 \end{cases}$$

$$k := 0; R := \{1\}; S := \{1\};$$

(b) iterarea de bază

pentru $i \in \Gamma_j \cap (X - R)$ execută $\lambda_i := k + 1$ sfp;
 $S := \Gamma_j S;$
 $R := R \cup S;$

(c) criteriu de oprire

dacă $R = S$ atunci stop
 altfel $k := k + 1;$
 goto (b);

sfd.

Observația 3.1.1.1.

- Testul $i \in X - R$, din algoritmul 3.1.1.2, este echivalent cu testul $\lambda_i > k$.
- Algoritmii 3.1.1.1 și 3.1.1.2 se pot modifica astfel încât să construiască o arborescență de rădăcină 1 care să conțină pentru fiecare vârf i un drum de valoare λ_i . Cum orice arborescență este definită de un vector p cu $p(i) = s$ dacă s este predecesor (vârful părinte) al vârfului i . Actualizarea valorilor vectorului p se poate face prin adăugarea instrucțiunii $p(i) = j$ după modificarea lui λ_i în iterată (c) respectiv (b) a algoritmilor 3.1.1.1 și 3.1.1.2. La algoritmul 3.1.1.2 vârful j poate fi orice vârf din S pentru care $i \in \Gamma_j \cap (X - R)$.

În cazul general, adică graful G și valorile arcelor $l(u)$ sunt oarecare se poate aplica algoritmul 3.1.1.3.

Algoritmul 3.1.1.3. [Moo57]

Determină valorile minime ale drumurilor de la vârful 1 la celelalte vârfuri într-un graf oarecare.

(a) inițializări

$$S := \{2, 3, \dots, n\}; \lambda_1 := 0;$$

$$\lambda_i := \begin{cases} l_{1i}, & i \in \Gamma_1 \\ \infty, & \text{altfel} \end{cases}, i \geq 2$$

(b) iterarea de bază

Fie $j \in S$ astfel încât $\lambda_j = \min\{\lambda_i \mid i \in S\}$;
 $S := S - \{j\}$
 pentru $i \in \Gamma_j$ execută
 dacă $\lambda_j + l_{ij} < \lambda_i$ atunci $\lambda_i := \lambda_j + l_{ij}$;
 $S := S \cup \{i\}$;
 sfđ;

(c) criteriu de stop

dacă $S = \emptyset$ atunci stop altfel goto (b) sfđ.

Observația 3.1.1.2.

Alegerea lui j așa cum se dă ca în etapa (b) a algoritmului 3.1.1.3 nu este necesară pentru convergența algoritmului, rezultatele ar fi aceleași și dacă se alege pentru j o valoare oarecare din S . Totuși modul de alegere al valorii lui j influențează numărul de iterații necesare algoritmului pentru a ajunge la valorile optime.

În continuare vrem să prezentăm *algoritmul lui Bellman-Kalaba*. Acest algoritm se bazează pe principiul optimalității al optimizării dinamice enunțat de Bellman astfel:

O politică este optimală dacă la fiecare moment, oricare ar fi deciziile precedente, deciziile care urmează a fi luate constituie o politică optimală în raport cu rezultatul deciziilor anterioare.

Dacă se notează cu λ_i valoarea minimă a drumurilor de la vârful 1 la vârful i , $\forall i \in X$ și considerând că

$$\Gamma_j^{-1} = \{k_1, k_2, \dots, k_p\},$$

atunci conform principiului lui Bellman trebuie ca

$$\lambda_j = \min(\lambda_{k_1} + l_{k_1 j}, \lambda_{k_2} + l_{k_2 j}, \dots, \lambda_{k_p} + l_{k_p j}) = \min(\lambda_i + l_{ij} \mid i \in \Gamma_j^{-1}).$$

Dacă se definește matricea valorilor arcelor $V = (v_{ij})_{i,j \in X}$ astfel:

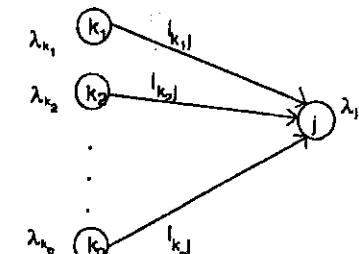


Fig. 3.1.1.2.

$$v_{ij} = \begin{cases} 0, & i = j \\ l_{ij}, & (i,j) \in U \\ \infty, & \text{altele} \end{cases}$$

Prin urmare valorile $\lambda_i, i \in X$, trebuie să verifice relațiile:

$$\begin{cases} \lambda_1 = 0 \\ \lambda_j = \min(\lambda_i + v_{ij} | i=1,n), j \geq 2 \end{cases}$$

Acest sistem de ecuații îl putem rezolva bazându-ne pe principiul lui Bellman folosind un procedeu iterativ. Metoda de rezolvare este cunoscută și cu numele de **algoritmul lui Bellman-Kalaba**.

Algoritmul 3.1.1.4. [Bel58]. Algoritmul lui Bellman-Kalaba

Determină valorile minime ale drumurilor de la vârful 1 la orice alt vârf într-un graf oarecare sau detectează existența unui circuit de valoare negativă.

(a) inițializări

$$\begin{cases} \lambda_1^0 = 0 \\ \lambda_j^0 = v_{1j}, j \geq 2 \\ k=1; \end{cases}$$

(b) iterată de bază

$$\begin{cases} \lambda_1^{k+1} = 0 \\ \lambda_j^{k+1} = \min(\lambda_i^k + v_{ij} | i=1,n), j \geq 2 \\ k=k+1; \end{cases}$$

(c) criteriul de stop

dacă $\lambda_j^{k+1} = \lambda_j^k$ pentru $j=1,n$ atunci stop sfid;
 dacă $k \leq m$ atunci goto (b)
 altfel tipărește "există circuit de valoare negativă";
 stop;
 sfid.

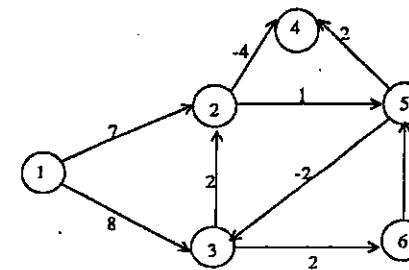


Fig. 3.1.1.3.

Exemplul 3.1.1.3.

Vom aplica algoritmul lui Bellman-Kalaba pentru graful din figura 3.1.1.3.

Matricea V a valorilor arcelor este:

$$\left(\begin{array}{cccccc} 0 & 7 & 8 & \infty & \infty & \infty \\ \infty & 0 & \infty & -4 & 1 & \infty \\ \infty & 2 & 0 & \infty & \infty & 2 \\ \infty & \infty & \infty & 0 & \infty & \infty \\ \infty & \infty & -2 & 2 & 0 & \infty \\ \infty & \infty & \infty & \infty & 3 & 0 \end{array} \right)$$

La sfârșitul aplicării algoritmului matricea valorilor λ_i^k , calculate de algoritm, este cea din tabloul de mai jos:

k	λ_1^k	λ_2^k	λ_3^k	λ_4^k	λ_5^k	λ_6^k
1	0	7	∞	∞	∞	∞
2	0	7	8	3	8	∞
3	0	7	6	3	8	10
4	0	7	6	3	8	8
5	0	7	6	3	8	8

Valorile de pe ultima linie (sau penultima) a tabloului λ_i^k sunt valorile minime ale drumurilor de la vârful 1 la vârful i .

Observația 3.1.1.3.

În aplicarea algoritmului lui Bellman-Kalaba pentru a determina elementul λ_i^{k+1} din tablou se procedează astfel: se adună linia k din tablou cu coloana i din matricea valorilor arcelor apoi se determină valoarea minimă a rezultatelor. Valoarea minimă se trece în tablou pe poziția λ_i^{k+1} .

Se observă ușor că numărul λ_i^k reprezintă valoarea minimă a drumurilor de la vârful 1 la vârful $i \in X$ care au lungimea cel mult k , adică valoarea minimă a drumurilor care trec prin cel mult $k + 1$ vârfuri. Dacă nu există circuite de valoare negativă atunci cum drumul de valoare minimă este elementar înseamnă că el trece prin cel mult n vârfuri și deci valoarea minimă se află în cel mult $n - 1$ ieratăii.

Dacă suntem siguri că graful nu conține circuite de valoare negativă, atunci algoritmul lui Bellman-Kalaba se îmbunătățește dacă în etapa (b) j parcurge succesiv valorile $2, 3, \dots, n$ în această ordine și calculele se fac cu formula:

$$\lambda_i^{k+1} := \min \{ \min \{ \lambda_i^{k+1} + v_{ij} \mid i < j \}, \min \{ \lambda_i^k + v_{ij} \mid i > j \} \}.$$

Dacă aceste calcule se fac într-un tablou unidimensional $\lambda_i, i = 1, n$ în locul unui tablou bidimensional $\lambda_i^k, i = 1, n$ și $k = 1, n - 1$ atunci algoritmul obținut se numește *algoritmul lui Ford* [For56].

Algoritmul 3.1.1.5. Algoritmul lui Ford

Determină valorile minime ale drumurilor de la vârful 1 la oricare alt vârf într-un graf fără circuite de valoare negativă.

(a) inițializări

$$\lambda_1 := \begin{cases} 0, & i = 1 \\ v_{1j}, & i \geq 2 \end{cases}$$

(b) ieratăia de bază

pentru $j=2,n$ execută
 $\lambda_j := \min \{ \lambda_i + l_{ij} \mid i \in \Gamma_j^{-1} \}.$
 sfps;

(c) criteriu de stop

repetă
 (b)
 până când λ nu-și schimbă nici o valoare
 stop.

Când rezolvăm pe hârtie sau la tablă (manual) probleme cu algoritmul lui Ford valorile λ se scriu lângă reprezentarea corespunzătoare a vârfului respectiv și

la fiecare modificare de valoare se taie vechea valoare și se scrie noua valoare. De asemenea corectarea valorilor λ se face prin a testa pentru fiecare arc (i, j) dacă $\lambda_i + l_{ij} < \lambda_j$ și în caz afirmativ valoarea λ_j se taie și se scrie lângă ea valoarea $\lambda_i + l_{ij}$. Selectarea arcelor pentru test se poate face în orice ordine. Parcurgerea tuturor arcelor la o ieratăie de tip (b) este obligatorie doar pentru a decide oprirea algoritmului. În concluzie pasul (b) din algoritm este:

(b) ieratăia de bază

pentru $(i,j) \in U$ execută
 dacă $\lambda_i + l_{ij} < \lambda_j$ atunci $\lambda_j = \lambda_i + l_{ij}$ sfds;
 sfp;

Modul de parcurgere al vâfurilor în etapa (b) a algoritmului lui Ford influențează numărul de ieratăii necesare, mai general spus influențează complexitatea algoritmului. Astfel, dacă vâfurile ar fi parcurse în ordine crescătoare a valorilor λ_i obținute din algoritmul 3.1.1.1 atunci ar fi suficientă doar o ieratăie de tip (b) pentru a obține valorile minime. O îmbunătățire a complexității algoritmului se obține dacă etapa (b) se înlocuiește cu subetapele (b') și (b'') în care parcurgerea vâfurilor se face de la 2 la n respectiv de la n la 2. Această sugestie a fost făcută de Yen în 1972 [Yen72].

S-a văzut că pentru grafele fără circuite algoritmul lui Ford poate conduce la rezultate într-o singură ieratăie dacă se parcurg vâfurile grafului într-o anumită ordine. Dacă graful este fără circuite și vârful 1 este singurul vârf cu $\Gamma_1^{-1} = \emptyset$ atunci un algoritm care determină valorile minime ale drumurilor la o singură parcurgere a vâfurilor este:

Algoritmul 3.1.1.6.

Determină valorile minime ale drumurilor de la vârful 1 la orice alt vârf într-un graf fără circuite.

(a) inițializări

$$\lambda_1 := 0; S := \{1\};$$

(b) alegerea vârfului j

fie $j \in X - S$ astfel încât $\Gamma_j^{-1} \subset S$

(c) ieratăia de bază

$\lambda_j := \min \{ \lambda_i + l_{ij} \mid i \in \Gamma_j^{-1} \};$
 $S := S \cup \{j\};$

(d) criteriul de stop

```

dacă  $S=X$  atunci stop
altfel goto (b)
sf.d.

```

Observația 3.1.1.4.

- Algoritmul conduce la valorile optime pentru că etapa (b) este totdeauna posibilă.
- Valorile λ_j pentru $j \in X$ sunt valorile minime ale drumurilor de la vârful 1 la vârful j .

Demonstrație:

a. Folosim metoda reducerii la absurd. Presupunem că $\forall j \in X - S, \Gamma_j^{-1} \not\subset S$. Fie $i_1 \in X - S$ cum $\Gamma_{i_1}^{-1} \not\subset S$ deducem că $\Gamma_{i_1}^{-1} \cap (X - S) \neq \emptyset$. Alegem $i_2 \in \Gamma_{i_1}^{-1} \cap (X - S)$ deci $i_1 \in \Gamma_{i_2}$.

Apoi alegem $i_3 \in \Gamma_{i_2}^{-1} \cap (X - S)$ deci $i_2 \in \Gamma_{i_3}$, și aşa mai departe. Deci există un drum infinit cu toate vârfurile în $X - S$ și care are ca ultime vârfuri pe i_3, i_2, i_1 . Cum mulțimea $X - S$ este finită înseamnă că drumul conține un circuit, ceea ce contrazice ipoteza.

b. Fie $j \in X$. Ne situăm la momentul când se calculează valoarea λ_j . Din modul de construcție al mulțimii S , pentru că la momentul respectiv $\Gamma_j^{-1} \subset S$ și $\Gamma_i^{-1} \subset S, \forall i \in S$ rezultă că toate drumurile de la 1 la j trec doar prin vârfuri din S . Conform principiului lui Bellman înseamnă că λ_j este valoarea minimă a drumurilor de la 1 la j . \square

Parcuregerea vârfurilor în algoritm 3.1.1.6 se poate face și în ordine crescătoare a valorilor funcției rang atașată grafului care este fără circuite și cu rădăcina 1.

Definiția 3.1.1.1. Funcția rang atașată grafului $G = (X, U)$ este $r : X \rightarrow N$ definită prin $r(i) = \min \{ | \mu | \mid \mu \text{ este drum de la } 1 \text{ la } i \}, \forall i \in X$.

Se observă ușor că nu orice graf are o funcție rang. În cazul nostru are loc $r(i) = \max \{ | \mu | \mid \mu \text{ este drum de la } 1 \text{ la } i \}$.

Determinarea valorilor funcției rang se poate face cu:

Algoritm 3.1.1.7.

Determinarea funcției rang într-un graf fără circuite și cu rădăcina 1.

(a) inițializări

```

pentru  $i=1, n$  execută
 $d(i) := |\Gamma_i^{-1}|$ 

```

```

sfp;
 $k=0; S:=X;$ 

```

(b) iterația de bază

```

 $S_k := \{i \in S \mid d(i)=0\};$ 

```

```

pentru  $i \in S_k$  execută  $r(i) := k;$ 

```

```

pentru  $j \in \Gamma_i^{-1}$  execută  $d(j) := d(j)-1$  sfp;

```

```

sfp;

```

```

 $S := S - S_k;$ 

```

```

 $k := k+1;$ 

```

(c) criteriul de stop

```

dacă  $S=\emptyset$  atunci stop
altfel goto (b)

```

```

sf.d.

```

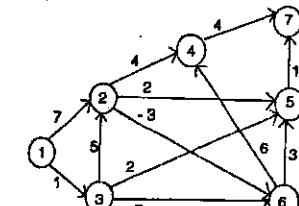


Fig. 3.1.1.4.

Exemplul 3.1.1.4.

Aplicând algoritmul 3.1.1.7 pentru graful din figura 3.1.1.4 se parcurg etapele:

(a) $d(1) = 0, d(2) = 2, d(3) = 1, d(4) = 2, d(5) = 3, d(6) = 2, d(7) = 2, k = 0,$

$S = \{1, 2, 3, 4, 5, 6, 7\}$

(b) $S_0 = \{1\}, r(1) = 0, d(2) = 1, d(3) = 0, S = \{2, 3, 4, 5, 6, 7\}, k = 1$

(b) $S_1 = \{3\}, r(3) = 1, d(2) = 0, d(5) = 2, d(6) = 1, S = \{2, 4, 5, 6, 7\}, k = 2$

(b) $S_2 = \{2\}, r(2) = 2, d(4) = 1, d(5) = 1, d(6) = 0, S = \{4, 5, 6, 7\}, k = 3$

(b) $S_3 = \{6\}, r(6) = 3, d(4) = 0, d(5) = 0, S = \{4, 5, 7\}, k = 4$

(b) $S_4 = \{4, 5\}, r(4) = 4, d(7) = 1, r(5) = 4, d(7) = 0, S = \{7\}, k = 5$

(b) $S_5 = \{7\}, r(7) = 5, S = \emptyset$

(c) stop.

Reprezentând graful din figura 3.1.1.4 prin a pune pe același nivel vârfurile care au același rang el arată ca în figura 3.1.1.5.

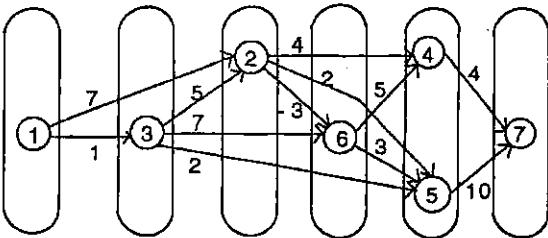


Fig. 3.1.1.5.

Observația 3.1.1.5.

Într-un graf care admite funcție rang vârfurile pot fi numerotate astfel încât $(i, j) \in U \Rightarrow i < j$.

Până acum ne-am ocupat doar de a determina valorile minime ale drumurilor de la un vârf la oricare dintre celelalte vârfuri. Uneori dorim să știm unul sau mai multe drumuri care au valoarea determinată. Algoritmul care urmează furnizează în ordine inversă vârfurile dintr-un asemenea drum.

Algoritm 3.1.1.8.

Determinarea unui drum de la vârful 1 la vârful i de valoare λ_i , după ce au fost determinate valorile minime λ_j , $\forall j \geq 2$.

(a) inițializări

$k=1$; $d(k)=i$;

(b) iterația de bază

$k=k+1$;

Fie $j \in \Gamma_i^k$ astfel încât $\lambda_j + l_{ji} = \lambda_k$;

$d(k)=j$; $i=j$;

(c) criteriul de stop

dacă $i=1$ atunci stop

altfel goto (b);

sfd.

Exemplul 3.1.1.5.

Vom aplica algoritm 3.1.1.8 pentru graful din figura 3.1.1.6, pentru care s-au calculat valorile minime ale drumurilor de la vârful 1 la celelalte vârfuri. Aceste valori sunt: $\lambda_2 = 7$, $\lambda_3 = 6$, $\lambda_4 = 3$, $\lambda_5 = 8$ și $\lambda_6 = 8$.

Dacă aplicăm algoritmul 3.1.1.8, pentru a afla un drum de la vârful 1 la vârful $i = 6$ de valoare $\lambda_6 = 8$, atunci valorile variabilelor, din algoritm, au următoarea evoluție:

(a) $k=1$, $d(1)=6$

(b) $k=2$, $j=3$, $d(2)=3$, $i=3$

(c) $i \neq 1$

(b) $k=3$, $j=5$, $d(3)=5$, $i=5$

(c) $i \neq 1$

(b) $k=4$, $j=2$, $d(4)=2$, $i=2$

(c) $i \neq 1$

(b) $k=5$, $j=1$, $d(5)=1$, $i=1$

(c) $i=1$ stop.

Deci parcugând vârfurile obținute în tabloul d de la valoarea lui k la 1 se obține drumul $\mu = \{1, 2, 5, 3, 6\}$ care are valoarea egală cu valoarea minimă a drumurilor de la vârful 1 la vârful 6.

Observația 3.1.1.6.

Algoritmii descriși pentru determinarea valorilor minime (maxime) de drumuri (lanțuri) se pot utiliza și pentru determinarea lungimilor minime (maxime) de drumuri (lanțuri). Singura schimbare este că valoarea fiecărui arc (muchii) se alege ca fiind 1.

3.1.2. Algoritmi matriceali pentru valori minime

Acești algoritmi calculează valorile minime ale drumurilor între oricare două vârfuri ale grafului.

Notăm cu $L^* = (l_{ij}^*)$ matricea cu elementele sale definite astfel:

$$l_{ij}^* = \begin{cases} \min\{l(\mu) | \mu \text{ drum de la } i \text{ la } j\}, & i \in X \text{ și } j \in \hat{X} \\ \infty, & \text{altfel} \end{cases}$$

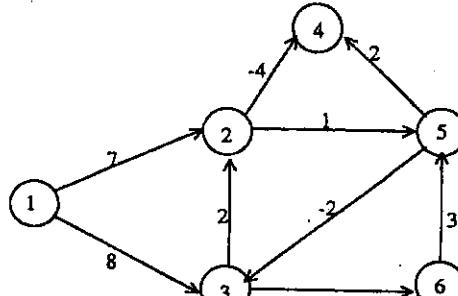


Fig. 3.1.1.6.

Considerăm matricea $V = (v_{ij})$ a valorilor arcelor grafului ca la algoritmul lui Bellman-Kalaba și definim matricile V^k pentru $k = 1, 2, \dots, n + 1$ astfel:

$$V^1 = V,$$

iar cunoșcând elementele matricei V^k le calculăm pe cele ale matricei V^{k+1} cu formula

$$v_{ij}^{k+1} = \min\{v_{ij}^k, v_{ik}^k + v_{kj}^k\}$$

Din modul de calcul remarcă că elementul v_{ij}^{k+1} conține valoarea minimă a drumurilor de la i la j care folosesc ca vârfuri intermedii doar vârfuri din mulțimea $\{1, 2, \dots, k\}$. Prin urmare are loc egalitatea $V^{n+1} = L^*$.

Răsonamentul de mai sus ne conduce la *algoritmul lui Floyd* [Flo62].

Algoritmul 3.1.2.1. Algoritmul lui Floyd-Hu

Determină matricea valorilor minime ale drumurilor între oricare două vârfuri ale unui graf care nu conține circuite de valoare negativă.

(a) inițializări

$$v_{ij}^1 := \begin{cases} 1_{ij}, & j \in \Gamma(i) \\ 0, & i = j \\ \infty, & \text{altfel} \end{cases}$$

(b) iterată de bază

```

pentru k=1,n execută
  pentru i=1,n execută
    pentru j=1,n execută
       $v_{ij}^{k+1} = \min\{v_{ij}^k, v_{ik}^k + v_{kj}^k\}$ 
    sf
  sf

```

Algoritmul lui Floyd este asemănător algoritmului lui Bellman-Kalaba și analog acestuia și algoritmului lui Floyd însă adus unele îmbunătățiri. Astfel se poate renunța la a construi o succesiune de matrici V^k și deci modificările să se facă tocmai în matricea valorilor arcelor. Algoritmul care urmează se aplică pentru cazul general și depistează dacă graful conține circuit de valoare negativă. Existența circuitului de valoare negativă este sesizată de faptul că o valoare v_{ii} devine negativă.

Algoritmul 3.1.2.2.

Determină matricea valorilor minime ale drumurilor între oricare două vârfuri ale unui graf sau depistează existența unui circuit de valoare negativă.

$k := 1;$

(a) $i := 1;$

(b) dacă $v_{ik} = \infty$ atunci goto (c) sfd;

dacă $v_{ik} + v_{ki} < 0$ atunci tipărește "există circuit de valoare negativă";
stop;

sfd;

pentru $j = 1, n$ execută

$$v_{ij} := \min(v_{ij}, v_{ik} + v_{kj});$$

sfp;

(c) $i := i + 1;$

dacă $i \leq n$ atunci goto (b) sfd;

$k := k + 1;$

dacă $k \leq n$ atunci goto (a) sfd;

stop.

Observația 3.1.2.1.

Algoritmul lui Flyod-Hu poate fi îmbunătățit pentru ca în paralel cu determinarea valorilor minime să fie determinate și drumurile care au aceste valori. Aceasta se realizează prin construirea unei matrici T de dimensiune $n \times n$, bazându-ne pe principiul lui Bellman. Conform acestui principiu dacă un drum de la i la j , de valoare minimă, trece printr-un vârf k atunci părțile de drum de la i la k respectiv de la k la j sunt de asemenei drumuri de valoare minimă. Algoritmul determină valorile matricii T și care permit următoarea interpretare:

$t_{ij} = k \Leftrightarrow k$ este vârf al drumului în drumul de valoare minimă de la i la j

sau

$t_{ij} = 0 \Leftrightarrow$ de la i la j nu există drum.

Algoritmul 3.1.2.3. Algoritmul lui Flyod-Warshall-Hu

Determină valorile minime ale drumurilor între oricare două vârfuri ale unui graf ce nu conține circuite de valoare negativă și matricea T din care se pot obține drumuri de valoare minimă.

(a) inițializări

$$v_i^1 := \begin{cases} I_j, & j \in \Gamma i \\ 0, & i = j \\ \infty, & \text{altfel} \end{cases}$$

$$t_{ij} = \begin{cases} i, & j \in \Gamma i \\ 0, & \text{altfel} \end{cases}$$

(b) iterată de bază

```

pentru k=1,n execută
  pentru i=1,n execută
    pentru j=1,n execută
      dacă  $v_{ij}^k > v_{ik}^k + v_{kj}^k$ 
        atunci  $v_{ij}^{k+1} := v_{ik}^k + v_{kj}^k$ ;  $t_{ij} := k$ ;
      altfel  $v_{ij}^{k+1} := v_{ij}^k$ 
    sfid;
  sfid;
  sfid.

```

Observația 3.1.2.2.

Algoritmul 3.1.2.3 se poate da în mai multe variante și de asemenei poate fi transformat într-un algoritm care să depisteze circuite (cicluri) într-un graf oarecare.

3.1.3. Determinarea valorilor maxime ale drumurilor

Raționând într-un mod analog cazului determinării valorilor minime deducem că valori maxime ale drumurilor într-un graf există dacă graful nu conține circuite de valoare pozitivă. Pentru determinarea valorilor maxime ale drumurilor se pot aplica algoritmii dați în paragraful 3.1.1 cu următoarele modificări:

- la inițializarea valorilor în locul lui $+\infty$ se utilizează valoarea $-\infty$;
- în condițiile de test se înlocuiește "min" și " $<$ " cu "max" respectiv " $>$ ".

Pentru exemplificare dăm algoritmul lui Bellman-Kalaba.

Algoritmul 3.1.3.1.

Determină valorile maxime ale drumurilor de la vârful 1 la oricare alt vârf al grafului.

(a) inițializări

$k := 1$;

(b) iterată de bază

$$\begin{cases} \lambda_1^{k+1} := 0 \\ \lambda_j^{k+1} := \max\{\lambda_i^k + v_{ij} \mid i = 1, n\}, & j \geq 2 \\ k := k + 1; \end{cases}$$

(c) criteriu de stop

```

dacă ( $\lambda_j^{k+1} = \lambda_j^k$  pentru  $j = 1, n$ ) atunci stop sfid;
dacă  $k \leq n$  atunci goto (b)
  altfel tipărește "există circuit de valoare pozitivă";
  stop;
sfid.

```

Rearmîntim observația:

Dacă se alege ca valoare pentru fiecare arc (muchie) valoarea 1, atunci algoritmii prezentați în paragraful 3.1 determină lungimile minime (maxime) ale drumurilor (lanțurilor) în graful dat (numai în aceste condiții valoarea fiecărui drum coincide cu lungimea drumului respectiv).

Problema 3.1.3.1.

Pentru graful din figura 3.1.3.1. să se determine:

- valoarea minimă a drumurilor de la vârful 1 la vârful 3;
- un drum de valoare minimă de la vârful 1 la vârful 3;
- valoarea minimă a lanțurilor de la vârful 1 la vârful 4;
- câte lanțuri de lungime 8 pornesc din vârful 2 (au pe 2 ca extremitate).

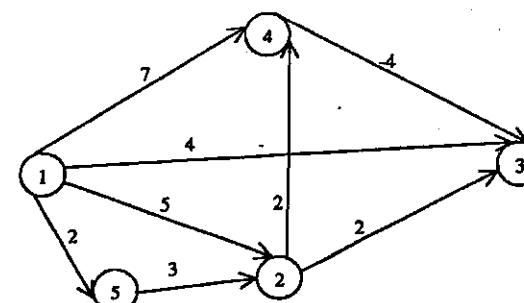


Fig. 3.1.3.1.

Nu vom rezolva explicit această problemă (lăsăm plăcerea pe seama cititorului). Totuși facem câteva observații-comentariu care pot aduce clarificări asupra noțiunilor și nuanțelor cu care se manevrează în teoria grafelor.

Pentru punctul a) se cere un număr care să fie valoarea minimă dintre valorile tuturor drumurilor de la vârful 1 la vârful 3. Acest număr se poate determina cu unul dintre algoritmii prezenți (Moore-Dijkstra, Bellman-Kalaba, Ford, Yen, Floyd-Hu sau Floyd-Warshall-Hu). Dar nu este recomandabil ca rezolvarea să folosească "funcția ochi", adică să enumerăm toate drumurile de la vârful 1 la vârful 3 și apoi afănd valoarea fiecărui drum să alegem valoarea minimă (această metodă este utilă doar pentru rezolvarea manuală și pentru grafe nu prea complexe). Precizăm, pentru verificare, că valoarea cerută este 3.

La punctul b) se cere un drum, deci o succesiune de vârfuri sau arce cu anumite proprietăți (drumul să aibă valoarea minimă). Rezolvarea are două etape mai întâi se determină valoarea minimă (etapă parcursă la punctul precedent) și apoi depistarea vâfurilor unui drum care să aibă valoarea egală cu valoarea minimă (în cazul nostru valoarea minimă este 3 și drum poate fi 15243 sau 1243 sau 143).

Pentru punctul c) deoarece se cere valoarea minimă a lanțurilor, mai întâi transformăm graful într-un graf neorientat (se înlocuiesc arcele cu muchii sau transformăm matricea valorilor într-o matrice simetrică) și apoi se aplică unul din algoritmii precizați pentru punctul a), deoarece graful are și arce (muchii) de valoare negativă se va aplica un algoritm care să depisteze dacă este cazul existență circuitelor (ciclurilor) de valoare negativă ceea ce conduce spre concluzia că în graful dat nu există drumuri (lanțuri) de valoare minimă. Deoarece ciclul 2342 are valoarea 0, există de exemplu o infinitate de lanțuri de valoare 7 de la vârful 1 la vârful 4 (dintre acestea în ordine crescătoare după lungimea lor precizăm: 14, 124, 1234, 1524, 15234, 14324, 1234234, 1524324, 14324234, 1234234234, etc.). Valoarea minimă a lanțurilor dintre vâfurile 1 și 4 este 0 și un asemenea lanț este lanțul 134, dar și lanțuri de valoarea 0 există o infinitate (sugerate de 134234, 134234234, etc.).

Pentru punctul d) să observăm că ne interesează lungimea nu valoarea lanțurilor. De aceea valorile arcelor (muchiilor) pentru problema aceasta sunt date suplimentare și care nu se vor folosi în calcule (aceste valori pot constitui bruijaje pentru rezolvator). Pentru a determina lungimile minime (maxime) cu algoritmii utilizati pentru determinarea valorilor minime (maxime) se vor considera ca valori pentru arce (muchii) valoarea 1. Dar numărul drumurilor se poate calcula și prin a ridica matricea de adiacență la puterea egală cu lungimea drumurilor (lanțurilor). Aici, fiindcă este vorba de lanțuri vom transforma arcele în muchii (se utilizează graful neorientat corespunzător grafului dat). Deci se construiește matricea de adiacență A atașată noului graf (simetrizarea matricii de adiacență atașată grafului dat prin înlocuirea unor zerouri din ea cu 1). Se calculează apoi A^2 , $A^4 = A^2A^2$ și $A^5 = A^4A$ și suma elementelor de pe linia sau coloana 2 ale matricii A^5 este

numărul lanțurilor de lungime 5 ce au vârful 2 ca extremitate (vezi teorema 1.2.2.1.)

Problema 3.1.3.2.

- Dacă $G = (X, U)$ este un graf conex, atunci oricare două lanțuri elementare de lungime maximă, ale lui G , au cel puțin un vârf comun.
- Dacă, în plus, G este arbore atunci toate laturile elementare de lungime maximă, ale lui G , au cel puțin un vârf comun.

Demonstrație:

a. Demonstrăm prin metoda reducerii la absurd. Fie L_1 și L_2 (figura 3.1.3.1.) două lanțuri elementare de lungime maximă între vâfurile a și c respectiv b și d și despre care presupunem că nu au nici un vârf comun. Cum G este un graf conex, rezultă că între vâfurile a și d există cel puțin un lanț L . Deoarece $a \in L_1$ și $d \in L_2$ în parcurgerea lui L de la a la d notăm cu p ultimul vârf din $L_1 \cap L$ și cu q succesorul lui p în L , deci $q \in L_2 \cap L$. Dintre sublanțurile $[a, p]$ și $[p, c]$ ale lui L_1 îl alegem pe cel de lungime mai mare, fie acesta $[a, p]$. Analog dintre sublanțurile $[b, q]$ și $[q, d]$ îl considerăm pe $[q, d]$ ca fiind de lungime mai mare. În acest caz $[a, p] \cup \{(p, q)\} \cup [q, b]$ este lanț elementar și are lungimea cu cel puțin 1 mai mare decât lanțurile L_1 și L_2 care au lungimea maximă. Această contradicție demonstrează cazul a) al problemei.

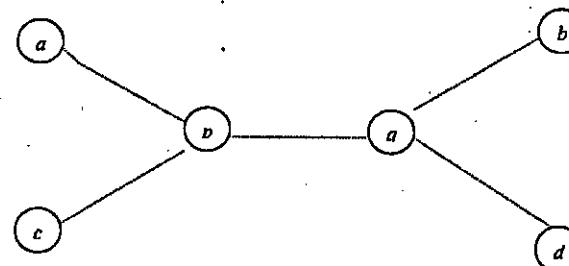


Fig. 3.1.3.1.

- b. Fie deci un lanț fix L_1 și un lanț oarecare L_2 (figura 3.1.3.1.), în arborele G , două lanțuri elementare de lungime maximă între vâfurile a și b respectiv c și d (extremitățile lanțurilor nu s-au ales ca la punctul precedent). Conform demonstrației de la punctul a) cele două lanțuri au cel puțin un vârf comun. În parcurgerea lanțului L_1 de la a la b fie p primul vârf și q ultimul vârf din $L_1 \cap L_2$. G fiind un arbore rezultă că între oricare două vârfuri există un singur arbore și deci sublanțul $[p, q] \subset L_1 \cap L_2$, ceea ce nu se întâmplă în cazul G conex. Pentru că L_1 și L_2 au

lungimea maximă în G și deci lungimile lor sunt egale deducem că sublanțurile $[a, p]$ și $[c, p]$ respectiv $[b, q]$ și $[q, d]$ au lungimi egale și fiecare are lungimea mai mică decât jumătate din lungimea maximă. Astfel, s-ar depista, analog cazului precedent, un lanț mai lung decât L_1 . De aici se deduce că vârful din mijlocul (sau cele două de la mijloc) se include în $L_1 \cap L_2$. Cum lanțul L_1 este fix, iar L_2 este un lanț oarecare între cele de lungime maximă din G , rezultă că cel puțin mijlocul lui L_1 aparține intersecției tuturor lanțurilor elementare de lungime maximă. Să remarcăm că intersecția acestor lanțuri, într-un arbore, este de asemenea un lanț.

Precizăm că există grafe conexe care au un singur lanț de lungime maximă și există arbori care chiar dacă au mai multe lanțuri de lungime maximă, intersecția acestora este un singur vârf.

Pe de altă parte ținând cont de teorema 1.4.1.6. deducem că în intersecția lanțurilor de lungime maximă toate vâfurile care nu sunt frunze sunt puncte de articulație și acestea formează o mulțime de articulație a arborelui G .

3.2. Optimizări de drumuri în ipoteza $l(\mu) = \prod_{u \in \mu} l(u)$

În aceste probleme valoarea drumurilor se calculează ca fiind produsul valorilor arcelor. Acest mod de calcul este aplicabil în cazurile în care valoarea arcului este probabilitatea unui eveniment (care este un număr pozitiv). În aceste cazuri parcurgerea drumului presupune realizarea tuturor evenimentelor care compun drumul. Astfel de probleme se întâlnesc în teoria jocurilor, cu aplicații în probleme militare. Algoritmii dați în paragraful 3.1 se transformă foarte ușor pentru acest caz prin a schimba: inițializările din 0 în 1 (elementul neutru față de înmulțire) și operația dintre valorile de arce și/sau de drumuri din adunare în înmulțire. Vom prezenta doar algoritmii lui Bellman-Kalaba pentru determinarea valorii minime respectiv a valorii maxime.

Algoritmul 3.2.1.

Determină valorile minime a drumurilor de la vârful 1 la oricare alt vârf sau depistează existența de circuite.

(a) inițializări

$$k := 1;$$

(b) iterată de bază

$$\begin{cases} \lambda_1^{k+1} := 1 \\ \lambda_j^{k+1} := \min\{\lambda_i^k v_{ij} \mid i = 1, n\}, \quad j \geq 2 \end{cases}$$

$$k := k + 1;$$

(c) criteriu de stop

dacă $(\lambda_j^{k+1} = \lambda_j^k$ pentru $j = 1, n)$ atunci stop sfid;

dacă $k \leq n$ atunci goto (b)

altfel tipărește "există circuit de valoare subunitară"

stop;

sfid.

Algoritmul 3.2.2.

Determină valorile maxime de la vârful 1 la orice alt vârf al grafului sau depistează existența de circuite în graf.

(a) inițializări

$$k := 1;$$

(b) iterată de bază

$$\begin{cases} \lambda_1^{k+1} := 1 \\ \lambda_j^{k+1} := \max\{\lambda_i^k v_{ij} \mid i = 1, n\}, \quad j \geq 2 \\ k := k + 1; \end{cases}$$

(c) criteriu de stop

dacă $(\lambda_j^{k+1} = \lambda_j^k$ pentru $j = 1, n)$ atunci stop sfid;

dacă $k \leq n$ atunci goto (b)

altfel tipărește "există circuit de valoare supraunitară"

stop;

sfid.

Observația 3.2.1.

Problema determinării valorilor optime ale drumurilor în ipoteza $l(\mu) = \prod_{u \in \mu} l(u)$ se reduce la problema determinării drumurilor optime în

ipoteza $l(\mu) = \sum_{u \in \mu} l(u)$, prin schimbarea valorilor arcelor din $l(u)$ în $\ln(l(u))$, dacă valorile sunt pozitive, deoarece funcție logaritmică este strict monotonă.

Observația 3.2.2.

În algoritmii de optimizare pentru a preciza că între două vârfuri nu există arc (în matricea V) sau că încă nu s-a depistat un drum (în vectorii λ) au fost utilizate valorile ∞ sau $-\infty$, depinzând de faptul că se căuta valoarea

minimă respectiv maximă a drumurilor. În programele pentru calculator se utilizează o valoare din afara domeniului datelor și rezultatelor intermediiare. Trebuie avute în vedere regulile de calcul cu valoare aleasă în locul lui ∞ , deoarece $\infty + a = \infty$ și $a\infty = \infty$ pentru $a > 0$ deci tot astfel trebuie să se facă operațiile și cu valoarea folosită în locul lui ∞ .

3.3. Problema ordonanțării. Drum critic

Așa cum reiese din exemplul 3.2 algoritmi de determinarea drumurilor optime se pot utiliza pentru alegerea unei variante de construirea unei căi de comunicație. Pentru aceasta prezintă interes doar un drum dintre toate drumurile grafului care conectează două vârfuri date ale grafului. Problema ordonanțării se ocupă cu programarea activităților prin desfășurarea cărora se realizează un obiectiv (stabilirea unui grafic de execuție al activităților). Deoarece trebuie executate toate activitățile, prezintă interes toate drumurile grafului dintre două vârfuri bine precizate ale grafului.

Pentru realizarea unui obiectiv sau scop trebuie să realizeze un număr mai mare sau mai mic de activități. Deoarece între aceste activități există o interdependență și o anumită coordonare este foarte importantă programarea desfășurării lor și cunoașterea punctelor sensibile pentru ca obiectivul propus să se realizeze în condiții de eficiență (timp, cost, beneficii, etc.). Această problemă este cunoscută cu numele de problema de ordonanțare. În continuare vom considera doar că singurele condiții de realizare a activităților sunt cele de timp, adică unele activități pot începe cel mai devreme imediat după terminarea altor activități bine precizate. Deci pentru fiecare activitate se cunoaște durata să în timp și care sunt activitățile ce trebuie realizate înainte ca ea să fie abordată. Reprezentarea acestei probleme sub formă de graf și studiul acestuia ne permite să avem o privire globală asupra problemei. Teoria grafelor ne ajută ca pentru o asemenea problemă să determinăm timpul minim necesar de realizare a obiectivului precum și marjele de timp pentru fiecare activitate. Astfel hotărările luate pot fi în consens cu scopul urmărit. Problemelor de ordonanțare li se pot ataşa mai multe tipuri de grafe. Noi vom prezenta două dintră acestea.

3.3.1. Graful potențiale-activități

Acest tip de graf a fost propus de către B. Roy în 1960 [Roy60]. Graful se construiește astfel:

- Fiecărei activități i se atașează un vârf.

- Definim un arc de la i la j dacă activitatea j trebuie să înceapă după ce s-a terminat activitatea i . Acestui arc $i \rightarrow j$ se dă valoarea d_{ij} care este durata de desfășurare a activității j . Graful astfel definit este fără circuite, căci existența unui circuit ar însemna că o activitate trebuie să se realizeze înainte ca ea să înceapă. Adăugăm grafului două vârfuri corespunzătoare la două activități fictive. Primul vârf corespunde activității de debut notată cu 0, de durată 0 și care trebuie să preceadă orice altă activitate, iar celălalt vârf corespunde activității de sfârșit notată $n + 1$ care trebuie să urmeze tuturor activităților.

Exemplul 3.3.1.1.

Construirea unei clădiri necesită realizarea următoarelor activități:

Nr.	Cod	Denumire activitate	Durată	Activ. precedente
1	A	Amenajare căi de acces	7	-
2	B	Săpare săncuri fundație	3	1
3	C	Constr. struct. de rez.	8	2
4	D	Montaj inst. sanitare și electrice	2	3
5	E	Montaje uși-ferestre	3	3
6	F	Montare schelă	1	3
7	G	Realizare fațadă	3	6
8	H	Zugrăveală interioară	2	4, 5
9	I	Amenajări interioare	1	8
10	J	Demontare schelă	1	7
11	K	Curățenie	2	9, 10
12	L	Imprejmuire	3	1

Pentru această problemă graful potențiale-activități atașat este prezentat în figura 3.3.1.1.

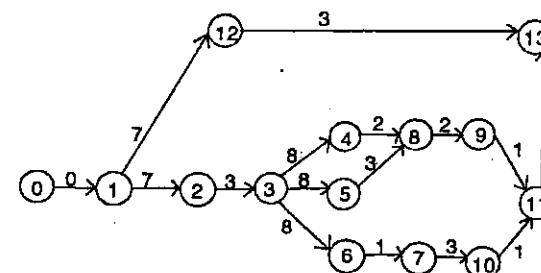


Fig. 3.3.1.1.

Pentru a demara o activitate trebuie ca toate activitățile care se află pe un drum de la debutul lucrării (activitatea 0) la vârful corespunzător acelei activități să fie executate. Dacă notăm cu t_i momentul cel mai devreme de începere a activității i deducem că aceste valori trebuie să verifice

$$t_i = \max\{t_j + d_j \mid j \in \Gamma_i^{-1}\}.$$

Prin urmare, t_i se calculează ca fiind valoarea maximă a drumurilor de la vârful 0 la vârful i , aceasta pentru că trebuie să se realizeze toate activitățile j cu $i \in \Gamma^+ j$. Deci, durata minimă de realizare a proiectului este t_{n+1} .

Dacă s-au determinat momentele cele mai timpuri de începere a activităților, t_i , $i = 1, n+1$, atunci se vor putea calcula și momentele cele mai târzii de începere a activităților, T_i , $i = 0, n+1$, care nu conduc la o amânare a realizării obiectivului. Modul de calcul se aseamănă cu cel de determinare a unui drum după ce se cunosc valorile optime ale drumurilor de la vârful 1 la celelalte vârfuri. Pentru că $t_{n+1} - T_i = \max\{l(\mu) \mid \mu \text{ este drum de la } i \text{ la } n+1\}$, calculele se fac pornind de la vârful $n+1$ către vârful 1 astfel:

(a) inițializări

$$T_{n+1} := t_{n+1}; S := \{n+1\};$$

(b) iterația de bază

$$\text{fie } i \notin S \text{ astfel încât } \Gamma_i \subseteq S; \\ T_i := \min\{T_j - d_{ij} \mid j \in \Gamma_i\}; S := S \cup \{i\};$$

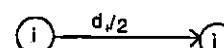
(c) criteriul de stop

$$\text{dacă } S = \{0, 1, 2, \dots, n+1\} \text{ atunci stop} \\ \text{altfel goto (b)} \\ \text{sfd.}$$

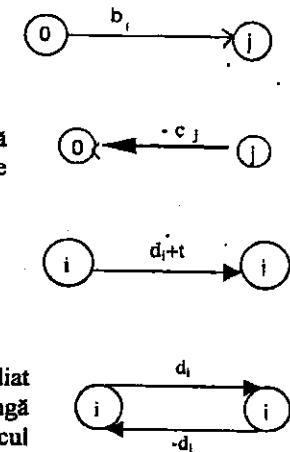
Valoarea $T_i - t_i$ notată m_i este marja de timp pentru activitatea $i \in \{0, 1, 2, \dots, n+1\}$. În intervalul de timp $[t_i, T_i]$ activitatea i poate demara oricând fără ca timpul de realizare a obiectivului să se modifice. Activitățile pentru care $t_i = T_i$ se numesc **activități critice**. Dacă activitatea i începe după momentul de timp T_i atunci cu timpul de depășire se întârzie realizarea obiectivului față de timpul optim.

În graful potențiale-activități se pot marca și unele restricții practice. În acest sens prezentăm câteva asemenea restricții și modul de marcarea lor în graf.

- a. Restricția "activitatea j nu poate începe decât după ce s-a realizat o parte (de exemplu jumătate) din activitatea i' " se marchează printr-un arc de la i la j de valoare egală cu timpul necesar realizării părții respective.



- b. Dacă activitatea j nu poate începe mai devreme de momentul b_j atunci se adaugă arcul $(0, j)$ de valoare b_j .
- c. Pentru activitățile j care trebuie să înceapă înaintea momentului c_j se adaugă arcele $(j, 0)$ de valoare $-c_j$.
- d. Dacă activitatea j nu poate începe decât după un interval de timp t de la realizarea activității i atunci se adaugă arcul (i, j) de valoare $d_i + t$.
- e. Dacă o activitate j trebuie să înceapă imediat după realizarea activității i atunci pe lângă arcul (i, j) de valoare d_i se mai adaugă arcul (j, i) de valoare $-d_i$.



3.3.2. Graful potențiale-etape (PERT)

Grafele PERT (*Programme Evaluation and Research Task*) au fost definite în 1959 de către Malcom, Roseboom, Clark și Fazar [MRC59]. În această reprezentare activităților le sunt asociate arcele grafului cărora li se atribuie ca valoare durata de realizare a activităților respective. Vârfurile se consideră a fi etape ale realizării proiectului, de debut sau de sfârșit a unor activități. Astfel apar etapele de debut și de încheiere a realizării obiectivului. Dacă o activitate j trebuie să succede activității i atunci extremitatea terminală a

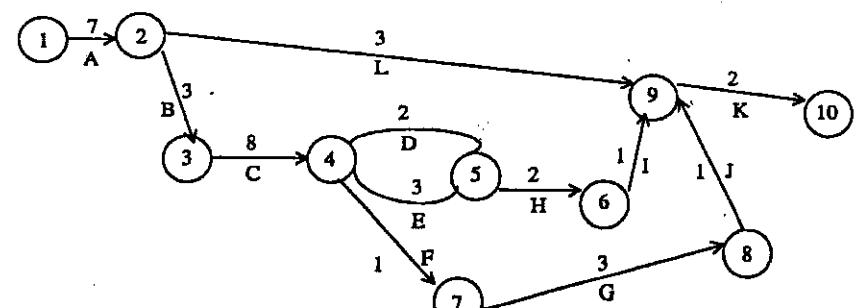


Fig. 3.3.2.1.

arcului i va coincide cu extremitatea inițială a arcului j . Graful astfel construit pentru exemplul 3.3.1.1 este cel din figura 3.3.2.1. Pentru a nu genera confuzii, arcele (activități) au fost marcate prin codul lor și pentru fiecare arc s-a trecut valoarea sa (durata de realizare a activității respective).

Deoarece în această reprezentare vârfurile sunt interpretate ca etape graful se numește potențiale-etape. În extremitatea inițială a oricărui arc vor "ajunge" drumuri de la etapa de debut pe care să se afle arcele atașate activităților care trebuie terminate înaintea începerii activității reprezentate de acel arc. Cel mai scurt timp de execuție, analog grafelor potențiale-activități, este valoarea maximă a drumurilor de la activitatea de debut la activitatea finală pe care o notăm cu n . Valorile maxime ale drumurilor de la activitatea de debut să le notăm cu λ (sugérând aplicarea unui algoritm de tip Bellman-Kalaba sau Ford). Dacă activitatea M are ca extremități etapele p și q și durata ei este d , atunci interpretarea valorilor λ găsite este:

activitatea M poate să înceapă cel mai devreme la momentul λ_p și cel mai târziu la momentul $\lambda_q - d$ astfel încât obiectivul să fie realizat în timp optim (în λ_n unități de timp de la debut).

Drumul de valoare maximă între etapa inițială și cea finală se numește *drum critic*, iar activitățile care compun acest drum se numesc *activități critice*. Într-un asemenea graf s-ar putea să existe mai multe drumuri critice. Orice activitate critică trebuie să înceapă imediat ce s-a terminat activitatea critică ce o precede pe un drum critic deoarece pentru aceste activități marjele de timp sunt 0.

Precizăm că drum critic există (s-a definit) numai pentru graful potențiale-etape, în timp ce activități critice există pentru ambele tipuri de grafe.

3.4. Problema comis-voiajorului

Un comis-voiajor trebuie să viziteze un număr oarecare de puncte (orașe) date. Considerând punctele ca vârfuri ale unui graf și rutile între puncte ca muchii, atunci probleme revine la a determina un ciclu hamiltonian. Graful este complet pentru că în cazul în care între două puncte ruta nu este directă se marchează lanțul dintre ele ca muchie. Această muchie semnifică faptul că pornind dintr-o extremitate a muchiei comis-voiajorul face următoarea escală în punctul desemnat de celalătă extremitate a muchiei respective. De regulă muchiile li se asociază valori care poate fi costul trecerii dintr-un vârf în celălalt, distanța de parcurs, durata trecerii dintr-un vârf în celălalt, etc. De aceea dintre ciclurile hamiltoniene interesează acel ciclu care are costul (valoarea) minimă. Așa cum s-a demonstrat în teorema 1.1.3, numărul ciclurilor hamiltoniene în K_n este $(n - 1)!/2$, deci enorm. Un algoritm care să nu parcurgă toate aceste cicluri nu se cunoaște, dar se

utilizează o serie de algoritmi euristică care dau soluții aproximative ale soluției optime. Dintre acești algoritmi, o să prezentăm doi.

Algoritm 3.4.1. (cel mai aproape vecin)

Se pornește dintr-un vârf oarecare al grafului. La fiecare pas se alege vârful, dintre cele neselectate, cu proprietatea că muchia între ultimul vârf selectat și acesta este de valoare minimă. Deci, dintre muchiile incidente ultimul vârf selectat și vârfurile neselectate se alege muchia cu valoarea cea mai mică și extremitatea acesteia se va selecta la acest pas. Procedeu continuă până când vor fi selectate toate vârfurile și apoi se va adăuga muchia determinată de primul vârf și ultimul vârf selectat. Este evident că pornind de la vârfuri diferite se vor obține cicluri hamiltoniene diferite de regulă și de valori diferite.

Algoritm 3.4.2. (muchia de valoare cea mai mică)

Se ordonează muchiile grafului K_n în ordine crescătoare a valorilor lor. Se analizează muchiile în această ordine și unele dintre ele vor fi selectate. Notăm cu $G = (X, U)$ graful ce are cele n vârfuri și U = muchiile selectate. Dacă în graful G împreună cu muchia care se analizează gradele vârfurilor este cel mult 2 și nu conține ciclu de lungime mai mică decât n , atunci muchia care se analizează va fi selectată (se adaugă în U). Acest procedeu continuă până când U conține n muchii (toate vârfurile în G au gradul 2).

Exemplul 3.4.1.

Considerăm graful din figura 3.4.1. și vom exemplifica aplicarea celor doi algoritmi prezenți.

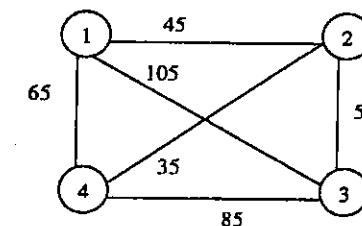


Fig.3.4.1.

Aplicăm acum algoritm 3.4.1. pornind de la vârful 1. Dintre muchiile candidate 12, 13 și 14 valoarea cea mai mică o are muchia 12 și este 45, deci va fi selectat vârful 2. Dintre muchiile candidate 23 și 24 valoarea cea mai mică o are

muchia 24 și este 35, deci se va selecta vârful 4. Există o singură muchie candidată 43 de valoare 85, deci se va selecta vârful 3. Ciclul hamiltonian va conține pe lângă muchiile selectate și muchia 31 de valoare 105. Deci soluția depistată este ciclul 12431 și care are valoarea 270.

Dacă aplicăm acum algoritm 3.4.1. pornind de la vârful 2. Dintre muchiile candidate 21, 23 și 24 valoarea cea mai mică o are muchia 24 și este 35, deci va fi selectat vârful 4. Dintre muchiile candidate 41 și 43 valoarea cea mai mică o are muchia 41 și este 65, deci se va selecta vârful 1. Există o singură muchie candidată 13 de valoare 105, deci se va selecta vârful 3. Ciclul hamiltonian va conține pe lângă muchiile selectate și muchia 32 de valoare 55. Deci soluția depistată este ciclul 24132 și care are valoarea 260.

Dacă aplicăm algoritmul 3.4.2, ordinea de analizare a muchiilor (crescătoare după valorile lor) este: 24, 12, 23, 14, 34, 13. Se selectează muchia 24, apoi 12. Apoi, se analizează muchiile 23 și 14 fără a fi selectate pentru că vârful 2 ar avea gradul 3 respectiv ar exista ciclul 1241 care are lungimea $3 < 4$. În final se selectează succesiv muchiile 34 și 13 astfel au fost selectate patru muchii ce formează ciclul hamiltonian 13421 care are valoarea 270 (același ciclu cu cel selectat la prima aplicare a algoritmului 3.4.1. de mai sus).

Cu toate acestea ciclul hamiltonian de valoare minimă pentru exemplu considerat este 21432 care are valoarea 250.

4. FLUXURI ȘI REȚELE DE TRANSPORT

Noțiunea de *flux* furnizează un model general aplicabil într-un mare număr de probleme concrete. Dintre aceste probleme amintim: probleme de programarea transportului (rutier, aerian, feroviar, etc.), structurarea și dimensionarea optimă a rețelelor de comunicații, probleme de gestiunea stocurilor, de ordonanțare, de afectare, etc. Noțiunea de flux a condus și la dezvoltarea unui aparat matematic specific aparținând, în mare parte, combinatoricii și matematicii discrete.

4.1. Definiții și proprietăți de bază

Definiția 4.1.1. Fie $G = (X, U)$ un graf conex de dimensiune m ($m = |U|$). *Flux* în graful G este orice vector $\varphi = (\varphi_1, \varphi_2, \dots, \varphi_m) \in \mathbb{R}^m$ (fiecarei arc u îi se atașează o valoare φ_u , semnificând fluxul transportat pe arcul respectiv), care verifică prima lege a lui Kirchhoff în orice vârf i al grafului, adică $\sum_{u \in \omega^-(i)} \varphi_u = \sum_{u \in \omega^+(i)} \varphi_u$, numită și *legea conservării fluxului în vârfuri (noduri)*.

Pentru arcul u valoarea φ_u se numește *cantitatea de flux* de pe arcul u sau simplu *fluxul* de pe arcul u .

Exemplul 4.1.1.

Curentul electric într-o rețea este un exemplu de flux într-o rețea. O astfel de schemă (graf) este dată în figura 4.1.1.

Așa cum se vede și pe figură fluxul din graful reprezentat în figura 4.1.1 este $\varphi = (5, 2, 3, -2, -1, 4)$. Alegera orientării unui arc se poate considera arbitrară (în acest caz), deoarece sensul de deplasare al curentului poate fi pe sens invers orientării arcelor acest fapt se deduce din semnul fluxului pe arc. De exemplu $\varphi_4 = -2$ indică faptul că pe arcul 4 curentul circulă de la vârful 2 spre vârful 3.

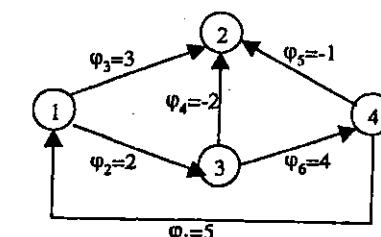


Fig. 4.1.1.

Considerând matricea de incidență $B = (b_{iu})$, $i \in X$, $u \in U$ a grafului G , deoarece $\omega^+(i) = \{u \in U \mid b_{iu} = 1\}$ și $\omega^-(i) = \{u \in U \mid b_{iu} = -1\}$, legea conservării fluxului în vârfuri se poate scrie în forma: $B\varphi = 0$. Prin urmare considerând aplicația liniară asociată matricii B , $g : \mathbb{R}^m \rightarrow \mathbb{R}^n$ cu $g(x) = Bx$, mulțimea Φ a fluxurilor pe graful G este nucleul funcției g . Adică

$$\Phi = \{\varphi \in \mathbb{R}^m \mid g(\varphi) = 0\}.$$

Aceasta înseamnă că Φ este un subspațiu liniar al lui \mathbb{R}^m și are loc $\dim(\Phi) + \text{rang}(B) = m$. Dacă G are p componente conexe, are loc $\text{rang}(B) = n - p$ (numărul cociclomatic) și deci $\dim(\Phi) = m - n + p$ care este numărul ciclomatic al grafului, notat cu $v(G)$.

Teorema 4.1.1. *Un flux pe un arbore (arborescență) este în mod necesar fluxul nul.*

Demonstrație:

Un arbore are $m = n - 1$ și $p = 1$, deci $\text{rang}(B) = n - 1$. Prin urmare sistemul $B\varphi = 0$, cu $n - 1$ necunoscute, este compatibil determinat și admite ca soluție doar soluția banală, altfel spus $\Phi = \{0\} \subset \mathbb{R}^m$. \square

Definiția 4.1.2. Fie φ un flux, notăm cu $\text{supp}\varphi = \{u \in U \mid \varphi_u \neq 0\}$ și o numim *suportul fluxului* φ .

Definiția 4.1.3. Un flux $\varphi \neq 0$ se numește *flux elementar* dacă și numai dacă suportul său este minimal în raport cu incluziunea.

Propoziția 4.1.1. *Un flux este elementar dacă și numai dacă are ca suport un ciclu elementar.*

Demonstrație:

Suficiență

Mai întâi să observăm că vectorii atașați ciclurilor elementare sunt fluxuri deoarece verifică legea conservării fluxului în noduri. Prin urmare un flux care are ca suport un ciclu elementar este produsul dintre un număr și vectorul atașat aceluui ciclu elementar. Dacă nu ar fi așa ar exista un vârf în care nu este verificată legea conservării fluxului.

Necesitatea

Un flux nenul conține în mod necesar cel puțin un ciclu în suportul său, deoarece altfel el este flux pe o arborescență ceea ce ar fi o contradicție înțând

cont de teorema 4.1.1. În sfârșit un flux care conține mai mult decât un ciclu elementar în suportul său nu poate fi elementar deoarece vectorul atașat aceluui ciclu este flux și are suportul conținut strict în suportul fluxului considerat.

Teorema 4.1.2. *Orice flux $\varphi \in \Phi$, este o combinație liniară a celor $v(G)$ vectori asociati ciclurilor elementare liniar independente ale grafului. Adică*

$$\exists \lambda_i \in \mathbb{R}, i = 1, v(G) \text{ astfel încât } \varphi = \sum_{i=1}^{v(G)} \lambda_i \mu^i.$$

Demonstrație:

Fie $P = (X, T)$ un arbore al grafului G dat. Pentru $u \in U - T$ notăm cu μ^u unicul ciclu din $T \cup \{u\}$, care este elementar (el există și este unic conform teoremei 1.4.3.1.) și cu $\bar{\mu}^u$ vectorul asociat ciclului astfel încât $\bar{\mu}_u^u = 1$. Deoarece $(\Phi, +)$ este un grup abelian, vectorul

$$\varphi' = \varphi - \sum_{u \in X - T} \varphi_u \bar{\mu}^u$$

este un flux cu proprietatea $\text{supp } \varphi \subset T$. Cum (X, T) este arbore deducem că

$$\varphi' = 0 \text{ deci } \varphi = \sum_{u \in X - T} \varphi_u \bar{\mu}^u. \square$$

4.2. Lema arcelor colorate (Lema lui Minty)

Lema 4.2.1. Lema lui Minty [Min66]

Fie $G = (X, U)$ un graf în care s-au colorat arbitrar arcele în negru, roșu sau verde (unele putând rămâne necolorate). Să presupunem că există cel puțin un arc colorat în negru, pe care îl notăm u_0 . Atunci numai una din următoarele propoziții este adevărată:

- Prin u_0 trece un ciclu elementar cu toate arcele colorate, astfel încât toate arcele sale negre sunt orientate în același sens (sensul lui u_0) și toate arcele sale verzi sunt orientate în sens invers.
- Prin u_0 trece un cociclu elementar fără arce roșii, dar poate avea arce necolorate astfel încât toate arcele sale negre sunt orientate în același sens și toate arcele sale verzi sunt orientate în sens contrar.

Observație:

Precizăm că ipoteza lemei se poate interpreta și astfel:

mulțimea arcelor grafului este descompusă în patru submulțimi disjuncte (fiecare având arcele sale de aceeași culoare), dar dintre acestea numai despre o submulțime știm sigur că este nevidă (cea care conține arcele colorate în negru).

Demonstrație:

Considerăm că $u_0 = (t, s)$. Vom marca din aproape în aproape vârfurile lui G pornind de la s astfel:

- se marchează vârful s ;
- dacă i este marcat atunci se va marca vârful j , nemarcat, dacă:
 - = fie (i, j) este negru;
 - = fie (i, j) -sau (j, i) este roșu;
 - = fie (j, i) este verde.

Se face marcarea vârfurilor până când nu se mai pot marca alte vârfuri. Să remarcăm faptul că selectând doar vârfurile marcate și muchiile (prin neglijarea orientării arcelor) care au cauzat marcarea vârfurilor se obține un arbore de rădăcină s . Deci, pentru fiecare vârf j marcat există un singur lanț de la rădăcina s la el. Este posibilă numai una dintre situațiile:

1. S-a marcat vârful t .

Aceasta conduce la concluzia că există un lanț elementar de la s la t , ce trece numai prin vârfuri marcate. Deci toate arcele lanțului sunt colorate, arcele negre sunt parcurse în sensul orientării lor, iar arcele verzi sunt parcurse în sens invers orientării lor. Adăugând arcul u_0 la acest lanț se obține un ciclu care face să fie adevărată propoziția a din lemă.

2. Nu s-a marcat vârful t .

Notăm cu A mulțimea vârfurilor marcate. Din modul de marcare al vârfurilor se deduce că: $s \in A$, $t \notin A$, $\omega^+(A)$ conține numai arce verzi sau necolorate, $\omega^-(A)$ conține numai arce negre sau necolorate și $\omega(A)$ nu are arce roșii. Prin urmare cocicul $\omega(A) = \omega^+(A) \cup \omega^-(A)$ verifică propoziția b din lemă. \square

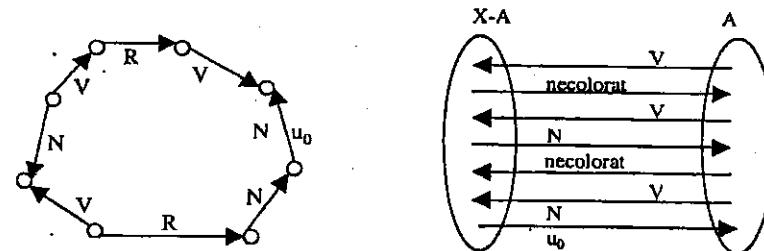


Fig.4.2.1.

Corolarul 4.2.1. Toate arcele unui graf aparțin fie unui circuit elementar fie unui cocircuit elementar.

Demonstrație:

Se colorează toate arcele grafului în negru, se aplică repetat lema lui Minty alegând pe rând ca arc u_0 fiecare arc al grafului. Deoarece graful are numai arce negre ciclurile și cociclurile a căror existență este asigurată de lema lui Minty sunt circuite respectiv cocircuite elementare. \square

Teorema 4.2.1. Fie φ un vector cu m componente nenegative. φ este flux într-un graf G dacă și numai dacă $\exists \lambda_1, \dots, \lambda_k \geq 0$ astfel încât $\varphi = \lambda_1 \bar{\mu}^1 + \dots + \lambda_k \bar{\mu}^k$, unde $\bar{\mu}^1, \dots, \bar{\mu}^k$ sunt vectorii asociați circuitelor elementare grafului.

Demonstrație:

Suficiență

Un vector φ care este combinație liniară cu scalari nenegativi ai unor fluxuri (vectori) cu componente nenegative (într-un circuit toate arcele au același sens) este flux cu componentele negative.

Necesitatea

Fie φ un flux în G cu componente negative. Dacă $\varphi \neq 0$ să considerăm graful G' obținut din G prin eliminarea arcelor u cu $\varphi_u < 0$. Deoarece fluxul în G' este nul înseamnă că G' conține cel puțin un ciclu elementar μ^1 . În virtutea legii conservării fluxului în vârfuri și pentru că toate componentele fluxului pe acest ciclu sunt strict pozitive respectivul ciclu este de fapt un circuit. Fie $\lambda_1 = \min\{\varphi_u \mid u \in \mu^1\} > 0$. Vectorul $\varphi' = \varphi - \lambda_1 \bar{\mu}^1$ este de asemenei un flux în G cu

componente nenegative, dar numărul componentelor nule a crescut. Repetăm procedeul precedent până când se ajunge la

$$\varphi - \lambda_1 \bar{\mu}^1 - \dots - \lambda_k \bar{\mu}^k = 0 \quad \varphi = \sum_{i=1}^k \lambda_i \bar{\mu}^i \text{ cu } \lambda_i > 0. \quad \square$$

4.3. Problema fluxului maxim într-o rețea de transport

Definiția 4.3.1. Se numește *rețea de transport* un graf orientat $G = (X, U)$ conex în care: există două vârfuri privilegiate unul s , numit *sursă* cu $\Gamma_s^{-1} = \Phi$, iar celălalt t , numit *destinație de transport*, cu $\Gamma_t = \Phi$ și fiecărui arc u i s-a atribuit un număr $c_u \geq 0$, numit *capacitatea sa*.

Vom considera doar cazul $c_u \in \mathbb{N}^*$.

Fiind dată o rețea de transport $G = (X, U)$ cu sursă s și destinație t notăm cu G^0 graful obținut din G prin adăugarea arcului (t, s) numit *arcul de return* și numerotat cu 0.

Definiția 4.3.2. Numim *flux de la s la t*, în rețea de transport G , vectorul $\varphi = (\varphi_1, \varphi_2, \dots, \varphi_m)'$ care verifică legea conservării fluxului în toate vârfurile cu excepția lui s și t , iar pentru acestea

$$\sum_{u \in \omega^+(s)} \varphi_u = \sum_{u \in \omega^-(t)} \varphi_u \text{ valoare notată cu } \varphi_0.$$

Cantitatea φ_0 se numește *valoarea fluxului* φ , se consideră fluxul corespunzător arcului se return 0.

Observația 4.3.1.

Dacă $\varphi = (\varphi_1, \varphi_2, \dots, \varphi_m)'$ este un flux de la s la t în G de valoare φ_0 atunci $\varphi = (\varphi_0, \varphi_1, \varphi_2, \dots, \varphi_m)'$ este un flux simplu în G^0 .

Problema fluxului maxim de la s la t într-o rețea de transport G este de a determina un flux $\varphi = (\varphi_0, \varphi_1, \varphi_2, \dots, \varphi_m)'$ în G^0 care verifică restricțiile de capacitate

$$0 \leq \varphi_u \leq c_u, \quad \forall u = 1, m$$

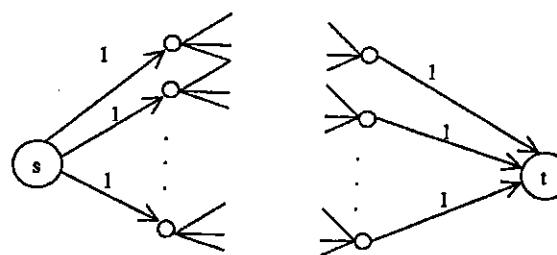
și valoarea fluxului adică φ_0 este maximă.

Un flux care verifică restricțiile de capacitate se numește *flux compatibil* cu capacitatele sau *flux compatibil* sau *flux admisibil*.

Exemplul 4.3.1.

Afectarea a p activități la q mașini.

Pentru a efectua p activități se dispune de q mașini. Cunoscând pentru fiecare mașină lista activităților ce le poate realiza, se pune problema de a determina numărul maxim de activități ce se pot desfășura simultan pe cele q mașini. Pentru a rezolva această problemă vom construi graful bipartit ce are ca vârfuri multimea de activități reunite cu multimea de mașini. Se va unui o activitate i cu o mașină j printr-un arc u dacă pe mașina j se poate realiza activitatea i . Atribuim arcelor orice capacitate $c_u \geq 1$. Adăugăm grafului două vârfuri. Unul s , din care vor pleca căte un arc spre fiecare vârf activitate. Acești arce le atribuim capacitatea 1. Celălalt vârf t , în care sosește căte un arc din fiecare vârf mașină și acestor arce le atribuim tot capacitatea 1. Orice flux de la s la t în acest graf are componente 0 și 1 și corespunde unei afectări și invers. Problema afectării revine, deci, la a găsi un flux maxim, cu componente întregi (booleene), de la s la t în graful de mai jos.



4.4. Tăietură într-o rețea de transport

Definiția 4.4.1. Fiind dată o rețea de transport G numim *tăietură*, ce separă pe s de t , o mulțime de arce de forma $\omega(A)$ (cociclu), unde $A \subset X$ și $s \in A$ și $t \notin A$.

Definiția 4.4.2. *Capacitatea tăieturii* este suma capacitaților arcelor care compun cocircuitul $\omega^*(A)$.

Dacă arcele au și limite inferioare ale capacitatei atunci prin capacitatea tăieturii se înțelege diferența dintre suma limitelor superioare a capacitateilor arcelor din $\omega^*(A)$ și suma limitelor inferioare a capacitateilor arcelor din $\omega(A)$.

Lema 4.4.1. *Valoarea maximă a fluxului de la s la t compatibil cu capacitatele intr-o rețea de transport nu depășește capacitatea nici unei tăieturi.*

Demonstrație:

Fie $\varphi = (\varphi_0, \varphi_1, \varphi_2, \dots, \varphi_m)'$ și $\omega(A)$ o tăietură oarecare un flux oarecare în G^0 . Cum în G^0 este verificată legea conservării fluxului în vârfuri deducem

$$\varphi_0 + \sum_{u \in \omega(A)} \varphi_u = \sum_{u \in \omega^*(A)} \varphi_u$$

de unde

$$\varphi_0 = \sum_{u \in \omega^*(A)} \varphi_u - \sum_{u \in \omega(A)} \varphi_u.$$

Deoarece $0 \leq \varphi_u \leq c_u, \forall u \in U$ obținem

$$\varphi_0 \leq \sum_{u \in \omega^*(A)} c_u. \quad \square$$

Corolar 4.4.1. $\max \varphi_0 \leq \min \{ \sum_{u \in \omega^*(A)} c_u \mid A \subset X, s \in A, t \notin A \}$.

Corolar 4.4.2. *Dacă φ este un flux de la s la t și $\omega(A)$ este o tăietură ce separă pe s de t, astfel încât valoarea fluxului este egală cu capacitatea tăieturii atunci φ este un flux maxim și tăietura are capacitate minimă.*

Definim *capacitatea unui drum într-o rețea de transport* ca fiind capacitatea minimă a arcelor care compun drumul.

Pentru orice rețea de transport mulțimea fluxurilor compatibile este nevidă. Despre problema fluxului maxim într-o rețea de transport spunem că are soluție dacă valoarea fluxului maxim este finită.

Corolar 4.4.3. *O condiție necesară și suficientă pentru ca problema fluxului maxim să aibă soluție este ca să nu existe drum de capacitate infinită de la s la t.*

Demonstrație:

Necesitatea

Este evident că dacă ar exista un drum de capacitate infinită atunci fluxul nu este maxim, deoarece pe arcele acelui drum fluxul poate crește oricără de mult.

Suficiență

Dacă nu există drum de capacitate infinită între s și t atunci graful $G' = (X, U')$ cu $U' = \{u \in U \mid c_u = \infty\}$ nu este conex. Fie A mulțimea celor vârfuri din X pentru care există drum de la s la ele în G' . Atunci arcele din $\omega^*(A)$ au toate capacitați finite. Conform corolarului 3.4.1 deducem

$$\max \varphi_0 \leq \sum_{u \in \omega^*(A)} c_u < \infty.$$

În continuare presupunem că această condiție este îndeplinită totdeauna.

Teorema 4.4.1. Teorema Ford-Fulkerson

Valoarea maximă a fluxului de la s la t într-o rețea de transport $G = (X, U)$ este egală cu capacitatea minimă a tăieturilor care separă pe s de t.

Demonstrație:

Fie $\varphi = (\varphi_0, \varphi_1, \varphi_2, \dots, \varphi_m)'$ un flux maxim în G^0 compatibil cu capacitatele c_u . Asociem lui fluxului φ colorarea următoare a arcelor lui G^0 :

- arcul 0 (de return) îl colorăm în negru;
- arcele u cu $\varphi_u = 0$ le colorăm în negru;
- arcele $u \neq 0$ cu $0 < \varphi_u < c_u$ le colorăm în roșu;
- arcele $u \neq 0$ cu $\varphi_u = c_u$ le colorăm în verde.

Deoarece fluxul φ este compatibil nu vor fi arce incolore. Considerăm $u_0 = 0$ și aplicăm lema lui Minty. Dacă am fi în cazul a al lemei, adică există un ciclu μ negru, roșu și verde ce trece prin 0 cu toate arcele negre în același sens, sensul lui 0, și toate arcele verzi în sens contrar. Fie μ vectorul asociat ciclului μ astfel încât $\mu_0 \rightarrow 1$. Atunci prin adăugarea fluxului $\epsilon \mu$ cu ϵ suficient de mic se obține un flux de valoare superioară, lui φ , dar cum φ este un flux maxim înseamnă că ne situăm în cazul b al lemei lui Minty. Deci există un cociclu $\omega(A)$ negru și verde fără arce roșii cu $s \in A$ și $t \notin A$ cu toate arcele din $\omega^*(A)$ verzi, iar cele din $\omega(A)$ negre (în particular $0 \in \omega^*(A)$). Prin urmare

$$\varphi_0 = \sum_{u \in \omega^+(A)} \varphi_u - \sum_{u \in \omega^-(A)} \varphi_u = \sum_{u \in \omega^+(A)} c_u.$$

Adică s-a găsit o tăietură $\omega(A)$ de capacitate φ_0 cum orice tăietură are capacitatea mai mare sau egală cu φ_0 înseamnă că această tăietură are capacitate minimă. \square

Observația 4.4.1.

Colorarea arcelor din teorema 4.4.1 împarte mulțimea arcelor în trei submulțimi. Fiecare submulțime conține arce care au proprietatea că valoarea fluxului compatibil suportă o anume modificare pentru ca fluxul obținut să fie tot un flux compatibil. Astfel pe arcele negre fluxul poate numai să crească, pe arcele verzi fluxul poate numai să scadă, iar pe arcele roșii fluxul poate fie să crească fie să scadă.

4.5. Algoritmul lui Ford-Fulkerson

Cu acest algoritm se poate afla fluxul maxim într-o rețea de transport. Pentru prezentarea algoritmului este necesar să definim mai întâi graful ecarturilor.

Definiția 4.5.1. Pentru un graf G fiecărui flux compatibil $\varphi = (\varphi_1, \dots, \varphi_m)$ și se atașează *graful ecarturilor* $G(\varphi) = (X, U(\varphi))$ care are aceeași mulțime de vârfuri ca și G , iar mulțimea arcelor $U(\varphi)$ se obține astfel:

pentru fiecare arc $u = (i, j)$ din G se adaugă în $U(\varphi)$ arcul $u^+ = (i, j)$ dacă $\varphi_u < c_u$ și $u^- = (j, i)$ dacă $\varphi_u > 0$.

Arcelor din $U(\varphi)$ li se atribuie capacitațile $c_u - \varphi_u$ pentru u^+ și φ_u pentru u^- .

Capacitațile atribuite arcelor din graful ecarturilor se numesc *capacități reziduale*. Capacitatea unui drum din graful ecarturilor (minimul capacitaților reziduale ale arcelor ce compun drumul) se mai numește *capacitatea reziduală a drumului* respectiv.

Teorema 4.5.1. Fie φ un flux de la s la t în rețeaua de transport G compatibil cu capacitațile și fie $G(\varphi)$ graful ecarturilor asociat lui φ . O condiție necesară și suficientă pentru ca fluxul φ să fie maxim este să nu existe un drum de la s la t în $G(\varphi)$.

Demonstrație:

Necesitatea

Demonstrăm cu metoda reducerii la absurd. Fie φ un flux maxim și presupunem că există un drum μ de la s la t în $G(\varphi)$. Drumului μ îi corespunde un drum între s și t în G astfel încât toate arcele negre sunt orientate în sensul de la s la t și toate arcele verzi în sensul de la t la s (se folosește colorarea de la teorema 4.4.1.). Prin adăugarea arcului de return 0 drumului μ se obține un ciclu în graful

G^0 . Aceasta înseamnă că φ nu este flux maxim, deoarece $\varphi + \varepsilon \xrightarrow{\rightarrow} \mu$ cu $\varepsilon > 0$ suficient de mic este de asemenea un flux de la s la t compatibil cu capacitațile și are valoarea strict mai mare decât φ_0 , adică o contradicție.

Suficiența

Dacă φ este un flux pentru care nu există drum de la s la t în $G(\varphi)$, atunci în mod necesar suntem în cazul b) al lemei lui Minty (utilizând colorarea de la teorema 4.4.1.). Alegem $A = \{i \in X \mid \exists$ drum de la s la $i\}$ și atunci $\omega^+(A)$ are numai arce verzi și pentru acestea $\varphi_u = c_u$, iar $\omega^-(A)$ are numai arce negre și pentru acestea $\varphi_u = 0$. De aici se obține:

$$\varphi_0 = \sum_{u \in \omega^+(A)} \varphi_u - \sum_{u \in \omega^-(A)} \varphi_u = \sum_{u \in \omega^+(A)} c_u.$$

Deci s-a precizat o tăietură care are capacitatea egală cu evaluarea fluxului și prin urmare fluxul φ este de valoare maximă. \square

Folosind teorema 4.5.1 se poate elabora algoritmul următor:

Algoritmul 4.5.1. Algoritmul lui Ford-Fulkerson (cu graful ecarturilor)

(a) inițializări

$k := 0$;

Fie φ^0 un flux compatibil (de exemplu $\varphi^0 = (0, 0, \dots, 0)$);

(b) criteriul de stop

dacă există drum μ^k de la s la t în $G(\varphi^k)$

atunci *go to (c)*

alțel stop (φ^k este flux optim).

sfd;

(c) fie ε capacitatea reziduală a drumului μ^k ;

$$\varphi_u^{k+1} = \begin{cases} \varphi_u + \varepsilon, & u^+ \in \mu^k \text{ sau } u=0 \\ \varphi_u - \varepsilon, & u^- \in \mu^k \\ \varphi_u, & \text{altfel} \end{cases}, \text{ pentru } u \in U$$

sfc.

k:=k+1;
goto (b).

Depistarea drumului μ^k în graful ecarturilor $G(\varphi^k)$ și a valorii ε , necesare algoritmului lui Ford-Fulkerson, se poate face utilizând următoarea procedură de etichetare (fără a construi efectiv graful $G(\varphi^k)$):

Orice vîrf $j \in X$ primește o etichetă formată din (e_1, e_2, e_3) , unde

$$e_1 \in X \cup \{0\}; e_2 \in \{+, -\}; e_3 \in \mathbb{R}_+$$

cu semnificațiile:

- dacă $e_2 = +$ și $e_1 = i$, atunci în G există arcul $u = (i, j)$ și în $G(\varphi^k)$ există un drum μ de la s la j care are ca ultim arc pe $u^+ = (i, j)$ și care drum are capacitatea e_3 ;
- dacă $e_2 = -$ și $e_1 = i$, atunci în G există arcul $u = (j, i)$ și în $G(\varphi^k)$ există un drum μ de la s la j care are ca ultim arc pe $u^- = (i, j)$ și care drum are capacitatea e_3 .

Etichetarea începe prin etichetarea doar a vîrfului s cu $(0, \bullet, \infty)$. Iterația de bază a procedurii de etichetare constă în:

Repetă

k := 0

pentru i ∈ X etichetăt

dacă există j ∈ X neetichetat astfel încât u = (i, j) ∈ U și φ_u < c_u

atunci j primește eticheta (i, +, min{e_3(i), c_u - φ_u}); k := 1

sfd;

dacă există j ∈ X neetichetat astfel încât u = (j, i) ∈ U și φ_u > 0,

atunci j primește eticheta (i, -, min{e_3(i), φ_u}); k := 1

sfd;

până când (s-a etichetat vîrful t) sau (k = 0);

Algoritmul de etichetare, de fapt, determină un arbore de rădăcină s al rețelei de transport. Dacă algoritmul de etichetare se termină prin etichetarea lui t , atunci în acel arbore există un singur drum de la s la t . Respectivul drum notat $\mu^k = \{d_p, d_{p-1}, \dots, d_1\}$, de la $s = d_p$ la $t = d_1$ necesar algoritmului, se poate descoperi pe cale inversă din valorile e_1 ale etichetelor începând cu $e_1(t)$. Adică,

p := I; d_I := t;
cătinp d_p ≠ execută
p := p+1;

$$d_p := e_1(d_{p-1});$$

Valoarea lui ε , capacitatea reziduală a drumului μ^k , este $e_3(t)$. Precizăm că deși pot fi etichetate multe vîrfuri ale rețelei G nu toate acestea aparțin drumului care conține toate arcele pentru care valoarea fluxului se modifică prin adunarea sau scăderea unui ε .

Cealaltă cauză a încheierii procedurii de etichetare, adică $k = 0$, conduce la concluzia "nu există drum în graful ecarturilor $G(\varphi^k)$ ".

Așa cum se observă varianta algoritmului care folosește procedura de etichetare nu necesită construirea grafului ecarturilor $G(\varphi^k)$ ci doar etichetarea vîrfurilor care se face utilizând rețea de transport G dată. Dăm în continuare o descriere compactă a acestei variante a algoritmului.

Algoritmul 4.5.2. Algoritmul lui Ford-Fulkerson (cu procedura de etichetare)

(a) inițializări

k := 0;
Fie φ un flux compatibil (de exemplu $\varphi = (0, 0, \dots, 0)$);

(b) criteriul de stop

Repetă

k = 0

pentru i ∈ X etichetăt

dacă există j ∈ X neetichetat astfel încât u = (i, j) ∈ U și φ_u < c_u

atunci j primește eticheta (i, +, min{e_3(i), c_u - φ_u}); k := 1

sfd;

dacă există j ∈ X neetichetat astfel încât u = (j, i) ∈ U și φ_u > 0,

atunci j primește eticheta (i, -, min{e_3(i), φ_u}); k := 1

sfd;

până când (s-a etichetat vîrful t) sau (k = 0);

dacă k = 0 atunci împărește "fluxul", φ , "de valoare", φ_0 "este flux maxim"

stop

sfd

(c) fie $\varepsilon := e_3(t)$;

j := t;

cătinp j ≠ execută

i := e_1(j);

dacă e_2(j) = + atunci φ_i := φ_i + ε, unde u = (i, j)

altfel φ_i := φ_i - ε, unde u = (j, i)

sfd;

j := i;

sfc;

φ_0 := φ_0 + ε

goto (b).

Deoarece capacitatele arcelor sunt numere naturale și fluxul de pornire se alege tot cu componente numere naturale deducem că în aplicarea algoritmului capacitatele reziduale sunt tot numere naturale. Prin urmare la fiecare pas valoarea fluxului crește cu un număr natural ($\epsilon \in \mathbb{N}^*$). În consecință după un număr finit de pași se obține valoarea maximă a fluxului.

Exemplul 4.5.1.

Vom aplica algoritmul 4.5.1. a lui Ford-Fulkerson pentru a determina fluxul maxim de la vârful 1 la vârful 3 în graful din figura 4.5.1.

În tabelul de mai jos dăm pentru fiecare iterație k valorile φ^k ale fluxurilor. Pe aceeași linie apar și drumul μ^k și valoarea ϵ calculată pentru drumul μ^k , care se folosesc pentru a calcula fluxul de pe linia următoare. Pe tabel s-au precizat arcele grafului atât prin indicele componentelor fluxurilor aleator ales de noi, cât și prin extremitățile lor. Pentru a urmări algoritmul am dat și grafurile ecarturilor pentru cele 6 iterării, precizăm că pentru $k = 0$ graful ecarturilor este chiar rețeaua de transport din fig 4.5.1.

φ_1	φ_2	φ_3	φ_4	φ_5	φ_6	φ_7	φ_8			
k	$(1,5)$	$(5,2)$	$(1,2)$	$(1,3)$	$(1,4)$	$(2,4)$	$(2,3)$	$(4,3)$	μ^k	ϵ
0	0	0	0	0	0	0	0	0	{1,2,4,3}	14
1	0	0	14	0	0	14	0	14	{1,5,2,3}	11
2	11	11	14	0	0	14	11	14	{1,4,3}	5
3	11	11	14	0	5	14	11	19	{1,2,3}	3
4	11	11	17	0	5	14	14	19	{1,3}	6
5	11	11	17	6	5	14	14	19	nu există	

Valoarea maximă a fluxului este 39 egală cu fluxul ce ieșe din vârful sursă și cu valoarea fluxului ce intră în vârful destinație, adică $39 = 11 + 17 + 6 + 5 = 14 + 6 + 19$. Tăieturi de capacitate egală cu valoarea fluxului maxim sunt date de $A = \{1, 5\}$.

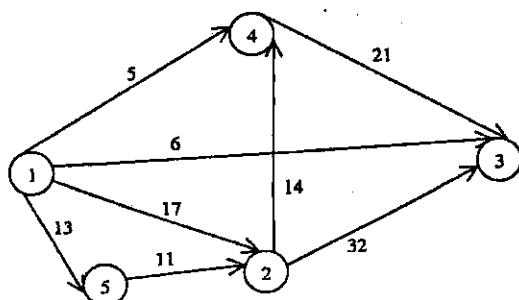
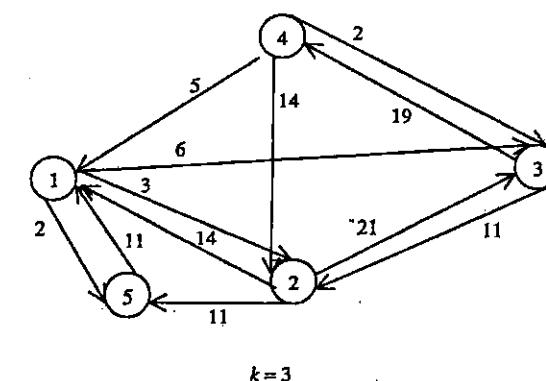
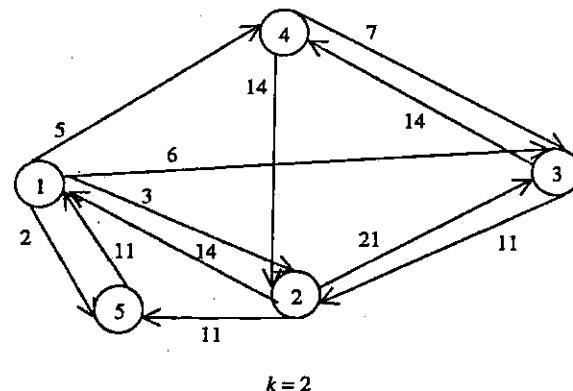
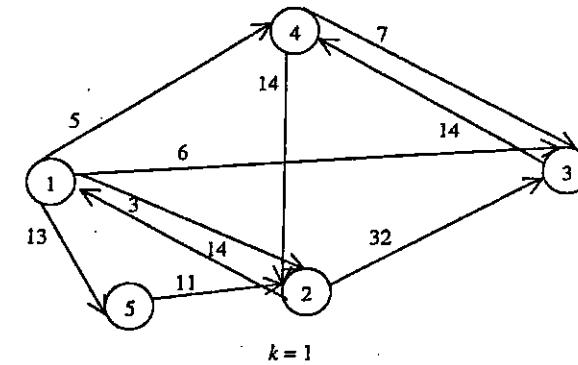
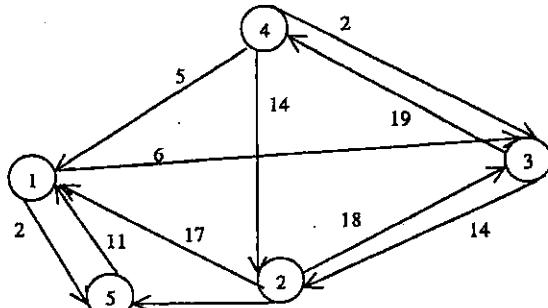
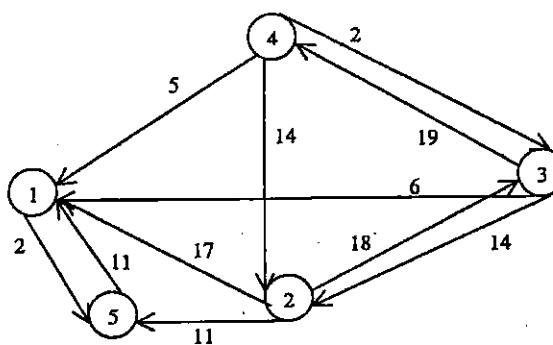


Fig. 4.5.1.





$k = 4$



$k = 5$

În rezolvarea problemei nici un drum din graful ecarturilor nu a folosit arce de tipul u^+ , exemplul următor oferă această posibilitate.

Depistarea drumului u și a valorii ε corespunzătoare lui se poate face fără a construi grafele $G(\varphi)$ prin construirea (marcarea vârfurilor) unei arborescențe de rădăcină ε .

Dacă vârful i a fost marcat ($p(i) \neq 0$) atunci se va marca vârful j , adică $p(j) := i$ dacă $u = (i, j) \in U$ și $\varphi_u < c_u$ caz în care $\varepsilon(j) := \min\{\varepsilon(i), c_u - \varphi_u\}$ sau dacă $u = (j, i) \in U$ și $\varphi_u > 0$ caz în care $\varepsilon(j) := \min\{\varepsilon(i), \varphi_u\}$.

În final dacă $p(t) \neq 0$ (s-a marcat vârful t), atunci fluxul φ nu este de valoare maximă și modificarea componentelor sale se face cu procedura:

$$i := p(t); j := t; \varepsilon := \varepsilon(t);$$

Repetă

Dacă $u = (i, j) \in U$ atunci $\varphi_u = \varphi_u + \varepsilon$ dacă;

Dacă $u = (j, i) \in U$ atunci $\varphi_u = \varphi_u - \varepsilon$ dacă;
 $j := i; i := p(j);$
 până când $j = s$;

Precizăm că informația conținută de vectorul p se poate interpreta și dacă $i, j \in X$ și

- dacă $p(i) = 0$, atunci vârful i nu aparține arborescenței și nu există drum în $G(\varphi)$ de la s la i ,
- dacă $p(i) = j$ atunci în arborescență vârful j este părinte al vârfului i și în graful ecarturilor $G(\varphi)$ există un drum de la s la i , care are ca ultim arc arcul u^+ dacă $u = (j, i) \in U$ respectiv arcul u^- dacă $u = (i, j) \in U$.

Exemplul 4.5.2.

Să se determine un flux maxim în rețeaua de transport dată în figura 4.5.2.

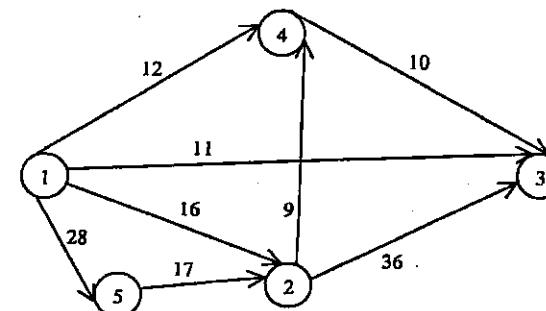


Fig. 4.5.2.

Să observăm mai întâi că rețeaua dată are sursă 1 și destinație 3 (ca și rețeaua de la exemplul precedent). Rezolvarea problemei o vom prezenta cu ajutorul a două tabele. Unul conține vârfurile grafului și etichetările făcute la diferite etape în evoluția algoritmului. Celălalt conține valorile în evoluție ale vectorului flux φ . Precizăm că celor două tabele li s-au completat succesiv și alternativ liniile odată cu evoluția variabilei k (numărul iterării). Deoarece la fiecare iterare primul vârf etichetat este vârful sursă (pentru exemplul nostru acesta este vârful 1) și de fiecare dată eticheta acestuia este $(0, *, \infty)$ în tabel nu vom mai trece coloanele corespunzătoare acestui vârf.

vt. k	2			3			4			5			Ordinea etichetăril vârfurilor
	e ₁	e ₂	e ₃	e ₁	e ₂	e ₃	e ₁	e ₂	e ₃	e ₁	e ₂	e ₃	
0	1	+	16	4	+	9	2	+	9				1, 2, 4, 3.
1	1	+	7	4	+	1	1	+	12	1	+	28	1, 2, 5, 4, 3.
2	5	+	17	2	+	17				1	+	28	1, 5, 2, 3.
3	1	+	7	2	+	7							1, 2, 3.
4	4	-	9	2	+	9	1	+	11				1, 4, 2, 3.
5				1	+	11							1, 3.
6							1	+	2	1	+	11	Vârful 3 e nemarcat

Tabelul precedent poate fi întocmit și în alte moduri, important este faptul că indiferent de modul de întocmire al său se obține un flux (poate altul decât cel obținut) dar care are aceeași valoare care este egală cu valoarea maximă a fluxului în rețeaua dată (valoare unică). În continuare dăm tabelul care precizează fluxurile obținute pentru rețeaua dată folosind etichetările și drumurile precizate de tabelul de mai sus.

k	φ ₁	φ ₂	φ ₃	φ ₄	φ ₅	φ ₆	φ ₇	φ ₈	μ ^k = {d ₁ , ..., d ₁₁ }	ε
0	(1,5)	(5,2)	(1,2)	(1,3)	(1,4)	(2,4)	(2,3)	(4,3)	{1,2,4,3}	9
1	0	0	0	0	0	0	0	9	{1,4,3}	1
2	0	0	9	0	0	9	0	10	{1,5,2,3}	17
3	17	17	9	0	1	9	17	10	{1,2,3}	7
4	17	17	16	0	1	9	24	10	{1,4,2,3}	9
5	17	17	16	0	10	0	33	10	{1,3}	11
6	17	17	16	11	10	0	33	10	nu există	

Deci, valoarea maximă a fluxului în rețeaua dată este 54 și este egală cu capacitatea tăieturii dată de multimea vârfurilor marcate la ultima iterație, adică $A = \{1, 5, 4\}$, care este o tăietură de capacitate minimă. Pentru că $\omega^+(A) = \{(5, 2), (1, 2), (1, 3), (4, 3)\}$, capacitatea tăieturii $\omega(A)$ este egală cu $c_2 + c_3 + c_4 + c_5 = 17 + 16 + 11 + 10 = 54$.

4.6. Modelarea problemei orarului ca flux în grafe

Problema orarului în câteva cuvinte se poate formula astfel:

Se dău:

- mulțimea profesorilor;
- mulțimea disciplinelor;
- mulțimea formațiunilor (grupa 222, anul 1mf, etc.);

- mulțimea sălilor;
- zilele și intervalele orare când pot fi planificate activități.

Se cere:

Înănd cont de toate interdependențele dintre cele cinci tipuri de date, să se realizeze un orar așa încât toate elementele celor patru mulțimi să fie "mulțumite", adică:

Toți profesorii să fie programați conform cu numărul de ore săptămânal a fiecăruiu, toate disciplinele să fie predate conform programei de învățământ și formațiile să-și desfășoare activitățile în săli adevărate (numărul participanților să fie mai mic sau egal cu capacitatea sălii, dotarea sălii să corespundă activităților care se vor desfășura).

Un orar poate fi interpretat ca o succesiune de informații grupate sau ca o bază de date din care se poate extrage: programul claselor, al sălilor, al profesorilor, etc. O înregistrare (poziție a orarului) ne imaginăm că ar conține nu neapărat în această ordine informațile: ziua, ora, disciplina, sala, anul (grupa sau clasa), profesorul. Prin realizarea unor codificări potrivite ziua și ora se pot identifica din valoarea variabilei ora, un mod de codificare ar fi să numerotăm intervalele de timp ale unei săptămâni de lucru. De exemplu considerând că deși toate activitățile (curs, seminar sau laborator) durează 2 ore (120 minute), vom numerota momentele de început al acestor activități care pot fi orele pare între 8 și 18 ale zilelor de luni până vineri sau sâmbătă. Vor fi astfel 35 de valori, dar orele pot fi altfel codificate (60 minute sau alte intervale ale zilelor). Pe de altă parte anul, grupă sau semigrupă de asemenea printr-o codificare corespunzătoare se poate identifica prin cunoașterea valorii codului (câmpului) formațiune. Un tip similar de codificare se poate realiza pentru disciplină + forma de activitate la cea disciplină (curs, seminar sau laborator) informație pe care o vom identifica din valoarea câmpului disciplină. Să mai precizăm că dacă pentru o formă (an, grupă sau semigrupă) se cunoaște profesorul se poate deduce și care este disciplina și invers din perechea formă-disciplină se deduce profesorul care intră în respectiva înregistrare.

Activitatea de întocmire a orarului o putem privi (modela) ca problemă de flux maxim într-un graf special. Astfel, elementele care intră în orar le putem considera ca vârfuri ale grafului. Astfel, ar exista cinci tipuri de vârfuri și anume ora, sala, formațiunea, profesorul și disciplina. De aici o înregistrare a orarului se poate interpreta ca fiind un drum în graf și care conține câte vârf din fiecare tip. Aceste drumuri pot fi drumurile în baza cărora crește valoarea fluxului într-o rețea de transport. Modelul ales trebuie să asigure atingerea valorii maxime a fluxului doar în momentul completării orarului.

Un prim model pe care îl propunem pentru realizarea orarului asociază acestei probleme un graf multipartit în felul următor:

- într-o partitură avem profesorii (cadre didactice sau laboranți sau maștrii-instructori, etc.);
- într-o partitură avem disciplinele (eventual pe forma de desfășurare curs laborator, seminar, etc.);
- într-o altă partitură avem formațiunile (anii de studiu, grupele, semigrupele, clasele, etc.);
- mai avem o partitură cu sălile (curs, seminar, laborator, ateliere, săli de sport);
- și o ultimă partitură cu orele (momentele posibile pentru a începe activitățile).

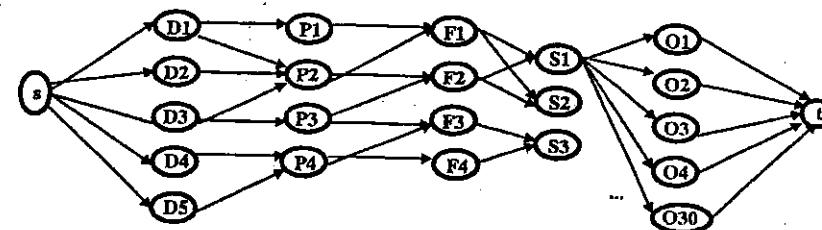
În acest caz pentru fiecare profesor, disciplină, de studiu, formăjune și sală vom avea câte un vârf al grafului, iar pentru fiecare ora (de fapt sunt 120 minute sau 60 minute) de curs seminar sau laborator vom avea câte un vârf al grafului în partitura orelor astfel:

- luni ora de la 10-12 va fi vârful cu numărul 3;
- miercuri ora de la 14-16 va fi vârful cu numărul 16

În total vor fi 30 de vârfuri, adică 5 zile/săptămână câte 6 ore pe zi. Dacă se dorește rezolvarea problemei pentru învățământul preuniversitar se vor considera 60 de vârfuri pentru cele 60 de ore dintr-o săptămână (5 zile x 12 ore pe zi) numerotate de la 1 la 60 sau după cum stabilește cel ce modeleză problema împreună cu conducerea instituției (școală, liceu, facultate, etc.).

Se vor conecta (va exista arc între) vârfurile care definesc posibilități (compatibilități) de desfășurare a unor activități. Adică: fiecare disciplină este predată de niște profesori, fiecare profesor desfășoară activități cu anumite formații de studiu (an, grupă, semigrupă sau clasă), fiecare formăjune își poate desfășura activități în anumite săli și fiecare sală poate fi disponibilă la anumite ore din săptămână.

Pe lângă cele cinci parturi ale grafului vom avea încă două vârfuri unul sursă și celălalt destinație, pentru transformarea grafului într-o rețea de transport. O astfel de rețea asociată unei probleme de acest tip este prezentat (schițat) în figura următoare:



Așa cum se vede și în figura anterioară, activitățile de la fiecare disciplină (curs, seminar, laborator, etc.) care va fi predată de anumiți profesori va fi legată de vârfurile definite de aceștia printr-un arc, fiecare profesor ce trebuie desfășoară activități cu o anumită formăjune va fi legat de formăjunea respectivă printr-un arc, fiecare formăjune va fi legată la rândul ei printr-un arc de acele săli care îndeplinește condițiile impuse de numărul de locuri sau dotare (laborator, sală de sport, etc.), iar fiecare sală va fi legată de acele ore la care este disponibilă. Vârfurile adăugate grafului vârful sursă (s) și vârful destinație (t) vor fi conectate de toate vârfurile asociate disciplinelor respectiv de toate vârfurile corespunzătoare orelor.

Notări:

$G = (V, E)$ - rețeaua problemei noastre, unde:

V : mulțimea vârfurilor grafului;

E : mulțimea arcelor.

$R = (G, s, t, c)$ - rețea de transport, unde:

s este vârful special numit *sursa* rețelei;

t un alt vârf special numit *destinația* rețelei;

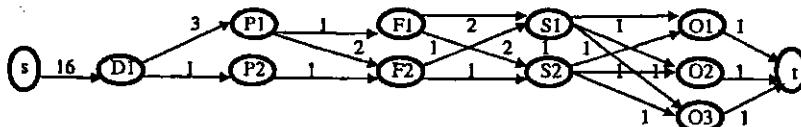
c este o funcție $c : E \rightarrow \mathbb{R}_+$, $c(i)$ se numește *capacitatea* arcului i și limitează valoarea fluxului pe arcul respectiv.

În rețeaua R definită mai sus, capacitatele arcelor vor primi valori în modul următor:

- dacă un profesor trebuie să predea un anumit număr de ore dintr-o disciplină oarecare, atunci arcul ce leagă vârfurile corespunzătoare va avea capacitatea egală cu acel număr de ore;

- dacă un profesor trebuie să desfășoare un număr de ore cu o anumită formăjune, atunci arcul ce leagă vârfurile corespunzătoare va avea capacitatea egală cu acel număr de ore;
- capacitatea fiecărui arc ce leagă o formăjune de o sală va fi egală cu cel puțin suma capacitațiilor arcelor ce intră în vârful formăjune din care pleacă arcul respectiv de aceea valoarea poate fi numărul maxim de ore dintr-o săptămână (30 sau 60 sau altfel);
- capacitatea arcelor ce leagă săliile de ore vor fi egale cu 1;
- capacitatea fiecărui arc ce pornește de la sursă avea ca valoare cel puțin numărul de ore săptămânale prevăzute de programa de învățământ pentru disciplina în care intră acel arc;
- capacitatea arcelor ce ajung la destinație vor avea valoarea 1.

Un exemplu pentru o disciplină, doi profesori, două formăjuni, două săli și trei ore, exemplu edificator pentru a se înțelege felul în care funcția de capacitate primește valori:



Se observă că fiecare drum de la s la t în rețeaua R ar putea fi o poziție în orar pentru că drumul precizează disciplina, profesorul, formăjunea, sala și ora și invers fiecare poziție a orarului este un drum al rețelei de transport R . Cum în orar trebuie plasate toate orele tuturor disciplinelor valoare egală cu capacitatea tăieturii $\omega^*(A)$ definită de $A = \{s\}$.

Deci, pentru a stabili un orar vom determina cu algoritmul lui Ford-Fulkerson descriis mai sus fluxul maxim în rețeaua R .

Fluxul de valoare maximă de la s la t , se determină prin alegerea succesivă a unor drumuri în grafele ecarturilor. Fiecare astfel de drum de la s la t care a fost selectat este o poziție din orar. Pentru fiecare arc al acestor drumuri, conform algoritmului, se va scădea din capacitatea sa capacitatea reziduală a aceluia drum. În etapa de obținere a fluxului maxim în orar au fost plasate toate pozițiile. În acest moment orarul nostru este complet. Prin această metodă se va determina o variantă de orar, care depinde de ordinea în care se aleg drumurile din graful ecarturilor.

Dacă se dorește o optimizare a variantei de orar întocmite, atunci propunem ca în rețeaua de transport definită mai sus să se mai adauge un alt tip de arce pe care le numim arce inhibitoare. Aceste arce definesc restricțiile impuse orarului. De exemplu dacă un anume cadru didactic nu este liber la anumite ore, atunci se va adăuga un arc inhibitor între vârfurile profesor și ora corespunzătoare. Un alt exemplu este în anumite săli nu se pot desfășura anumite activități (discipline) și se va adăuga între aceste vârfuri un arc inhibitor. Rolul arcelor inhibitori este în aplicarea algoritmului lui Ford-Fulkerson nu se vor selecta drumurile cu proprietatea că între două vârfuri ale sale există un arc inhibitor. Prin urmare rețeaua se va reprezenta prin cele două tipuri de arce, de exemplu două matrici de adiacență sau o matrice cu valori egale cu capacitatea arcelor sau 0 dacă nu există arc respectiv valoarea -1 pentru existența unui arc inhibitor. Fiecare rezolvitor decide asupra modului de reprezentare a rețelei definite de modul descris aici pentru problema orarului.

4.7. Extensiile ale algoritmului lui Ford-Fulkerson

1. Algoritmul 4.5.1 se poate aplica și pentru grafele neorientate după ce se înlocuiește fiecare muchie cu o pereche de arce care au sensuri opuse și capacitatele egale cu capacitatea muchiei.
2. Se poate aplica algoritmul 4.5.1 și pentru grafe $G = (X, U)$ care au restricții de capacitate în vârfuri $c(i)$, $i \in X$.

Aceste restricții pentru problemele din practică înseamnă spațiu de depozitare (cazare) a fluxului în vârfuri. Se va aplica algoritmul lui Ford-Fulkerson pentru graful G' obținut din G prin următoarele:

Se înlocuiește fiecare vârf i al grafului cu două vârfuri i' și i'' , arcelor din G care intră în i li se vor asocia în G' arce care vor intra în i' , iar arcelor care ies din i li se vor asocia arce care ies din i'' . Acestor arce, din G' , li se atribuie capacitați egale cu cele ale arcelor, din G , cărora le-au fost asociate. În G' se mai adaugă arcele de tipul (i', i'') cărora li se atribuie capacitațiile $c(i)$. Vârful i' se mai numește *intrarea* în i , iar vârful i'' se numește *ieșirea* din i .

3. Fluxuri pe grafe dinamice

Modelul acestui tip de probleme de transport este următorul. Din diferite puncte a_i (orașe, baze, etc.) pleacă vehicule, ce transportă ceva, urmând ca ele să ajungă la o destinație comună b . Dacă există o legătură directă între a_i și a_j atunci se dă numărul c_{ij} de vehicule care pot pleca din a_i către a_j în unitatea de timp, precum și timpul t_{ij} necesar pentru a ajunge din a_i în a_j . Se

mai dă numerele s_i de vehicule disponibile în a_i la momentul 0 și numărul maxim c_i de vehicule care pot staționa în vârful a_i . Se cere să se organizeze traficul pentru ca într-un interval de timp dat T , numărul vehiculelor care sosesc în b să fie cel mai mare posibil. Această problemă se poate reduce la o problemă de flux maxim într-un graf G' , numit *graful dezvoltat*. Să presupunem că t_0 și T sunt multiplii ai aceleiași unități de timp. Vom discretiza problema considerând momentele de timp $0, 1, 2, \dots, T$. Pentru fiecare a_i se asociază în G' vârfurile a_i^t , $t = 0, T$, să presupunem că $i = 1, n$. De asemenei lui b îi asociem vârfurile b^t , $t = 0, T$. Se introduc arcele (a_i^t, a_i^{t+1}) , $t = 0, T - 1$ și $i = 1, n$ cărora li se asociază capacitatea c_i . Pentru punctele a_i și a_j care au legătură directă se vor asocia arcele $0, t = 0, T - t_{ij}$ și acestora li se atribuie capacitatea c_{ij} . Se adaugă sursa s și destinația d precum și arcele (s, a_i^0) , $i = 1, n$ cărora li se atribuie capacitatele s_i și arcele (b^t, d) , $t = 0, T$ cu capacitatea ∞ . Fluxul maxim în G' determină traficul rutier care face ca la destinația b să ajungă un număr maxim de vehicule.

Exemplul 4.7.1.

Se consideră trei puncte în teren cu legăturile și timpii precizați ca în graful din figura 4.7.1. Considerând $T = 4$ graful dezvoltat apare ca în figura 4.7.2.

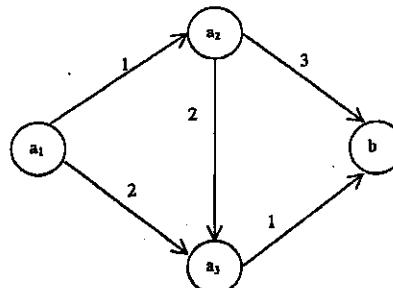


Fig. 4.7.1.

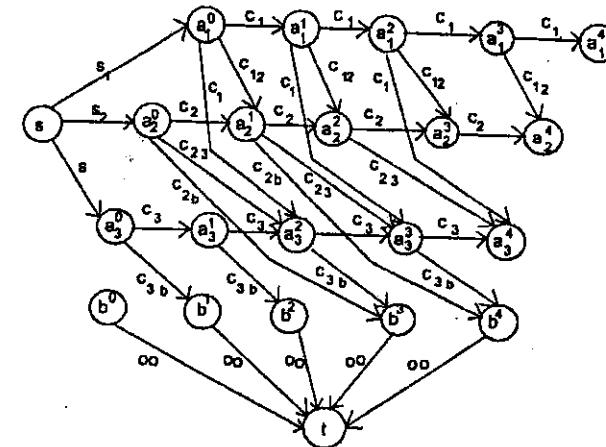


Fig. 4.7.2.

- O altă extensie a algoritmului lui Ford-Fulkerson consideră cazul în care marginea inferioară a fluxului pe fiecare arc u nu este 0 ci o valoare oarecare $b_u \geq 0$. Pentru exemplificare să revenim la problema afectării celor p activități la q mașini (exemplul 4.3.1.). Dacă cerem în plus ca toate activitățile sau un număr dat de activități să se desfășoare simultan atunci este suficient ca în graful G^0 arcului de return să-i impunem o limitare inferioară a fluxului, b_0 egală cu numărul de activități ce dorim să se desfășoare simultan.

Algoritmul 4.5.1 este valabil și pentru această problemă cu precizarea că pentru un arc $u = (i, j)$ din G în graful ecarturilor $G(\varphi)$ construiește arcul $u^+ = (i, j)$ dacă $\varphi_u < c_u$ cu capacitatea reziduală $c_u - \varphi_u$ și arcul $u^- = (j, i)$ dacă $b_u < \varphi_u$ cu capacitatea reziduală $\varphi_u - b_u$. În acest caz apare însă problema determinării unui flux compatibil pentru pornire (initializare). Este evident că un flux compatibil cu capacitatele nu există pentru orice graf.

4.8. Fluxuri de cost minim

Un alt caz particular al problemei de flux în grafe este problema de transport. Particularitatea constă în faptul că $X = X_s \cup X_d$, adică graful conține numai vârfuri sursă și vârfuri destinație în consecință, este un graf bipartit complet. Să presupunem că $|X_s| = m$ și $|X_d| = n$.

Prin urmare în vîrful $i \in X_s$ există (se produce) o cantitate a_i din produsul de transportat (resursă), iar în $j \in X_t$ se solicită cantitatea b_j de resursă. Transportul pe arcul (i, j) se face cu o cheltuială c_{ij} unități bănești pe unitatea de resursă. Problema cere să se determine un program de transport (x_{ij}) , pentru $(i, j) \in X_s \times X_t$, astfel încât costul total de transport care este

$$\sum_{i=1}^m \sum_{j=1}^n c_{ij} x_{ij}, \text{ notat } f(X),$$

să fie minim. Vom presupune că problema este echilibrată, adică $\sum_{i=1}^m a_i = \sum_{i=1}^n b_i$.

Dacă problema nu este echilibrată atunci se adaugă un vîrf sursă sau un vîrf destinație fictiv care să producă respectiv să ceară diferența de resursă cu cheltuieli de transport 0. De asemenea presupunem că datele problemei (a_i, b_j, c_{ij}) sunt numere naturale. Datele problemei se pot prezenta și sub forma unor tabele (matrici cu valorile arcelor grafului bipartit complet), pe care unii algoritmi le folosesc și pentru înscrierea rezultatelor. Un astfel de tabel arată astfel:

	1	2	.	.	j	.	.	n	
1	c_{11}	c_{12}	.	.	c_{1j}	.	.	c_{1n}	a_1

i	c_{i1}	c_{i2}	.	.	c_{ij}	.	.	c_{in}	a_i

m	c_{m1}	c_{m2}	.	.	c_{mj}	.	.	c_{mn}	a_m
	b_1	b_2	.	.	b_j	.	.	b_n	

Pentru rezolvarea unei probleme de transport se utilizează algoritmul lui Dantzig, cunoscut în mai multe variante. Acest algoritm are două etape și anume etapa de inițializare și etapa de determinare a soluției optime. În etapa de inițializare se determină o soluție admisibilă care în etapa a doua este îmbunătățită succesiv până se obține o soluție optimă. Sistemul care exprimă condițiile problemei are mn necunoscute și $m + n$ ecuații, dintre care $m + n - 1$ sunt liniar independente datorită condiției de echilibru. Din acest motiv numărul necunoscutele bazice este $m + n - 1$, celelalte fiind nebazice au valoarea 0. Deci o soluție are cel mult $m + n - 1$ valori x_{ij} diferite de 0. În tabel se vor trece doar valorile variabilelor bazice (unele pot avea valoarea 0), celelalte valori nu se trec în tabel și se consideră că sunt 0. Cele $m + n - 1$ arce corespunzătoare variabilelor bazice din graful bipartit complet, care are $m + n$ vîrfuri, formează un arbore parțial al grafului.

1. Determinarea unei soluții inițiale

Se poate face cu metoda colțului de nord-vest (sau a diagonalei) sau cu metoda costurilor minime.

a. Metoda colțului de nord-vest

```

i:=1;
j:=1;
căt timp  $a_i b_j \neq 0$   $x_{ij} := \min(a_i, b_j);$ 
     $a_i := a_i - x_{ij};$ 
     $b_j := b_j - x_{ij};$ 
    dacă  $a_i = b_j$ 
        atunci  $x_{ij+1}$  sau  $x_{i+1,j} := 0$ 
        dacă  $i < m$  atunci  $i := i + 1;$ 
                     $j := j + 1;$ 
    sfđ;
    altfel
        dacă  $b_j = 0$  atunci  $j := j + 1$ 
            altfel  $i := i + 1$ 
    sfđ;
    sfđ;
    sfđc.

```

b. Metoda costului minim

```

I := {(i,j) | i=1,m și j=1,n};
căt timp  $I \neq \emptyset$  fie  $(k,l)$  din  $I$  cu  $c_{kl} = \min(c_{ij} | (i,j) \in I)$ 
    dacă  $a_k < b_l$  atunci  $x_{kl} := a_k$ ;  $b_l := b_l - x_{kl};$ 
         $I := I - \{(k,l)\} | j=1,n\};$ 
    altfel
        dacă  $a_k = b_l$  atunci  $x_{kl} := a_k$ ;
            alege  $(k,j)$  sau  $(i,l)$  din  $I$ ;
             $x_{kl}$  sau  $x_{kl} := 0$ ;
             $I := I - \{(k,j)\} | j=1,n\} - \{(i,l) | i=1,m\};$ 
        altfel  $x_{kl} := b_l$ ;  $a_k := a_k - x_{kl};$ 
             $I := I - \{(i,l) | i=1,m\};$ 
    sfđ;
    sfđ;
    sfđc.

```

Observația 4.8.1.

Așa după cum s-a mai precizat variabilele care primesc valoarea 0 în algoritm se consideră că fac parte dintre variabilele bazice în timp ce variabilele care nu primesc nici o valoare, în algoritmii de inițializare, se consideră că au de asemenea valoarea 0 dar sunt variabile nebazice. Corespondența variabilelor x_{ij} și celulele (i, j) , din tabel, se împart în celule bazice și nebazice.

2. Determinarea soluției optime

Mai întâi definim noțiunea de ciclu atașat unei celule nebazice în tabelul de date al problemei de transport. Fie (i, j) o celulă nebazică, prin ciclu atașat ei înțelegem o succesiune de celule ce pornește din (i, j) , trece numai prin celule bazice, ajunge la celula (i, j) , și oricare două celule consecutive se află pe aceeași linie sau pe aceeași coloană fără ca trei celule consecutive să fie pe aceeași linie sau coloană.

(a) Pentru (i, j) celulă nebazică execută

determină un ciclu atașat ei;

marchează celulele din ciclul alternativ cu + și - începând cu celula (i, j)

marcată cu +;

determină d_{ij} = suma algebrică a costurilor din celulele

marcate având ca semn marcajul celulei;

sfp;

Fie $d_M = \min\{d_{ij} \mid (i, j)$ celulă nebazică și μ ciclul atașat celulei $(k, l)\}$;

Dacă $d_M \geq 0$ atunci stop (s-a obținut soluția optimă) sfđ;

Fie $\theta = \min\{x_{ij} \mid (i, j)$ celulă marcată cu - din ciclul $\mu\}$;

Pentru (i, j) celulă din μ marcată cu - execută

$$x_{ij} := x_{ij} - \theta$$

sfp;

Pentru (i, j) celulă din μ marcată cu + execută

$$x_{ij} := x_{ij} + \theta$$

sfp;

goto (a).

Teorema 4.8.1. Dacă x^2 este programul de transport obținut de algoritmul de mai sus din programul de transport x^1 , atunci are loc $f(x^2) = f(x^1) + \theta d_M$. Deci, dacă $d_M \geq 0$, atunci $f(x^2) \geq f(x^1)$, iar dacă $d_M \leq 0$, atunci $f(x^2) \leq f(x^1)$.

Un alt algoritm pentru obținerea soluției optime este cel cunoscut sub numele de metoda potențialelor care are însă o complexitate mai mare decât algoritmul prezentat, cu care de altfel se asemănă foarte mult.

Exemplul 4.8.1.

Vom rezolva problema de transport dată în tabelul:

15	40	5	20	100
22	33	9	16	60
28	6	23	25	150
80	90	60	80	

Deoarece problema dată este echilibrată ($100 + 60 + 150 = 80 + 90 + 60 + 80$) trecem direct la etapa de determinare a unei soluții de pornire (soluție admisibilă inițială).

În tabelele de mai jos sunt date soluțiile inițiale obținute atât cu metoda nord-vest, tabelul a, cât și cu metoda costului minim, tabelul b. În celulele bazice ale tabelelor s-au trecut și valorile variabilelor bazice x_{ij} sub diagonala fiecărei celule.

15	40	5	20
80	20		
22	33	9	16
	60		
28	6	23	25
	10	60	80

a.

15	40	5	20
40		60	
22	33	9	16
		60	
28	6	23	25
40	90		20

b.

Să observăm că pentru programul de transport dat de tabelul a costul transportului este

$$80 \times 15 + 20 \times 40 + 60 \times 33 + 10 \times 6 + 60 \times 23 + 80 \times 25 = 7420.$$

Acum vom aplica algoritmul de obținere a soluției optime pornind de la tabelul a. Calculăm valorile d_{ij} pentru celulele nebazice și obținem:

$$d_{13} = 5 - 40 + 6 - 23 = -52; \quad d_{14} = 20 - 40 + 6 - 25 = -39;$$

$$d_{21} = 22 - 33 + 40 - 15 = 14; \quad d_{23} = 9 - 33 + 6 - 23 = -41;$$

$$d_{24} = 16 - 33 + 6 - 25 = -36; \quad d_{31} = 28 - 15 + 40 - 6 = 47.$$

Valoarea minimă este $d_{13} = -52$. Deoarece $d_{13} \leq 0$ și ciclul atașat celulei $(1, 3)$ trece și prin celulele $(1, 2), (3, 2)$, și $(3, 3)$ obținem $\theta = \min\{20, 60\} = 20$ și facem modificările:

$$x_{13} = x_{13} + \theta = 20, \quad x_{12} = x_{12} - \theta = 0, \quad x_{32} = x_{32} + \theta = 30, \quad x_{33} = x_{33} - \theta = 40.$$

Noul tabel este:

15	40	5	20
80		20	
22	33	9	16
60			

28	6	23	25
30	40	80	

Pentru acăst program de transport costul este

$$f(x) = 80 \times 15 + 20 \times 5 + 60 \times 33 + 30 \times 6 + 40 \times 23 + 25 \times 80 = \\ = 6380 + d_{kl} \times \theta = 7420 + 52 \times 20.$$

Calculând valorile d_{ij} pentru actualul program de transport se obține:

$$\begin{array}{ll} d_{12} = 40 - 6 + 23 - 5 = 52; & d_{14} = 20 - 5 + 23 - 25 = 13; \\ d_{21} = 22 - 33 + 6 - 23 + 5 - 15 = -38; & d_{23} = 9 - 23 + 6 - 33 = -41; \\ d_{24} = 16 - 25 + 6 - 33 = -36; & d_{31} = 28 - 23 + 5 - 15 = -5. \end{array}$$

Deci $d_{kl} = d_{23} = -41 < 0$. Algoritmul continuă prin a stabili valoarea $\theta = \min\{40, 60\} = 40$

și apoi

$$x_{23} = x_{23} + \theta = 40, \quad x_{33} = x_{33} - \theta = 0, \quad x_{32} = x_{32} + \theta = 70, \quad x_{22} = x_{22} - \theta = 20.$$

Deci noul program de transport este cel dat în tabelul:

15	40	5	20
80		20	
22	33	9	16
20	40		

28	6	23	25
70	80		

Acest program de transport se face cu costul

$$f(x) = 6380 - 41 \times 40 = 4740.$$

În continuare algoritmul conduce la valorile:

$$d_{12} = 40 - 33 + 9 - 5 = 11; \quad d_{14} = 20 - 5 + 9 - 33 + 6 - 25 = -28;$$

$$d_{21} = 22 - 33 + 40 - 15 = 14; \quad d_{24} = 16 - 33 + 6 - 25 = -36;$$

$$d_{31} = 28 - 6 + 33 - 9 + 5 - 15 = 36; \quad d_{33} = 23 - 6 + 33 - 9 = 41;$$

Deci $d_{kl} = d_{24} = -36 < 0$, ceea ce înseamnă că algoritmul continuă.

$$\theta = \min\{20, 80\} = 20,$$

iar

$$x_{24} = x_{24} + \theta = 20, \quad x_{22} = x_{22} - \theta = 0, \quad x_{32} = x_{32} + \theta = 90, \quad x_{34} = x_{34} - \theta = 60.$$

Programul de transport obținut este cel prezentat în tabelul:

15	40	5	20
80		20	
22	33	9	16
40		20	

28	6	23	25
90		60	

și are costul $f(x) = 4740 - 36 \times 20 = 4020$.

Calculând valorile d_{ij} pentru celelele nebazice se obține:

$$d_{12} = 40 - 5 + 9 - 16 + 25 - 6 = 47; \quad d_{14} = 20 - 16 + 9 - 5 = 8;$$

$$d_{21} = 22 - 9 + 5 - 15 = 3; \quad d_{22} = 33 - 16 + 25 - 6 = 36;$$

$$d_{31} = 28 - 25 + 16 - 9 + 5 - 15 = 0; \quad d_{33} = 23 - 25 + 16 - 9 = 5.$$

Cum $d_{kl} = 0$ înseamnă că programul obținut, care are valorile diferite de zero

$$x_{11} = 80, x_{13} = 20, x_{23} = 40, x_{24} = 20, x_{32} = 90 \quad și \quad x_{34} = 60,$$

este de cost minim.

Să observăm însă valoarea minimă d_{kl} este exact 0, de aici deducem că există cel puțin un alt program de transport care are costul egal 4020 (costul minim). Acest program este dat în tabelul următor.

15	40	5	20
40		60	
22	33	9	16
		60	
28	6	23	25
40	90		20

emplul 4.8.2.

Să se determine un program de transport de cost minim pentru problema de transport următoare:

5	4	3	100
4	3	5	60
3	5	4	50
80	20	60	

Observăm că această problemă este neechilibrată, deci trebuie să o echilibrem și pentru că $100 + 60 + 50 > 80 + 20 + 60$ se mai adaugă o coloană (un șif destinație către care costul transportului din fiecare sursă este 0 unități bănești = unitate de resursă). Cantitatea solicitată de acest vîrf destinație este 50 (diferența dintre ofertă și cerere). Astfel problema este:

5	4	3	0	100
4	3	5	0	60
3	5	4	0	50
80	20	60	50	

Aplicăm acum metoda colțului de nord-vest și se obține ca soluție de optimizare programul din tabelul următor:

5	4	3	0
80	20	60	
4	3	5	0
0		60	
3	5	4	0
		0	50

Așa cum se observă celulele (2, 2) și (3, 3) sunt celule bazice, dar putea fi aleasă ca celulă bazică în locul acestora celula (1, 3) respectiv (2, 4).

Etapa a doua, cea de determinare a unei soluții optime o lăsăm spre plăcerea cititorului. Pentru control dăm o soluție optimă, care are costul 510, în tabelul de mai jos.

5	4	3	0
		60	40
4	3	5	0
30	20		10
3	5	4	0
50			

5. CUPLAJE ÎN GRAFE

În acest capitol vom considera doar grafe neorientate.

Definiția 5.1. Fiind dat un graf simplu $G = (X, U)$, cuplaj al său este orice submulțime de muchii K care are proprietatea că oricare două muchii nu sunt adiacente.

Un vîrf $i \in X$ este vîrf saturat de cuplajul K dacă și numai dacă cuplajul are o muchie incidentă vîrfului i .

Un cuplaj se numește cuplaj perfect dacă saturează toate vîfurile grafului. Un cuplaj se numește cuplaj maxim dacă în mulțimea cuplajelor grafului el are număr maxim de muchii.

Fie K un cuplaj, iar μ un lanț elementar al grafului $G = (X, U)$. μ este lanț alternant relativ la K dacă muchiile sale alternează în K și în $U - K$. Un lanț alternant se numește lanț alternant crescător dacă ambele extremități ale sale sunt vîfurile nesaturate. Un lanț alternant crescător are lungime impară, iar numărul muchiilor sale din $U - K$ este cu 1 mai mare decât numărul de muchii din K .

În reprezentarea geometrică a grafului convenim ca muchiile ce fac parte din cuplaj să se reprezinte mai îngroșat.

Dacă K este un cuplaj și μ este un lanț alternant care dacă are vreo extremitate saturată atunci unica muchie din K incidentă acelei extremități face parte din μ , atunci $K' = K \Delta \mu = (K - \mu) \cup (\mu - K)$ este de asemenea un cuplaj. Această operatie, de schimbare a muchiilor îngroșate cu celelalte muchii din lanț o numim transfer de lungime cu lanțul alternant μ . O operatie de transfer de lungime cu un lanț alternant crescător mărește cardinalitatea cuplajului cu o unitate.

Lema 5.1. Fie $G = (X, U)$ un graf simplu, iar K și K' două cuplaje diferite ale lui G . Componentele conexe ale grafului parțial $H = (X, V)$, unde $V = K \Delta K'$ pot fi de trei tipuri:

- tipul 1: puncte izolate;
- tipul 2: ciclu elementar par;
- tipul 3: lanț elementar.

Demonstrație:

În H pentru orice vîrf i are loc $g(i) \leq 2$, aceasta din cauză că un vîrf oarecare este incident cel mult unei muchii din K sau cel mult unei muchii din K' .

Vom pune în evidență componentele conexe ale lui H prin determinarea vîfurilor fiecărei componente conexe cu ajutorul unui procedeu de selectare a vîfurilor. Selectăm mai întâi căte un vîrf i cu $g(i) = 0$, obținând astfel componente de tip 1 ale lui H .

Selectăm, apoi, un vîrf cu $g(i) = 1$, selectăm vîrful adjacent acestuia. Vom selecta succesiv, dintre vîfurile lui H neselectate vîrful adjacente, ultimului vîrf selectat până când vîrful selectat are gradul 1 (deci operația de selectare nu mai poate continua). Mulțimea vîfurilor selectate formează o componentă de tip 3 a lui H . Repetăm acest procedeu până când vîfurile neselectate au gradul 2 sau au fost selectate toate vîfurile grafului.

În continuare selectăm un vîrf i dintre cele neselectate, deci $g(i) = 2$. Vom selecta apoi succesiv vîrful adjacente, în H , ultimului vîrf selectat până când va fi selectat vîrful i (vîrful de la care s-a pornit). Această selectare este posibilă deoarece toate vîfurile neselectate, din H , au gradul 2. Mulțimea selectată este o componentă de tipul 2 a grafului H . Se repetă procedeul de punere în evidență a componentelor conexe ale lui H până când vor fi selectate toate vîfurile grafului. \square

Teorema 5.1. Teorema lui Berge [Ber57]

Un cuplaj K este maxim dacă și numai dacă nu există un lanț alternant crescător relativ la K .

Demonstrație:

Necesitatea

Vom utiliza metoda reducerii la absurd. Să presupunem că există un lanț μ alternant crescător relativ la cuplajul K . Fie $K' = K \Delta \mu$ deci $|K'| = |K| + 1 > |K|$ ceea ce contrazice maximalitatea lui K .

Suficiența

Fie K un cuplaj astfel încât nu există lanț alternant crescător și fie K' un cuplaj maxim. Conform primei părți a demonstrației nici relativ la K' nu există lanțuri alternante crescătoare. Considerăm acum graful $H = (X, K \Delta K')$. Deoarece relativ la ambele cuplaje nu există lanțuri alternante crescătoare înseamnă că dintre componentele posibile (lema 4.1) H are numai componente de tipul 3 de lungime pară. Deci $|K - K'| = |K' - K|$ și de aici $|K| = |K'|$, adică K este și el un cuplaj maxim. \square

Corolar 5.1. Un cuplaj $K \subset U$ care nu lasă mai mulți de un vîrf nesaturat este maxim.

Deoarece există cel mult un vârf nesaturat înseamnă că în graf nu există șiruri alternante crescătoare și deci cuplajul este maxim.

Lemă 5.2. Fie K un cuplaj maxim al lui $G = (X, U)$. Atunci orice cuplaj maxim K' al lui G se poate obține din K printr-o succesiune de transferuri de lungime cu lanțuri alternate pare două câte două disjuncte (unele lanțuri pot fi cicluri).

monstrărie:

Fie K' un cuplaj maxim și considerăm graful $H = (X, K \Delta K')$. Deoarece șirurile cuplajelor sunt maxime înseamnă că H nu are componente de tip 3 impare. Îcă pornind de la cuplajul K se fac transferuri de lungime cu fiecare componentă nevoie de tip 2 sau 3 se obține cuplajul K' . \square

Definiția 5.2. Fie $G = (X, U)$ un graf și $Y \subset X$ o mulțime de vârfuri ale sale. *Graful restrâns* al lui G relativ la Y , notat G/Y , este

$$G/Y = (X \cup \{y\} - Y, V)$$

unde

$$\begin{aligned} V = & \{(i, j) \mid i, j \in X - Y \text{ și } (i, j) \in U\} \cup \\ & \cup \{(i, y) \mid i \in X - Y, \exists k \in Y \text{ și } (i, k) \in U\}. \end{aligned}$$

Deci graful restrâns se obține din G prin înlocuirea subgrafului generat de adică a lui $G(Y)$, cu un pseudovârf y care se conectează cu alte vârfuri din G/Y că acestea sunt conectate în G cu vreun vârf din Y .

Un exemplu de graf restrâns pentru $Y = \{2, 4\}$ se dă în figura 5.1.

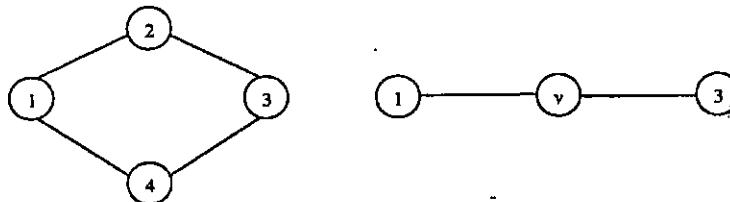


Fig. 5.1.

Teorema 5.2. Fie $Y \subset X$ mulțimea vârfurilor unui ciclu elementar impar μ . Dacă K_1 este un cuplaj al lui G/Y atunci există un cuplaj maxim K_Y al lui μ astfel încât $K = K_1 \cup K_Y$ este un cuplaj al lui G .

Demonstrație:

Cum K_1 este un cuplaj în G/Y înseamnă că cel mult o muchie a lui K_1 este incidentă (pseudo)vârfului y al lui G/Y .

Dacă nici o muchie a lui K_1 nu este incidentă lui y atunci lema este verificată dacă se alege orice cuplaj al lui μ de cardinal $(|Y| - 1)/2$, care este maxim deoarece lasă un singur vârf nesaturat.

Dacă există o muchie a lui K_1 incidentă lui y atunci muchia corespunzătoare ei din G este incidentă unui vârf k a lui Y . Vom alege cuplajul maxim K_Y al lui μ care lasă pe k nesaturat și deci $K_1 \cup K_Y$ este un cuplaj al grafului G . Să observăm că μ fiind impar orice cuplaj maxim al său are $(|Y| - 1)/2$ muchii și lasă exact un vârf nesaturat. \square

O consecință imediată a lemei 5.2 este:

Dacă există un lanț alternant crescător relativ la K_1 în G/Y atunci există un lanț alternant crescător în G relativ la cuplajul $K_1 \cup K_Y$.

Altfel spus, dacă K_1 nu este un cuplaj maxim al lui G/Y atunci $K_1 \cup K_Y$ nu este un cuplaj maxim al lui G . Cuplajul K_Y fiind cel din lema 5.2.

Exemplul 5.1.

Considerăm graful G din figura 5.2, în care alegem ciclul definit de $Y = \{5, 6, 7, 8\}$. În graful redus G/Y cuplajul $K_1 = \{(2, y)\}$ nu este un cuplaj maxim, deoarece lanțul alternant $\{1, 2, y, 3\}$ este crescător. Pentru K_1 se alege cuplajul maxim, din ciclul impar $G(Y)$, $K_Y = \{(7, 6), (4, 5)\}$ și deci nici $K_1 \cup K_Y = \{(2, 8); (7, 6); (4, 5)\}$ nu este maxim în graful G . Aceasta se deduce și din existența lanțului alternant crescător $\{1, 2, 8, 7, 6, 5, 4, 3\}$ în G relativ la $K_1 \cup K_Y$.

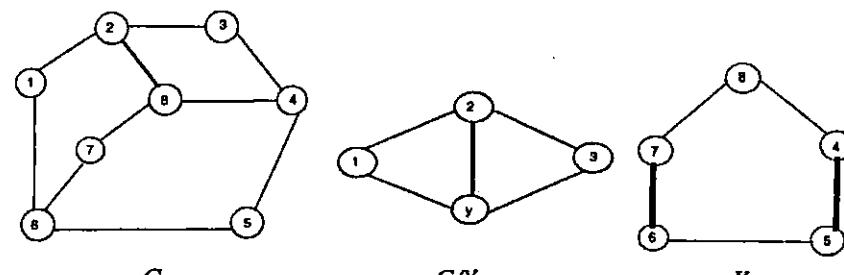


Fig. 5.2.

Lema 5.3. Fie K un cuplaj al lui G care lasă cel puțin două vârfuri nesaturate și fie $\mu = (Y, V)$ un ciclu elementar impar astfel încât $K\mu = K \cap V$ este maxim în μ . Atunci K este un cuplaj maxim al lui G dacă și numai dacă $K_1 = K - K\mu$ este un cuplaj maxim al lui G/Y .

Demonstrație:

Necesitatea

Rezultă din lema 5.2.

Suficiența

Vom utiliza metoda reducerii la absurd. Să presupunem că K_1 este un cuplaj maxim al lui G/Y , dar K nu este un cuplaj maxim al lui G . Notăm cu j_0 vârful lui μ nesaturat de $K\mu$ și presupunem că j_0 este unit cu un vârf i_0 nesaturat de K printr-un lanț alternant L_0 . Deoarece K_1 este maxim în G/Y atunci L_0 nu este lanț alternant crescător deci are lungime pară (eventual de lungime 0). Considerăm cuplajele K' și K'_1 obținute din cuplajele K respectiv K_1 prin transfer de lungime cu lanțul L_0 . Cum L_0 este par deducem $|K'| = |K|$ și prin urmare K' nu este maxim în G . Pe de altă parte K'_1 este un cuplaj în G/Y cu $|K'_1| = |K_1|$, deci el este maxim.

Vârful i_0 , care este nesaturat de K , este saturat de K' . Pe de altă parte vârful j_0 rămâne nesaturat și de K' , prin urmare în graful G/Y pseudovârful y este nesaturat de K'_1 .

Cum K' nu este maxim în G înseamnă că există două vârfuri a și b nesaturate de K' între care există un lanț L alternant crescător. Lanțul L trebuie să se intersecteze cu μ deoarece altfel L ar fi un lanț alternant crescător pentru K'_1 ceea ce ar contrazice maximalitatea lui K'_1 . Cel puțin un vârf dintre a și b diferă de j_0 . Să presupunem că acesta este a , deci $a \neq j_0$. Fie L_a porțiunea lanțului L cuprinsă între a și primul vârf din μ . În mod necesar muchia lui L_a incidentă lui μ nu este din K' , deoarece nici o muchie incidentă cu μ nu este din K' , deci ea nu aparține nici lui K'_1 . În G/Y lanțul alternant L_a conectează două vârfuri, a și y , nesaturate de K'_1 și este crescător, ceea ce contrazice maximalitatea lui K'_1 , astfel lema este demonstrată.

Fie $K \subset U$ un cuplaj al grafului $G = (X, U)$ și fie i_0 un vârf nesaturat. Dacă i_0 este singurul vârf nesaturat atunci K este maximal. Dacă există și alte vârfuri nesaturate, trebuie găsită o metodă care să permită găsirea unui lanț alternant crescător, de la i_0 spre un alt vârf nesaturat, sau să concluzioneze că un asemenea lanț nu există. Acest lucru se va face cu ajutorul unei proceduri care pornind de la graful G , cuplajul K și vârful i_0 decide dacă există sau nu un lanț alternant crescător cu extremitatea în i_0 . Procedura va construi un arbore alternant maximal de rădăcină i_0 . Pentru aceasta mai întâi introducem noțiunea de arbore alternant de rădăcină i_0 . În cazul depistării unui lanț alternant crescător se va face

un transfer de lungime cu acesta obținându-se un cuplaj de cardinal mai mare. Altfel, căutarea se face pentru următorul vârf nesaturat. Algoritmul reia iterația de bază până când relativ la fiecare dintre vârfurile nesaturate se știe că nu există lanț alternant crescător. Metoda căutată are deci, ca iterație de bază algoritmul de construire a arborilor altermanți ai unui graf asociat unui cuplaj dat. \square

Definiția 5.3. Un *arbore alternant* de rădăcină i_0 relativ la cuplajul K este un subgraf parțial $P = (Y, T)$ conex, fără cicluri al lui G , astfel încât:

- $i_0 \in Y$ este singurul vârf nesaturat al arborelui;
- pentru orice $j \in Y$ lanțul unic din P între i_0 și j , notat $L_P(j)$, este lanț alternant;
- lanțurile care conectează i_0 și un vârf terminal au un număr par de muchii.

Vârfurile din Y le numim *marcate*, iar cele din $X - Y$ nemarcate. Vârfurile $j \in Y$, marcate, se împart în două: unele pare iar celealte impare. Astfel dacă $L_P(j)$ are lungime pară spunem că vârful j este *par* (în particular i_0 este par), altfel j se numește *impar*.

Observația 5.1.

Într-un arbore alternant orice vârf terminal este vârf par și orice vârf i impar are $g(i) = 2$ fiind incident unei muchii îngroșate și unei muchii subțiri.

Construirea unui arbore alternant de rădăcină i_0

Deoarece algoritmul transformă graful și arboarele curent prin operații de reducere cu cicluri de lungime impară vom nota cu:

- $G' = (X, U)$ graful curent;
- $P = (Y, T)$ arboarele curent;
- $Y^0 \subset Y$ mulțimea vârfurilor arborelui marcate și pare.

Unele dintre vârfurile lui X , Y sau Y^0 pot fi pseudovârfuri.

a. inițializări

$$Y = Y^0 = \{i_0\}; \quad T = \Phi; \quad G' = G;$$

- Fie $P = (Y, T)$ arboarele alternant curent, dorim să mărim acest arbore. Pentru aceasta se deosebesc patru cazuri.

Cazul 1. $\exists i \in Y^0$ și \exists o muchie $u = (i, j)$ cu j nesaturat și nemarcat. Atunci lanțul $L' = L_p(i) \cup \{u\}$ este lanț alternant crescător în G . Stop (ieșirea A).

Cazul 2. $\exists i \in Y^0$ și $\exists u = (i, j)$ cu j saturat și nemarcat. Deci există și muchia $u' = (j, l)$ din K incidentă lui j . În acest caz se dezvoltă arborele alternant astfel:

$$T := T \cup \{u, u'\}; \quad Y := Y \cup \{j, l\}; \quad Y^0 := Y^0 \cup \{l\};$$

goto (b).

Cazul 3. $\exists i \in Y^0$ și $\exists u = (i, j) \in U - T$ cu $j \in Y^0$. Deci $T \cup \{u\}$ conține un ciclu unic μ , în mod necesar impar. Lanțurile $L_p(i)$ și $L_p(j)$ au cel puțin un vârf comun i_0 . Primul vârf comun, notat r , de la i spre i_0 este par, deoarece dacă $r = i_0$ este par și în caz că $r \neq i_0$ atunci $g(r) \geq 3$ deci nu poate fi impar. Astfel sublanțurile L_i și L_j dintre r și i respectiv r și j sunt de lungime pară, prin urmare $\mu = L_i \cup L_j \cup \{u\}$ este un ciclu de lungime impară pe care îl numim **orbită** a lui K , iar pe r îl numim **rădăcina orbitei**. În acest caz modificăm graful și arborele alternant astfel:

$$G' := G' / \mu; \quad P := P / \mu; \quad \text{goto (b).}$$

Cazul 4. Dacă nu ne aflăm în nici un caz precedent, adică nici un alt vârf nu mai poate fi marcat și nici o orbită nu se poate construi, atunci Stop, arborele P este complet construit și este maximal, procedura se termină, spunem, cu ieșirea B.

Teorema 5.2. Fie $X_1 \subset X$ mulțimea vârfurilor marcate ale lui G la momentul în care nici un marcat nu mai este posibil. Notăm $X_2 = X - X_1$, $G_1 = (X_1, U_1) = G(X_1)$ subgraful generat de X_1 , $G_2 = (X_2, U_2) = G(X_2)$ subgraful generat de X_2 , $K_1 = K \cap U_1$, și $K_2 = K \cap U_2$. Cuplajul K este maxim în G dacă și numai dacă K_2 este un cuplaj maxim în G_2 .

monstrătie:

Mai întâi să observăm că nu există $u = (i_1, i_2) \in K$ astfel încât $i_1 \in X_1$, $i_2 \in X_2$. Aceasta din cauză că dacă $i_1 \in X_1$ atunci el este un vârf saturat și marcat, și singura muchie din K incidentă lui i_1 trebuie să facă parte din arborele alternant. Această muchie fiind u deducem că și cealaltă extremitate a sa, adică i_2 , este un vârf marcat. În concluzie $K = K_1 \cup K_2$.

Necesitatea

Utilizăm metoda reducerii la absurd. Dacă K_2 nu este un cuplaj maxim în G_2 atunci relativ la K_2 există un lanț alternant crescător în G_2 . Deoarece $X_1 \cap X_2 = \emptyset$ acel lanț este lanț alternant crescător și în G , relativ la K , ceea ce este în contradicție cu maximalitatea lui K .

Suficiența

O demonstrăm tot cu metoda reducerii la absurd. Presupunem că K_2 este un cuplaj maxim în G_2 dar K nu este cuplaj maxim în G . Deci există un lanț alternant crescător L în G relativ la K . Să notăm extremitățile sale, care sunt vârfuri nesaturate, cu j și k . Remarcăm că i_0 este rădăcina arborelui alternant. Nu putem avea $i_0 = j$ și nici $i_0 = k$, deoarece lanțul L ar fi fost depistat de algoritmul de construire a arborelui alternant și algoritmul s-ar fi terminat cu ieșirea A. Din motive similare j și k nu fac parte din X_1 . Deci $j, k \in X_2$. Pe de altă parte $K \cap U_1 \neq \emptyset$, căci altfel L ar fi un lanț alternant crescător în G_2 relativ la K_2 care este cuplaj maxim. Fie $l \in L \cap X_1$, deci există un lanț L_1 în arborele alternant de la i_0 la l . Să notăm cu l' primul vârf din L întâlnit în parcurgerea lui L_1 de la i_0 spre l . Cum $l' \in L \cap L_1$ și nu este extremitate a lanțului alternant L înseamnă că l' este marcat impar (altfel L nu ar fi lanț alternant). Pe de altă parte L fiind impar înseamnă că unul dintre sublanțurile sale, cel de la l' la j sau cel de la l' la k (care sunt tot lanțuri alternante) are lungime pară. Cum j și k sunt vârfuri nesaturate, iar l' marcat impar înseamnă că l' aparține sublanțului par. Prin compunerea acestuia cu sublanțul lui L_1 de la i_0 la l' se obține un lanț alternant crescător de la i_0 la j sau k (vezi figura 5.3). Prin urmare de la i_0 la una dintre extremitățile lanțului L există un lanț alternant crescător care ar fi trebuit depistat de cazul 1 al algoritmului de construire a arborelui alternant. Cum aceasta nu s-a întâmplat înseamnă că presupunerea făcută este falsă și deci K este un cuplaj maxim pentru G . \square

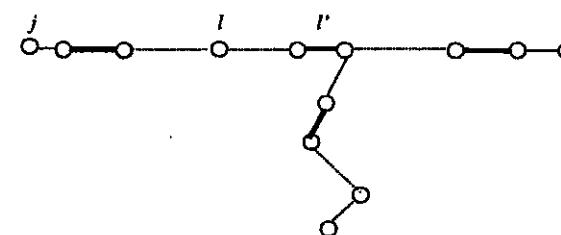


Fig. 5.3.

În continuare vom prezenta **algoritmul lui Edmonds** [Edm65a], pentru determinarea unui cuplaj maxim într-un graf oarecare.

Algoritmul 5.1. Algoritmul lui Edmonds

Determină un cuplaj maxim într-un graf $G = (X, U)$.

- se alege K , un cuplaj de pornire, poate fi $K = \emptyset$; fie $G' := G$;
- dacă nu există două vârfuri nesaturate în G'
atunci stop (K este cuplaj maxim al lui G)
altfel fie i_0 un vârf nesaturat;
construim arborele alternant de rădăcină i_0
- ieșirea B (s-a determinat un arbore alternant complet)
alege următorul la vârf nesaturat
cu G' subgraful generat de vârfurile nemarcate
goto (b);
- ieșirea A (s-a găsit un lanț alternant crescător μ)
se face transfer de lungime cu lanțul μ ;
construiesc cuplajul induș în graful inițial
prin spargerea pseudovârfurilor (alegând din
aceste cuplaje maxime conform lemei 4.2);
se sterg marcajele;
goto (b).

Exemplul 5.2.

Vom aplica algoritmul lui Edmonds pentru graful din figura 4.4 pornind de la cuplajul $K = \{(2, 3), (4, 5), (6, 7)\}$.

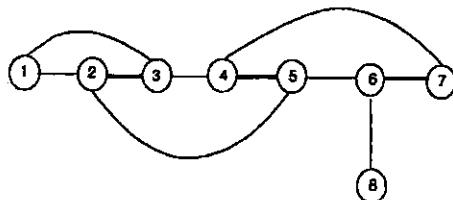
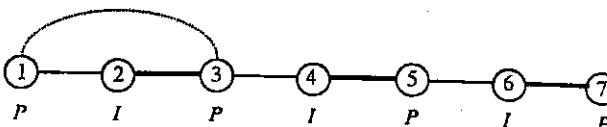


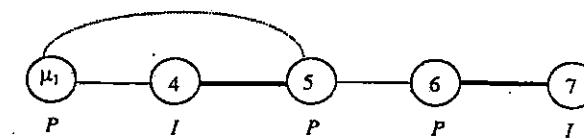
Fig. 5.4.

Deoarece există două vârfuri, 1 și 8, nesaturate vom construi arborele alternant de rădăcină $i_0 = 1$.

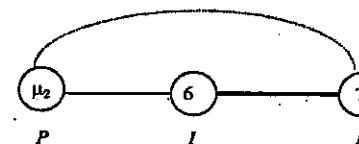
Aplicând în mod repetat în cazul 2, se marchează par vârfurile 1, 3, 5, și 7 respectiv impar vârfurile 2, 4 și 6. În continuare vom reprezenta muchiile arborelui prin linii continue, iar muchiile din G' prin linii întrerupte.



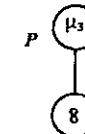
Muchiia (1,3) conectează două vârfuri pare și conform cazului 3 se depistează orbita $\mu_1 = \{1, 2, 3\}$ și facem o restrângere atât a grafului G' cât și a arborelui alternant.



Muchiia ($\mu_1, 5$) conectează de asemenei două vârfuri marcate par și conform cazului 3 mai facem o reducere a lui G' și a arborelui alternant cu orbita $\mu_2 = \{\mu_1, 4, 5\}$.



Analog situațiilor precedente se face o reducere cu orbita $\mu_3 = \{\mu_2, 6, 7\}$.



Apoi se depistează conform cazului 1 lanțul alternant crescător $\{\mu_3, 8\}$ și vom face un transfer de lungime cu acest lanț și vom construi cuplajul induș în graful inițial prin spargerea pseudovârfurilor ca în figura 5.5. S-a obținut cuplajul $\{(1, 3), (2, 5), (4, 7), (6, 8)\}$. Acest cuplaj lasă mai puțin de două vârfuri nesaturate prin urmare este un cuplaj maxim.

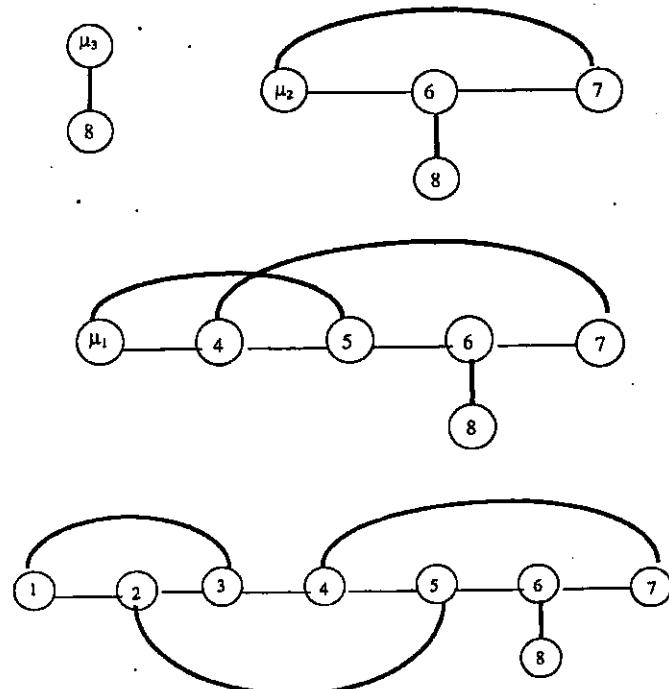


Fig. 5.5.

BIBLIOGRAFIE

- [App76] Appel K., *The proof of the four-color problem*, New Scientist, 1976
- [Bel58] Bellman R.E., *On a routing problem*, Quat. Appl. Math. 16(1958) pp.269-271
- [Ber57] Berge C., *Two theorems in graph theory*, Proc. Nat. Ac. Sciences, USA, 43(1957) pp.842
- [Ber69] Berge C., *Teoria grafurilor și aplicațiile ei* (Introducere elementară), Ed. Didactică și Pedagogică, București, 1969
- [Ber70] Berge C., *Graphes et hypergraphes*, Dunod, Paris, 1970
- [BLW76] Biggs N.L., Loyd E.K., Wilson R.J., *Graph Theory 1736-1936*, Oxford University Press, 1976
- [Cro92] Croitoru C., *Tehnici de bază în Optimizarea Combinatorie*, Editura Univ. "Al. I. Cuza", Iași, 1992
- [Dij59] Dijkstra E.W., *A note on two problems in connexion with graphs*, Numerische Mathematik 1(1959) pp.269-271
- [Dir52] G. A. Dirac, *Some theorems on abstract graphs*, Proc. London Math. Soc. Vol.2.(3)(1952) pp.69-81
- [Edm65a] Edmonds J., *Paths, trees and flowers*, Canad.J. of Math., 17(1965) pp.449-467
- [Edm65b] Edmonds J., *Maximum matching and a polyhedron with 0-1 vertices*, Journal Res. Nat. Bureau Standards, Vol.69-B (1965) no. 1-2, pp125-130
- [Eul36] Euler L., *Solutis Problematis ad Geometriam Situs Pernautis*, Comentarii Academiae Scientiarum Imperialis Petropolitane, 1736, 8
- [Flo62] Floyd R.W., *Algorithm 97, Shortest path*, Comm. A.C.M. 5(1962) pp.345
- [For56] Ford L.R.Jr., *Network flow theory*, The Rand Corporation, 1956, pp.293
- [Ful61] Fulkerson D.R., *An out-of kilter method for minimal cost flow problem*, Journal S.I.A.M. vol.9(1961) no.1, pp.18-27

- [Ful71] Fulkerson D.R., *Blocking and anti-blocking pairs of polyhedra*, Mathematical Programming 1(1971)pp.168-194
- [GiH64] Gilmore P.C., Hoffman A.J., *A characterization of comparability graphs and of interval graphs*, Canad. J. of Math. 16(1964), pp.87-90
- [GoM79] Gondran M., Minoux M., *Graphes et Algorithmes*, Ed. Eyrolles, Paris, 1979
- [HoT74] Hopcroft J.E., Trajan R.E., *Efficient Planarity Testing*, J.A.C.M. 21(1974), no.4, pp.549-568
- [Hu68] Hu T.C., *A decomposition algorithm for shortest paths in a network*, Operation Research, 16(1968), pp.91-102
- [Kle67] Klein M., *A primal method for minimal cost flow problems*, Management Science, 14(1967), pp.205-212
- [Kru56] Kruskal J.B., *On the shortest spanning subtree of a graph and the traveling salesman problem*, Proc. Am. Math. Soc. 71(1956), pp.48-50
- [KTT79] Kasa Z., Tarjia C., Tâmbulea L., *Culegere de probleme de teoria grafelor*, Lito. Univ. "Babeş-Bolyai", Cluj-Napoca, 1979
- [Kur30] Kuratowski K., *Sur le problème des courbes gouches en topologie*, Fund. Math., 15(1930), pp.271-283
- [Lov72] Lovasz L., *Normal hypergraphs and the perfect graph conjecture*, Discrete Math., 2(1972), pp.1389
- [Min66] Minty G.J., *On the axiomatic foundations of the theories of directed linear graphs, electrical networks and network programming*, Journal of Mathematics and Mechanics, vol.15 (1966), no.3, pp.485-520
- [Mol85] Moldovan Gr., *Bazele Informaticii II*, Lito. Univ. "Babeş-Bolyai", Cluj-Napoca, 1985
- [Moo59] Moore E.F., *The shortest path through a maze*, Proc. Inf. Symp. on Theory of switching, University 30, Harvard University Press, 1959
- [MRC59] Malcolm D.G., Roseboom J.H., Clark C.E., Fazar W., *Application of a technique for research and development program evaluation*, Operation Research, vol.7(1959), no.5
- [Ore60] O.Ore, *Note on Hamiltonian circuits*, Amer. Math. Monthly, 67(1960), pp.55
- [Pri57] Pim R.C., *Shortest connection networks and some generalizations*, Bell. Syst. Techn. J., 36(1957), pp.1389-1401
- [Ros66] Rosenstiehl P., *L'Arbre minimum d'un graphe*, in *Théorie des graphes*, Rome, Dunod, 1966.
- [Ros74] Roşu Al. Al., *Teoria grafelor, Algoritmi, aplicaţii*, Ed. Militară, Bucureşti, 1974
- [Roy64] Roy B., *Cheminement et connexité dans les graphes à l'étude des problèmes d'ordonnancement*, communication à la 2^e conférence internationale sur la recherche opérationnelle Aix-en-Provence, septembre 1960, paru dans: *Les Problèmes d'ordonnancement, applications et méthodes*, Dunod, 1964, pp.109-125
- [Tar72] Trajan R., *Depth-first Search and Linear Graph Algorithms*, Siam J. Computing 1(1972), pp.146-160
- [TCI94] Toader T., Cataranciu S., Iacob E-M., *Probleme de teoria grafelor*, Lito. Univ."Babeş-Bolyai", Cluj-Napoca, 1994
- [Toa92] Toader T., *Elemente de teoria grafelor*, Lito.Univ."Babeş-Bolyai" Cluj-Napoca, 1992
- [ToC97] Toader T., Cozac I., *Branch and Bound Methods and Independent and Dominated Sets in Graphs*, Studia Univ."Babeş-Bolyai", Cluj-Napoca, 1997, vol.XLII (no.1), pp.17-28
- [ToS95] Toader T., Stoica F., *Some aspects of graphs planarity*, Studia Univ. "Babeş-Bolyai", Mathematica, vol.XL(no.3), 1995, pp.123-145
- [Tom75] Tomescu I., *Grafuri şi Programare liniară (Introducere elementară)*, Ed. Tehnică, Bucureşti, 1975
- [Tom82] Tomescu I., *Ce este teoria grafurilor?*, Ed. St. şi Enciclopedică, Bucureşti, 1982
- [Vos91] Voss H.-J., *Cycles and Bridges in Graphs*, Kluwer Academic Publishers, 1991
- [Yen72] Yen J.Y., *Finding the lengths of all shortest paths in N-node non-negative-distance complete network using $1/2N^3$ addition and N^3 comparisons*, Journal of A.C.M. vol.19(1972) no.3, pp.423
- [Whi67] White L.J., *A parametric study of matchings and coverings in weighted graphs*. Doctoral thesis. Tech. Report 06920-11-T. Department of Electrical Engineering, University of Michigan, 1967, Ann Arbor

